
IOTDM 文档

2017-08-23



目录

1 产品描述	1
1.1 介绍	1
1.2 优势	1
1.3 功能	2
2 产品定价	3
3 最佳实践	5
3.1 端到端配置示例	5
3.1.1 操作步骤	5
设备接入云端	5
设备向云端上报状态	9
通过设备影子控制设备状态	19
可视化展现物影子	21
设备在线状态的说明	25
3.2 基于物管理快速搭建一个应用	27
3.2.1 云端创建设备影子	28
在物管理中创建物模型	28
在物管理中创建物影子	29
3.2.2 向云端上报状态	30
3.2.3 APP端通过设备影子控制设备状态	31
3.2.4 服务端进行业务层管理	31

4 操作指南	33
4.1 介绍	33
4.2 在云端创建物影子	33
4.2.1 物模型	33
创建物模型	33
管理物模型	35
4.2.2 物影子	36
创建物影子	36
查看影子详情	37
获取连接配置	39
4.3 将实体设备接入云端	40
4.3.1 使用Baidu IoT Edge SDK	40
4.3.2 使用MQTT客户端SDK	40
更新设备状态到设备影子	40
从设备影子获取设备状态	42
通过设备影子控制设备状态	43
删除设备影子	43
订阅设备影子的变化	44
Device Profile	45
4.4 物管理操作指南 - V1.0	46
5 API参考 - V3	47
5.1 目录	47
5.2 介绍	48
5.2.1 简介	48
5.3 设备列表管理	48
5.3.1 创建单个设备	48
5.3.2 删除设备	49

5.3.3 获取设备Profile	50
5.3.4 获取设备View	51
5.3.5 获取及查询影子列表	52
5.3.6 获取设备接入详情	55
5.3.7 更新密钥	55
5.3.8 更新设备属性	56
5.3.9 更新单个设备View信息	58
5.3.10 更新单个设备注册表信息	60
5.3.11 重置设备影子	61
5.4 设备模板管理	62
5.4.1 创建模板	62
5.4.2 删除模板	64
5.4.3 获取模板列表	64
5.4.4 获取模板	66
5.4.5 更新模板	67
5.5 参数定义	69
5.5.1 DeviceList参数列表	69
5.5.2 DeviceProfile参数列表	69
5.5.3 DeviceBasicInfo参数列表	69
5.5.4 DeviceAttributes参数列表	70
5.5.5 DeviceView参数列表	70
5.5.6 DeviceViewAttributes参数列表	70
5.5.7 DeviceAccessDetail参数列表	70
SchemaProperty参数列表	71
Schema参数列表	71
6 API参考 - V2	73
6.1 介绍	73

6.1.1 简介	73
6.1.2 调用方式	73
概述	73
通用约定	73
公共响应头	74
响应状态码	74
通用错误返回格式	74
公共错误码	75
签名认证	76
签名生成算法	76
6.1.3 多区域选择	76
6.2 Endpoint管理	76
6.2.1 创建Endpoint	76
6.2.2 删除Endpoint	78
6.2.3 获取endpoint列表	78
6.3 设备列表管理	79
6.3.1 创建设备	79
6.3.2 删除设备	81
6.3.3 获取设备Profile	82
6.3.4 设备查询	84
6.3.5 获取设备接入详情	85
6.3.6 更新设备属性	86
6.3.7 更新设备注册表信息	88
6.3.8 禁用设备	89
6.3.9 启用设备	89
6.4 设备操作	90
6.4.1 设备重启	90
6.5 参数定义	91

6.5.1 DeviceProfile参数列表	91
6.5.2 DeviceAttributes参数列表	92
6.5.3 DeviceOperation参数列表	92
6.5.4 Device.Page参数列表	92
6.5.5 ThingDetail参数列表	93
6.5.6 EndpointInResponse参数列表	93
7 API参考 - V1	95
7.1 介绍	95
7.1.1 简介	95
7.1.2 调用方式	95
概述	95
通用约定	95
公共响应头	96
响应状态码	96
通用错误返回格式	97
公共错误码	98
签名认证	98
签名生成算法	98
7.1.3 多区域选择	98
7.2 设备列表管理	98
7.2.1 创建设备	98
7.2.2 删除设备	100
7.2.3 获取设备Profile	101
7.2.4 设备查询	102
7.2.5 获取设备接入详情	104
7.2.6 更新设备属性	105
7.2.7 更新设备注册表信息	106

7.2.8 禁用设备	107
7.2.9 启用设备	108
7.3 设备分组管理	108
7.3.1 创建设备组	108
7.3.2 删除设备组	110
7.3.3 修改设备组设置	110
7.3.4 查询设备组信息	111
7.3.5 查询指定设备组子节点列表	112
7.3.6 查询根目录下的设备组	113
7.3.7 查询device设备组	113
7.4 设备操作	113
7.4.1 设备重启	113
7.5 附录	114
7.5.1 DeviceProfile参数列表	114
7.5.2 DeviceAttributes参数列表	115
7.5.3 DeviceGroup参数列表	115
7.5.4 DeviceOperation参数列表	115
7.5.5 Device.Page参数列表	115
7.5.6 ThingDetail参数列表	116
8 Java SDK文档	117
8.1 V3	117
8.1.1 概述	117
8.1.2 安装SDK工具包	117
8.1.3 创建IotDmV3Client	118
8.1.4 设备管理	119
创建设备	119
删除设备	120

获取设备Profile	120
获取设备View	121
获取及查询影子列表	121
获取设备接入详情	122
更新密钥	122
更新设备属性	122
更新设备View	123
更新设备注册表信息	123
重置设备影子	124
8.1.5 模板管理	124
创建模板	124
删除模板	125
更新模板	125
获取模板详情	126
获取及查询模板列表	126
8.1.6 版本说明	126
v0.10.21	126
8.2 旧版	127
8.2.1 概述	127
8.2.2 安装SDK工具包	127
8.2.3 创建IotDmClient	128
8.2.4 设备列表管理	129
创建设备	129
删除设备	130
获取设备相关信息	130
获取设备信息列表	130
获取设备接入相关信息	130
更新设备Profile	131

更新设备注册表信息	131
8.2.5 设备操作	131
禁用设备	131
恢复设备	131
重启设备	132
8.2.6 设备组列表管理	132
8.2.7 版本说明	133
v0.10.13	133
9 常见问题	135
9.1 物管理常见问题	135
9.1.1 物管理中是否能够批量创建设备 ?	135
9.1.2 物管理与物接入的关系是什么 ?	135
9.1.3 编辑设备影子中, reported字段和desired字段代表什么意思 ?	135
9.1.4 编辑设备影子中, profileVersion是什么意思 ?	135
9.1.5 设备上传到云端的字段会覆盖设备影子吗 ?	136
9.1.6 物管理服务中的设备与物接入服务中的设备是否一一对应 ?	136
9.1.7 物管理服务可管理多少个实体设备 ?	136
9.1.8 设备影子中 “lastActiveTime” 字段是什么意思 ?	136

第1章

产品描述

1.1 介绍

说明

推荐用户使用新版物管理操作平台，新版操作平台为天工物联网的统一控制台入口，用户可通过该平台无缝对接其它天工服务，便于物管理服务与其它服务集成和对接。有关旧版（即物管理V1.0）操作指导，请参看[物管理操作指南 - V1.0](#)。

为了给您提供更高效的一站式物联网开发服务，物管理升级全新版本。在新版本的物管理中，您需要了解两个概念：物模型和物影子。

- 物模型由一个或多个属性构成，您可以用来表示一类设备。
- 物影子是物理世界的物在云端的『影子』或『数字双胞胎』，这个物的属性可以从物模型中继承过来。运行时，物将监控值上报给物影子，物影子会给这个物做一个状态暂存，天工其他产品或您的应用程序可以直接使用；与此同时，您也可以通过操作物影子来控制物。

物影子将作为天工产品使用的核，天工产品都将会读取物影子的数据（陆续对接上线）。另外，天工物联网会提供一个统一的控制台入口，让您更便捷搭建物联网的应用。

注意 推荐用户使用chrome, firefox, edge, safari这四类浏览器执行控制台操作，目前暂不支持IE浏览器。

1.2 优势

监测与控制

实时监测设备状态，并提供控制设备的能力。

在线操作

支持对设备在线操作，大大提高效率并降低成本。

识别认证

有效识别设备的合法身份，并将之纳入设备层级体系中进行管理。

强大的API

提供开放标准的API，可通过调用API实现控制台操作，方便第三方应用快速集成云端服务。

1.3 功能

同步设备状态至云端

实时同步设备状态至云端，用户可在云端对设备当前状态进行监控。

设备上线后自动同步云端配置

在云端创建虚拟设备并录入配置信息；待实体设备上线后，可自动请求并同步云端配置。

云端主动刷新设备配置

用户可在云端主动管理和配置实体设备，实时下发配置，变更和调整设备状态。

通过设备组管理设备层级

物管理提供基于树结构的设备层级管理。例如可设置一个设备属于”/百度/北京/科技园/K2/F5/空调组”，其中“百度”、“北京”、“科技园”等都是层级结构中的一个节点，用户可基于设备组对设备进行管理。

第2章

产品定价

百度云物管理服务为免费服务，物管理通过MQTT协议更新设备影子时产生的请求/响应消息将计入物接入服务的当月套餐额度。关于物接入的计费标准，请查看[物接入产品定价](#)。

第3章

最佳实践

3.1 端到端配置示例

本文档介绍了物管理端到端配置示例，帮助用户理解和使用物管理。

3.1.1 操作步骤

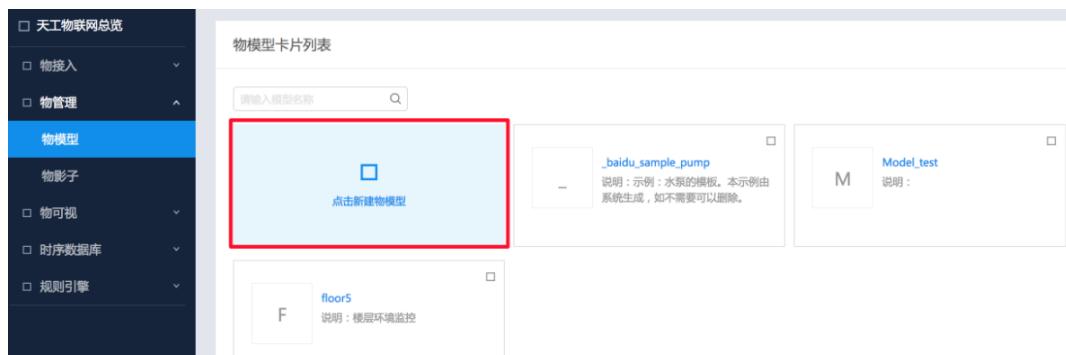
本示例分成以下几步：

1. [设备接入云端](#)
2. [设备向云端上报状态](#)
3. [通过设备影子控制设备状态](#)
4. [可视化展现物影子](#)
5. [设备在线状态的说明](#)

[设备接入云端 在物管理中创建物模型](#)

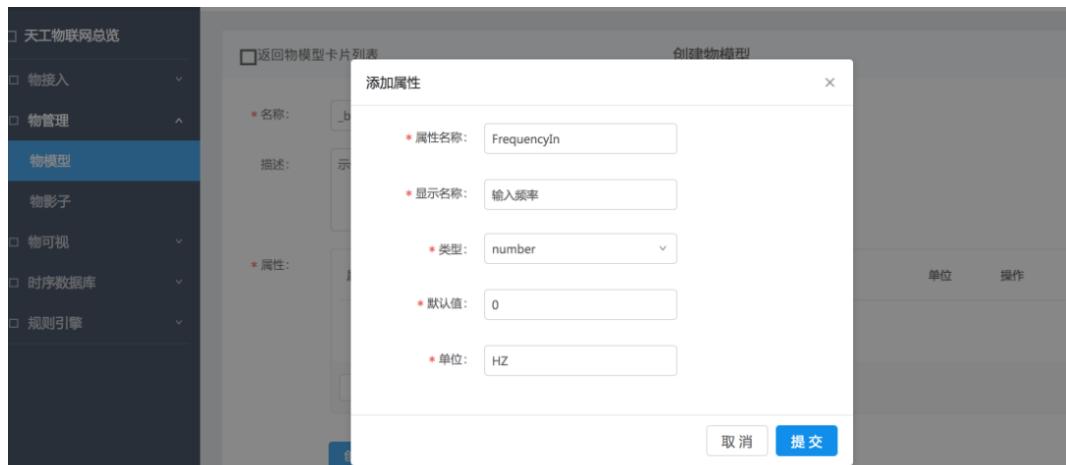
通过物模型可以为设备定义一套属性模板，在创建物影子时可以引用该模板，实现业务的快速部署。具体操作方法如下：

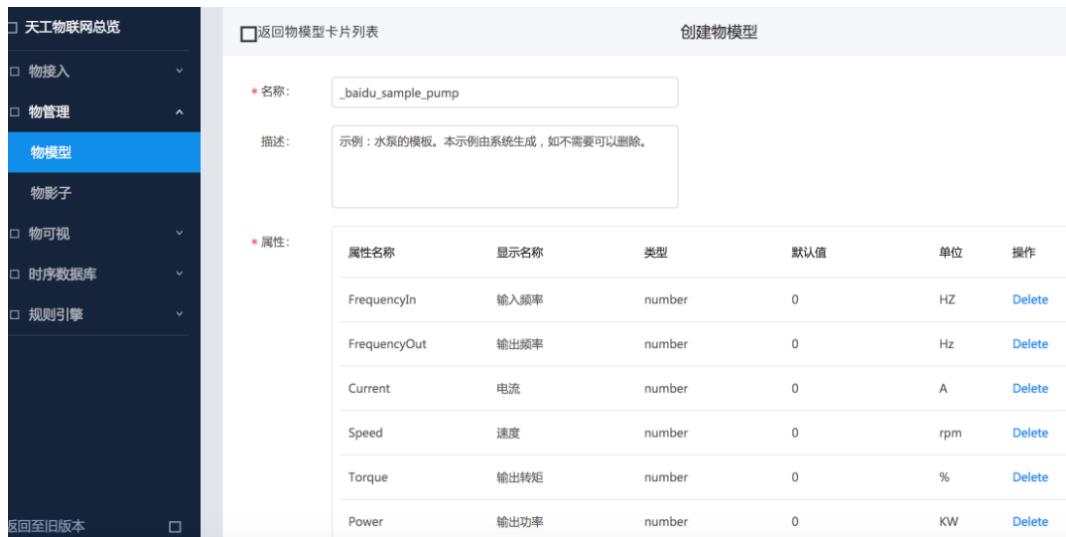
1. 登录百度云官网，点击右上角的“管理控制台”，快速进入控制台界面。在控制台中，选择“产品服务>物管理 IoT Device>物模型”，在物管理中创建物模型。
2. “[点击新建物模型](#)”进入物模型配置界面。



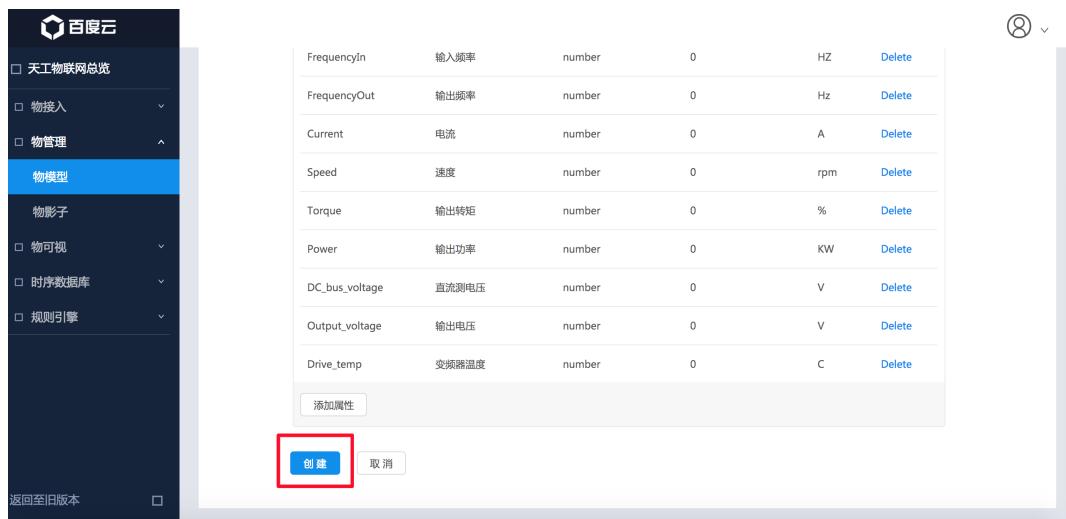
3. 填写物模型配置，包括名称、描述和属性。点击添加属性为物模型新增一条属性。

“添加属性”中物模型的每一条属性需要填写属性名称、显示名称、类型、默认值、单位。其中属性名称要求是英文字符。注意：若创建成功后物模型名称和属性名称是无法修改的，同一物模型下属性名称必须唯一。





完成属性配置后，点击“创建”，完成物模型创建。



在物管理中管理物模型

创建物模型以后，可以对以下信息进行编辑修改，包括：物模型描述信息，删除或新增属性，属性描述、默认值、单位。

1. 选择需要修改的物管理模型卡片，以物模型示例_baidu_sample_pump为例，点击卡片进入详情页面。



2. 点击“编辑”进入编辑模式，此时可对物模型进行属性编辑等操作。

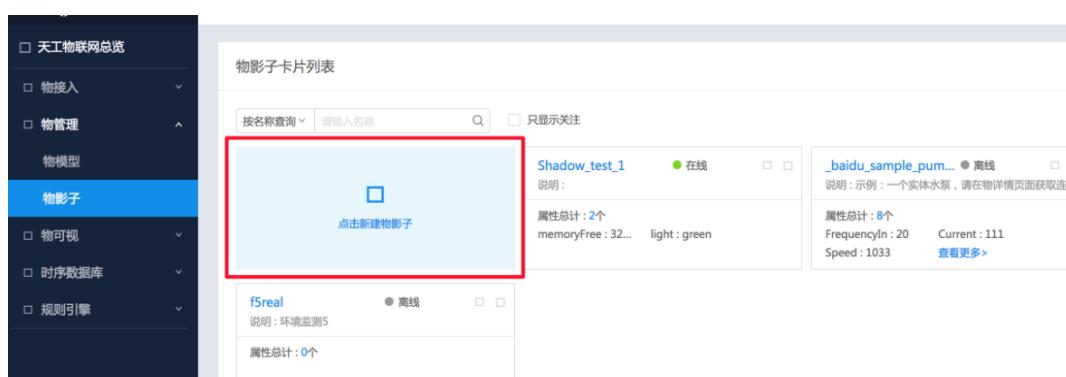


3. 完成操作后点击左下角“保存”使配置生效。

在物管理中创建物影子

物影子与实体设备一一对应，在创建物影子之前，必须先创建物模型。

1. 登录百度云官网，点击右上角的“管理控制台”，快速进入控制台界面。在控制台中，选择“产品服务>物管理 IoT Device>物影子”，在物管理中创建物影子。
2. “点击新建物影子”进入物影子配置界面。



3. 填写物影子配置，包括名称、描述和选择对应的物模型。此步骤中创建物影子 \baidu\sample\pump\instance为例，对应的物模型为上一个步骤中创建的\baidu\sample_pump。



4. 点击“创建”完成物影子创建。

5. 物影子介绍：物影子与实体设备一一对应，用户可通过物影子查看对应的实体设备的属性和状态等信息，如设备id、设备名称等。物影子的属性对应一个json文档，如下例所示：

```
"device": {
  "reported": {
    "FrequencyIn": 300,
    "FrequencyOut": 400
  },
  "desired": {
    "Current": 100,
    "Speed": 2000
  },
  "profileVersion": 102
}
```

“device”中的属性需要同步到对应的设备实体。其中，“reported”表示设备端通过MQTT连接汇报到物管理的设备状态。“desired”表示控制端希望控制设备变换到的目标状态。

设备向云端上报状态 物影子作为现实物理世界的设备在云端上的『影子』或『数字双胞胎』，这个设备的属性可以从物模型中继承过来，在云端创建物影子以后，可以使得设备向云端上报状态，并且可以对设备信息进行编辑修改，通过才做物影子来控制设备。

获取物影子的连接配置

创建物影子后，系统将自动生成与该物影子对应的物接入连接配置，包括endpoint实例地址、设备名称和密钥信息。该连接配置信息可用于将设备连接至物管理，实体设备与物影子一一对应。

具体操作如下：

1. 登录百度云官网，点击右上角的“管理控制台”，快速进入控制台界面。在控制台中，选择“产品服务>物管理 IoT Device>物影子”，点击具体的物影子卡片，选择“物详情”。

The screenshot shows the Baidu Cloud Management Console interface. On the left is a sidebar with various service categories like 天工物联网总览, 物接入, 物管理, 物模型, and 物影子. Under '物影子', a specific device card for '_baidu_sample_pump_instance' is displayed. The card has tabs for '返回物影子列表' and '物详情'. The '物详情' tab is currently active, indicated by a blue underline and a red box around it. Below the tabs, there are fields for '名称' (Name), '描述' (Description), and '来自模型' (From Model). At the bottom right of the card are two buttons: '编辑' (Edit) and '连接配置' (Connection Configuration).

2. 点击“连接配置”获取信息。

- 点击“更新密钥”后，原有密钥将失效，会导致已经接入的设备断开连接。
- 点击“下载”，可将相关连接配置信息保存至本地。

This screenshot shows the 'Connection Configuration' dialog box for the '_baidu_sample_pump_instance' device shadow. The dialog contains fields for 'TCP Address' (tcp://4fbf4731a8254b508cc8751660235f2c.mqtt.iot.gz.baidubce.com:1883) and 'SSL Address' (ssl://4bf4731a8254b508cc8751660235f2c.mqtt.iot.gz.baidubce.com:1884). A warning message at the top states: '更新密钥后会导致原有的连接断开，需要用新的密钥连接，请谨慎操作！'. At the bottom of the dialog are three buttons: '编辑' (Edit), '更新密钥' (Update Key) which is highlighted with a red box, and '下载' (Download) which is also highlighted with a red box.

将实体设备接入云端

1. 接入方式1：使用Baidu IoT Edge SDK

说明

推荐用户优先使用百度云提供的SDK。

百度云提供的SDK封装了MQTT客户端SDK，屏蔽了MQTT客户端SDK的细节和topic信息，可以帮助用户实现业务的快速部署。

用户可以在设备端安装百度云提供的SDK，并配置[连接信息](#)，实现设备与百度云

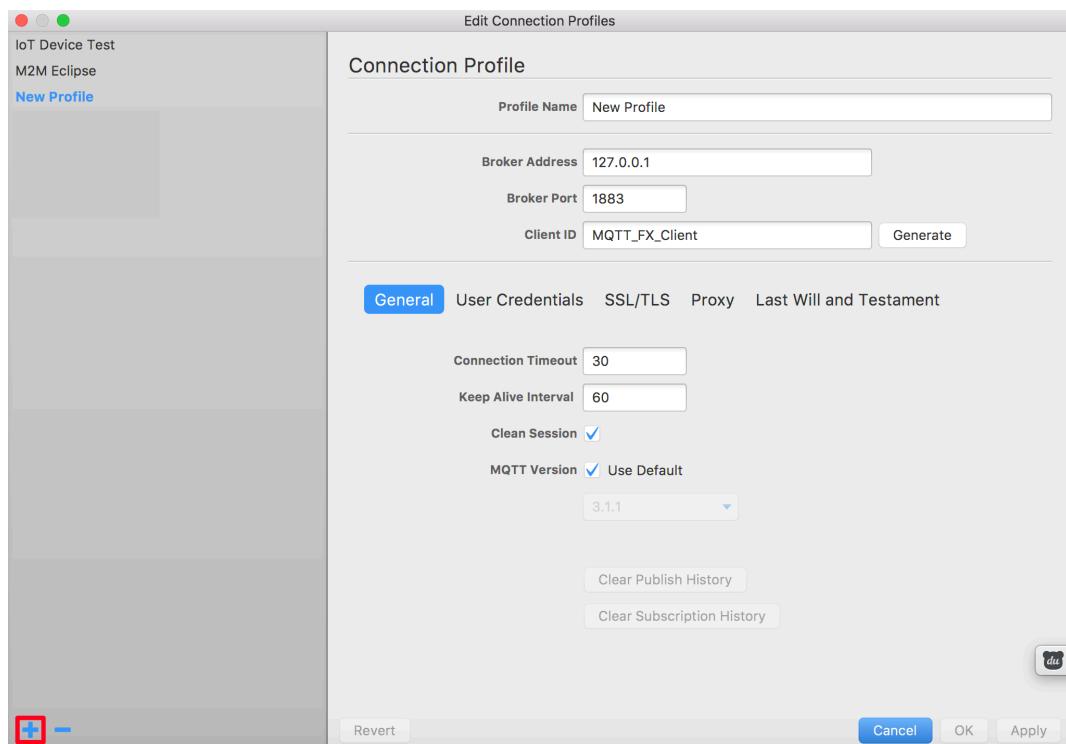
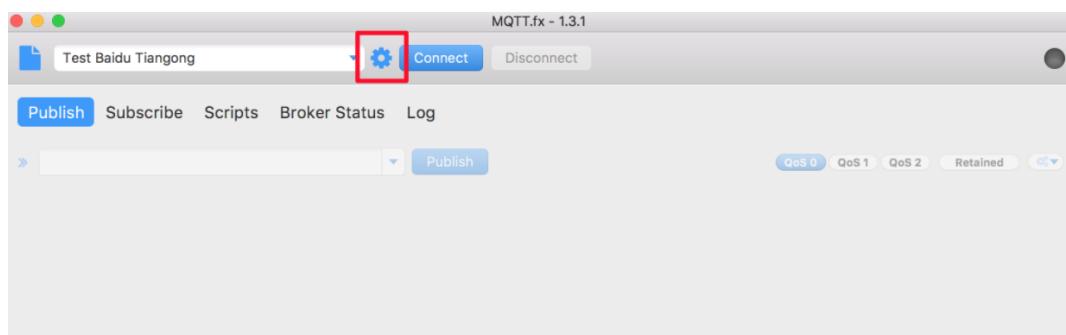
物管理的快速对接。
有关 IoT Edge SDK 的下载和安装, 请查看: <https://github.com/baidu/iot-edge-sdk-for-iot-parser/releases/tag/v1.0.1>

2. 接入方式2: 使用MQTT客户端SDK

如果设备端已经调用了Paho (即MQTT Client SDK), 可以通过与特定的topic通信实现与百度云的对接。在物管理中定义了系统topic用于物管理服务和设备端基于物接入服务以及MQTT协议进行通信。

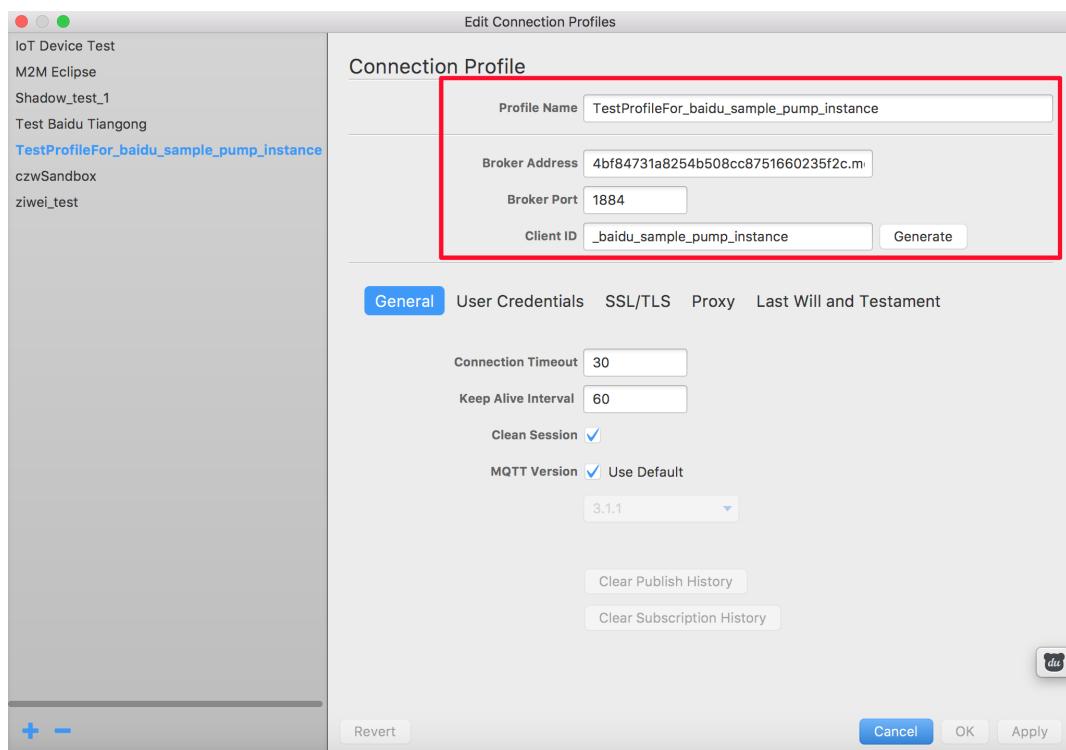
本示例使用mqtt.fx客户端模拟设备端, 与物影子示例\baidu\sample\pumpinstance相连。

- 关于MQTT.fx客户端在物接入部分有相关介绍: [MQTT.fx](#)
 - 登录[MQTT.fx下载页面](#), 找到适合的版本下载并安装MQTT.fx客户端
1. 打开MQTT.fx, 选择设置项进入设置页面, 点击左下角“+”按键, 创建一个新的Connection Profile配置文件。

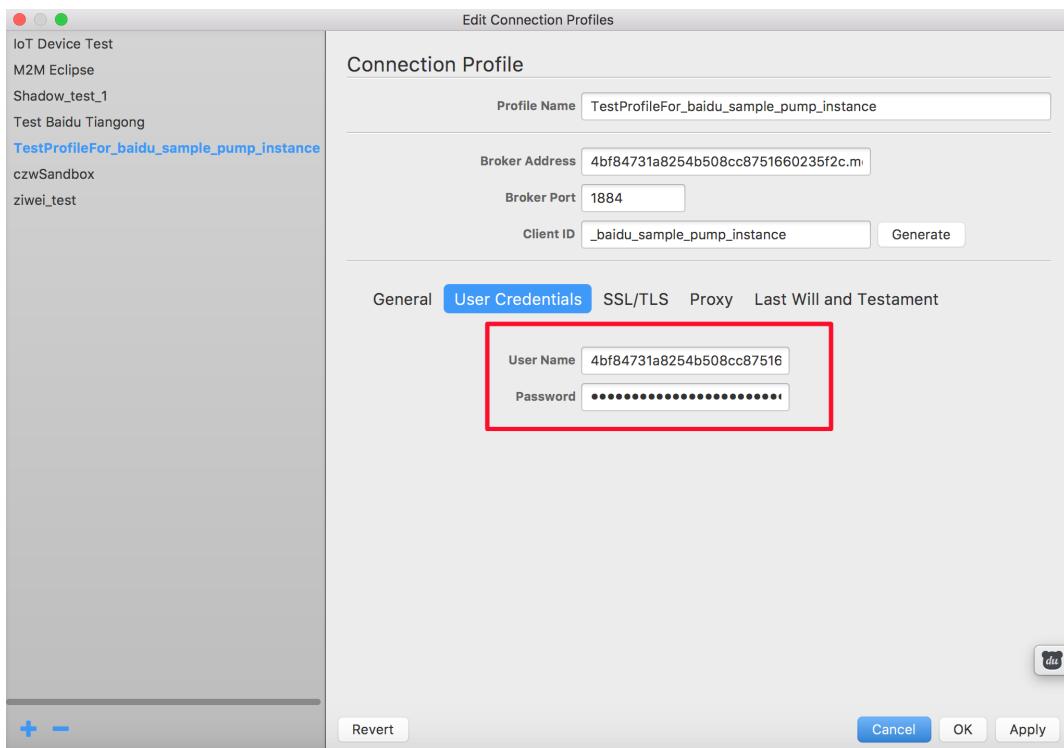


2. 根据物影子示例_baidu_sample_pump_instance的"连接配置"填写Connection Profile的相关信息

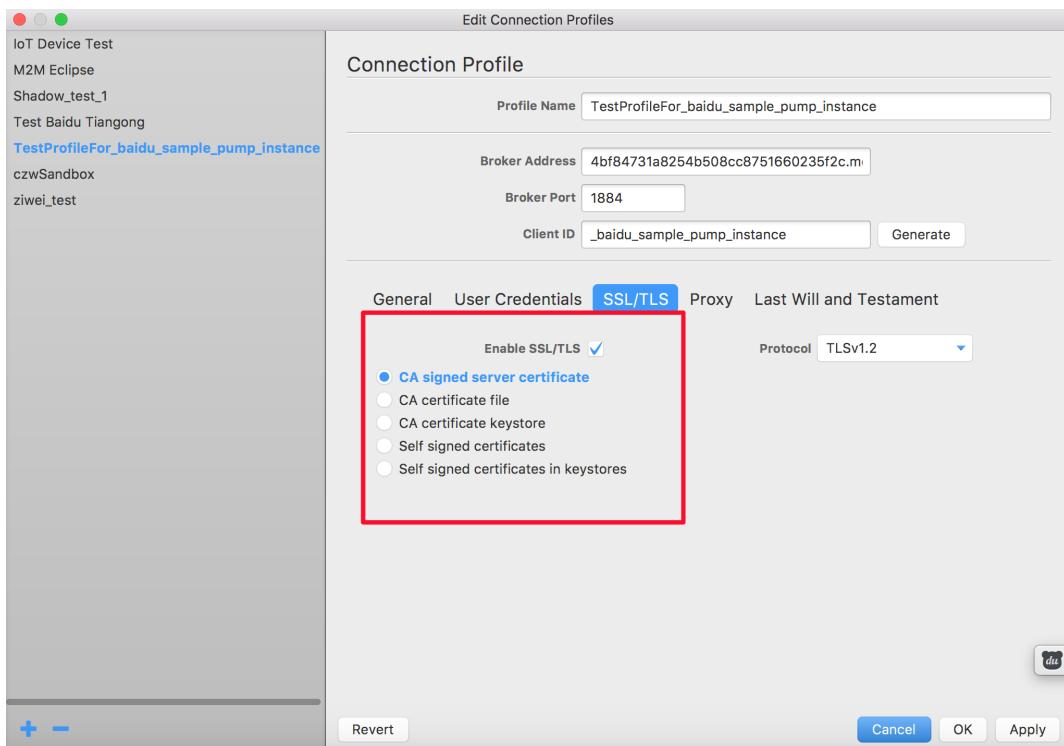
- * **Profile Name:** 该配置文件名，用户自定义。
- * **Broker Address:** 物影子连接配置中Address的hostname，例如"4bf84731a8254b508cc8751660235f2c.mqtt.iot.baidu.com"
- * **Broker Port:** ssl加密连接方式，端口使用1884；tcp不加密连接，端口使用1883。
- * **Client ID:** 客户端ID。
- * 支持"a-z","0-9","_","-"字符，且不能大于128bytes，UTF8编码，在MQTT.fx客户端中支持随机生成。
- * 在同一个实例下，每个实体设备需要有一个唯一的ID，不同实体设备使用同一个client id建立连接会导致其它连接下线。
- * 目前只有当Client ID与物影子名称一致时（例如物影子为"_baidu_sample_pump_instance"，设备端Client ID同为"_baidu_sample_pump_instance"），控制台物影子的在线、离线状态由同名称的Client ID客户端反映相同的在线、离线状态，其他的Client ID所在的客户端无法反映



a. 选择User Credential，输入物影子连接配置中的name作为user name，key作为password。

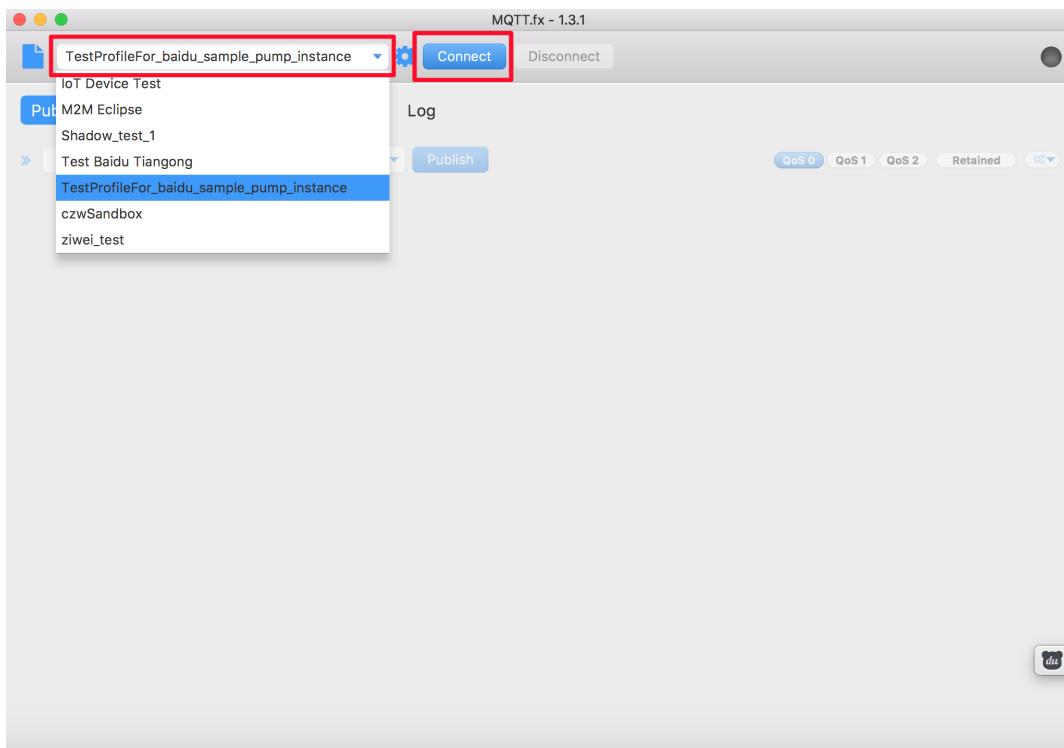


b.选了SSL/TLS，如果您选择SSL安全认证方式连接物影子，需要配置SSL/TLS安全认证，勾选 Enable SSL/TLS，选择CA signed server certificate认证。

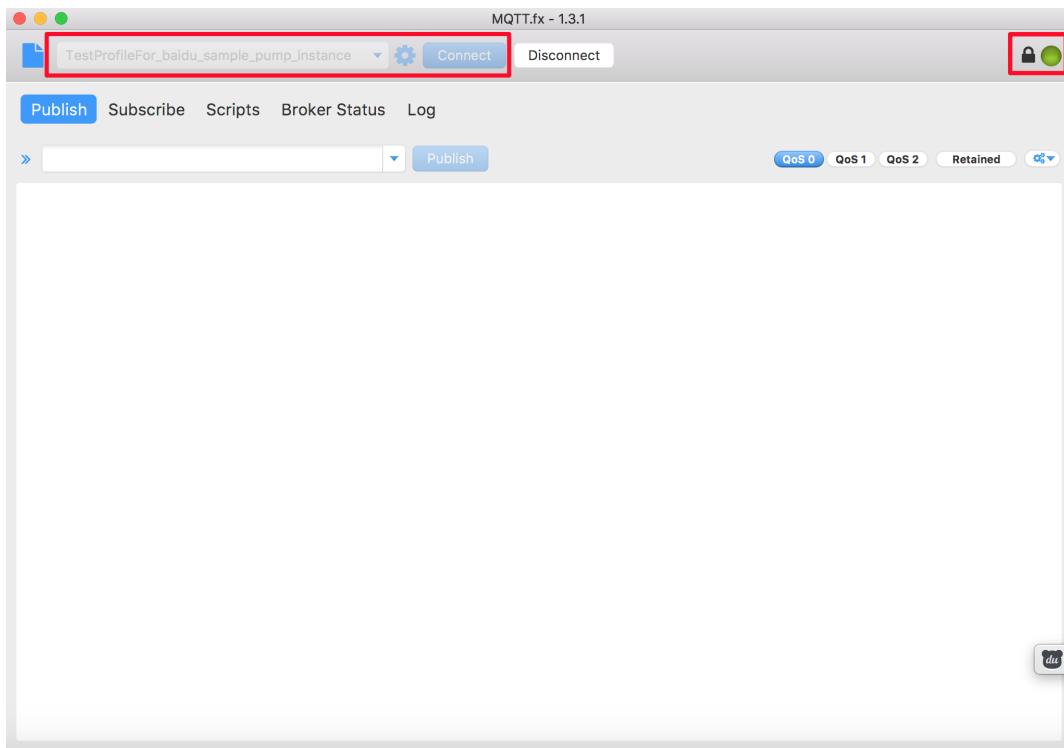


若您选择TCP连接，无需配置SSL安全认证，忽略本步骤即可。

c.点击右下角的“OK”按键，完成客户端与云端的接入配置。返回MQTT.fx客户端界面，选择刚刚创建好的配置文件，点击“connect”按键连接服务。



d.左侧下拉框与连接按钮置灰，右侧状态灯变绿，表示连接成功。

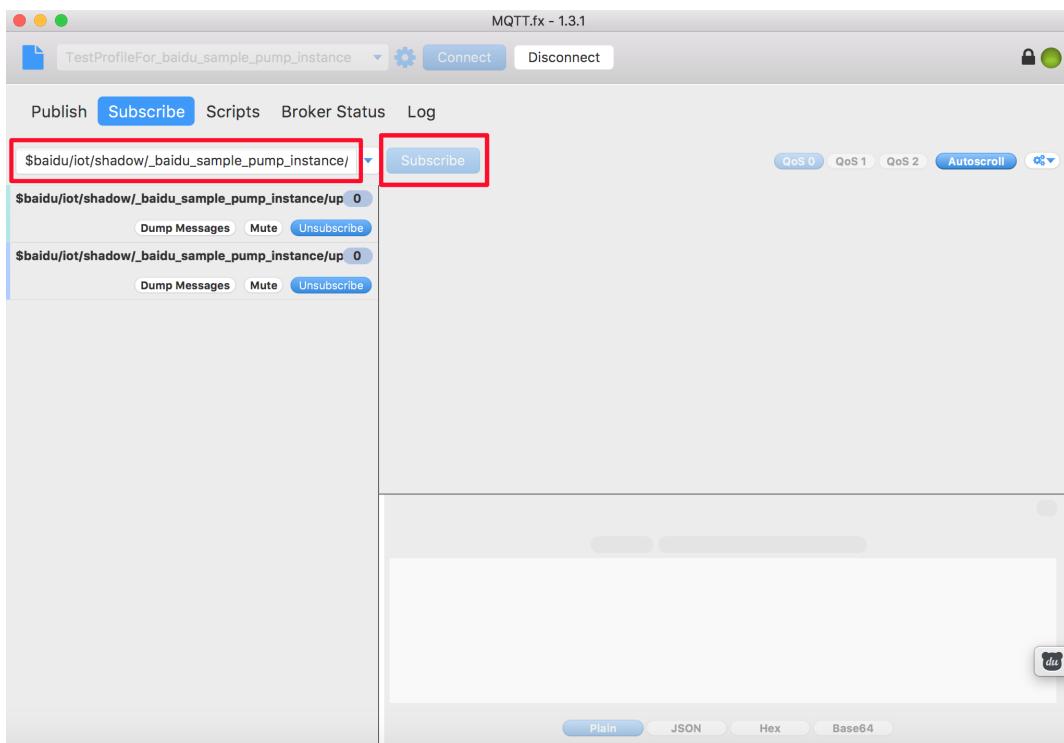


更新设备状态到设备影子

1. 订阅主题：获取设备影子更新状态情况

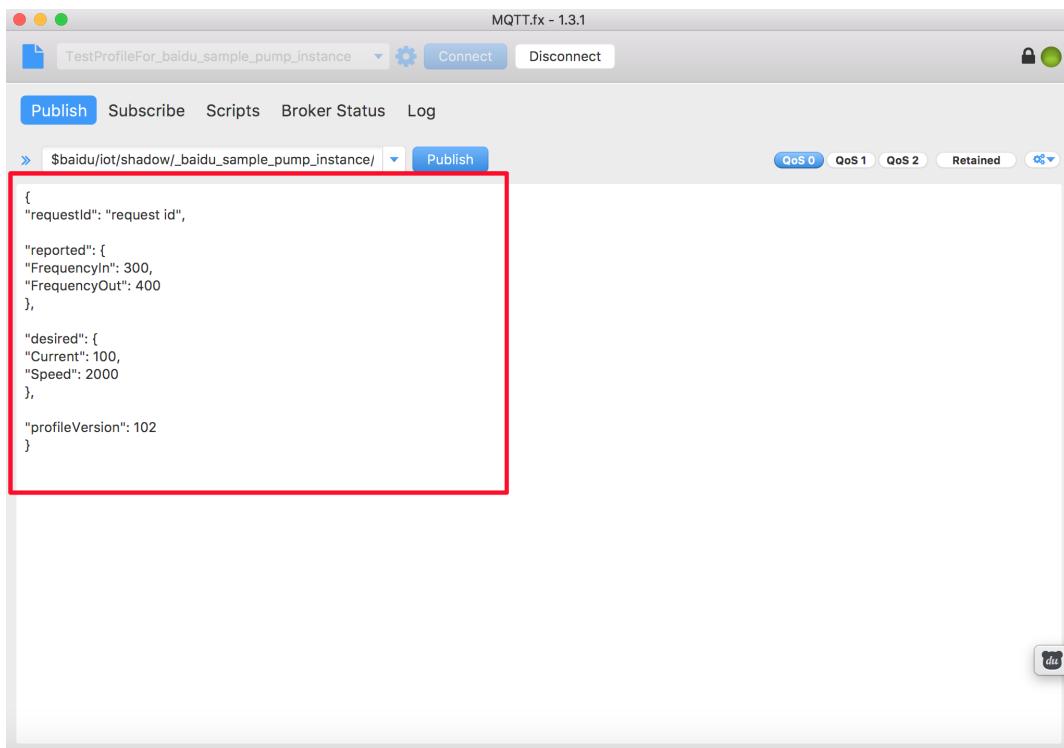
- 选择Subscribe标签，输入需要订阅的Topic名称，点击“Subscribe”按钮。例如\$baidu/iot/shadow/\baidu\sample\pump\instance/update/accepted表示订

阅 “\baidu\sample\pump\instance” 这个设备影子更新成功的信息，\$baidu/iot/shadow/\baidu\sample\pump\instance/update/rejected表示订阅 “\baidu\sample\pump\instance” 这个设备影子更新失败的信息。



2. 发布主题：将消息发布到主题\$baidu/iot/shadow/{deviceName}/update，可以实现将设备状态更新到设备影子，并且通过订阅主题查看更新结果。

- 选择 Publish 标签，输入需要发布的 Topic 名称、消息内容，点击“Publish”按钮。例如 \$baidu/iot/shadow/\baidu\sample\pump\instance/update 表示发布消息来更新 “\baidu\sample\pump\instance” 这个设备影子的设备状态。



- 示例消息内容：

```
{  
  "requestId": "request id",  
  "reported": {  
    "FrequencyIn": 300,  
    "FrequencyOut": 400  
  },  
  "desired": {  
    "Current": 100,  
    "Speed": 2000  
  },  
  "profileVersion": 102  
}
```

- 相关字段解释如下：

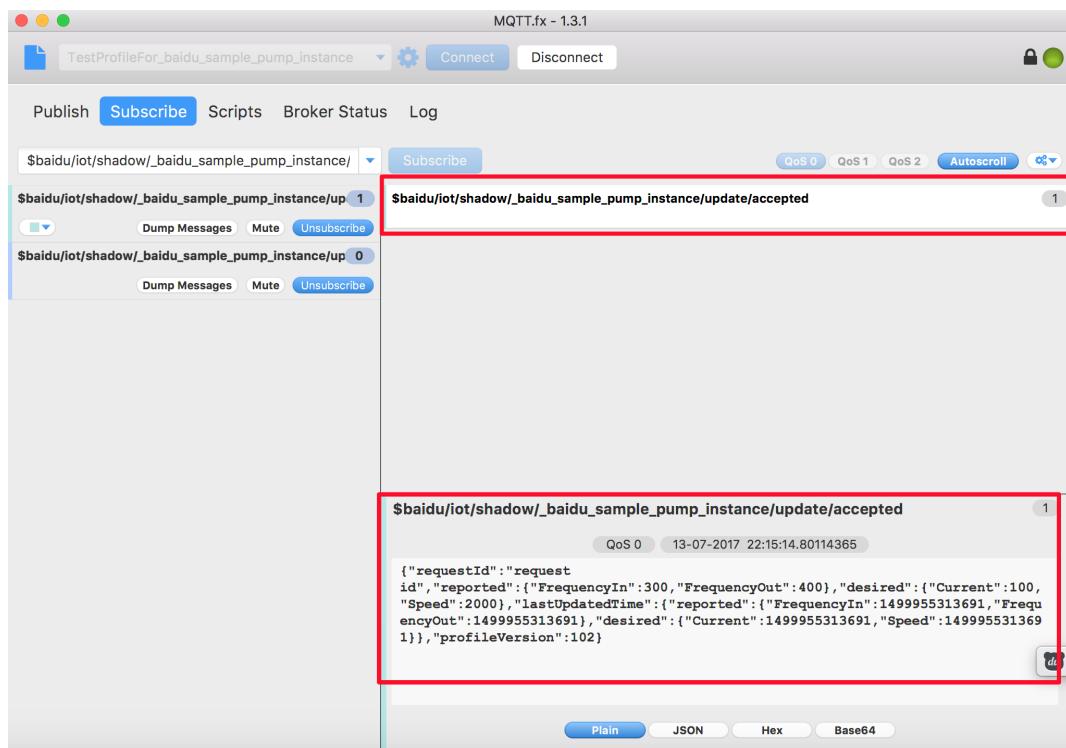
- * "requestId"为请求的唯一标识符，每一个请求的requestId是唯一的，可随机生成。
- * "reported"为可选字段，表示设备影子的"reported"中的相关属性需要更新。
- * "desired"为可选字段，表示设备影子的"desired"中的相关属性需要更新。
- * "profileVersion"为可选字段，当未指定profileVersion时，物管理接收设备影子更新请求后，会将profileVersion自动加1；若指定profileVersion，物管理会检查请求中的profileVersion是否大于当前的profileVersion。只有在大于的情况下，物管理才会接受设备端的请求，更新设备影子，并将profileVersion更新到相应的版本。

- 更新设备影子适用于两种应用场景：

a. 设备同步状态到物管理服务。设备在状态发生变化时，将实时的状态同步到物管理服务，包括状态的自动变化以及设备反控后状态的变化。更新设备状态，通常更新"reported"字段中的相关属性。对于反控后更新状态，设备可以用实时状态同时更新该属性的"reported"和"desired"中的值。

b. 通过MQTT协议反控设备状态。如果需要通过MQTT协议反控设备属性，可以通过更新"desired"字段实现。当物管理接收到"desired"相关属性的更新后，会diff设备影子中"reported"和"desired"相关字段，将diff后的结果发送到delta主题。设备端通过订阅delta主题，可将设备状态同步到"desired"的状态。状态反控后，更新设备影子，使"reported"和"desired"的值一致。物管理对设备的反控请参考通过设备影子控制设备状态。

- 发布成功后可以在Subscribe中订阅的主题看到结果。



- 同时在控制台中可以看到物影子状态更新，与publish的消息内容一致。

The screenshot shows the Baidu IoT Cloud console with the following details:

- Device Information:**
 - 名称: _baidu_sample_pump_instance
 - 描述: 示例：一个实体水泵，请在物详情页面获取连接配置，并添加到设备端SDK中，即可完成设备端与云端的连接。本示例由系统生成，如不需要可以删除。
 - 创建时间: 2017-07-06 14:14:54
 - 最后活跃时间: 2017-07-13 22:15:13
 - 物影子版本号: 102
- Shadow Attribute Table:**

属性名称	显示名称	类型	默认值	单位	当前值	修改时间	期望值	发送时间
FrequencyIn	输入频率	number	0	Hz	300	2017-07-13 22:15:13		2017-07-13 22:23:10
FrequencyOut	输出频率	number	0	Hz	400	2017-07-13 22:15:13		2017-07-13 22:23:10
Current	电流	number	0	A	111	2017-07-06 14:14:54	100	2017-07-13 22:15:13
Speed	速度	number	0	rpm	1033	2017-07-06 14:14:54	2000	2017-07-13 22:15:13

3. 云端对设备数据进行更新

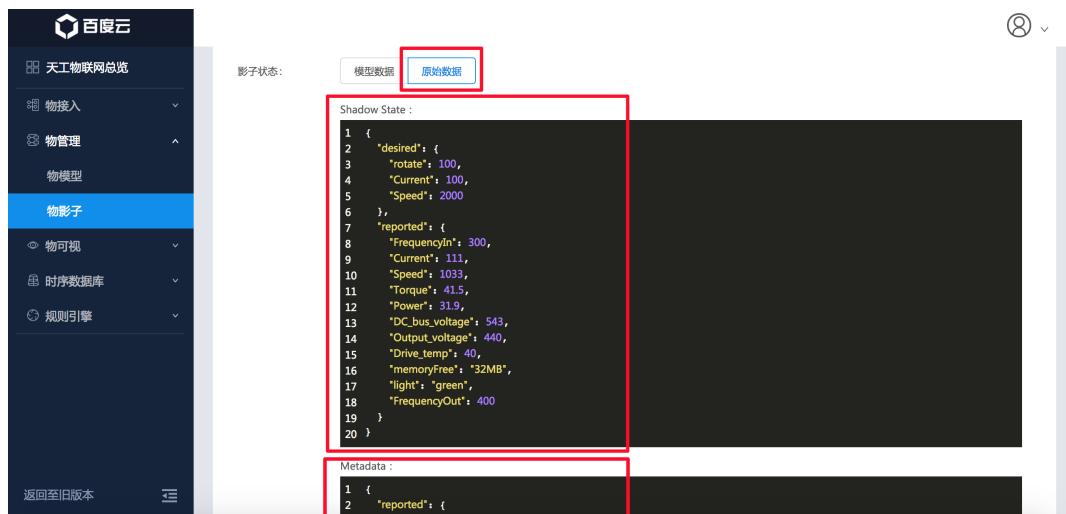
- 在物影子详情界面，用户可以看到以下信息：
- 模型数据，通过图表展示该物影子关联的物模型定义的所有属性，如下图所示：（在对应的物模型中存在的字段才展示在模型数据中。如果用户reported了不在该物模型定义的属性，则通过“原始数据”查看。）
- 相关字段解释如下：
 - 当前值：指设备最后一次上报的该属性的reported值。如果没有上报则为空。
 - 修改时间：指设备最后一次更新该属性的值。
 - 期望值：指通过控制台或者应用程序性修改的desired值。如果没有。期望值则为空。
 - 发送时间：指该属性最后一次有desired值修改的时间

属性名称	显示名称	类型	默认值	单位	当前值	修改时间	期望值	发送时间
FrequencyIn	输入频率	number	0	Hz	300	2017-07-13 22:15:13		2017-07-13 22:35:15
FrequencyOut	输出频率	number	0	Hz	400	2017-07-13 22:15:13		2017-07-13 22:35:15
Current	电流	number	0	A	111	2017-07-06 14:14:54	100	2017-07-13 22:15:13
Speed	速度	number	0	rpm	1033	2017-07-06 14:14:54	2000	2017-07-13 22:15:13
Torque	输出转矩	number	0	%	41.5	2017-07-06 14:14:54		2017-07-13 22:35:15
Power	输出功率	number	0	KW	31.9	2017-07-06 14:14:54		2017-07-13 22:35:15
DC_bus_voltage	直流侧电压	number	0	V	543	2017-07-06 14:14:54		2017-07-13 22:35:15

* 点击“编辑”，对物影子的属性进行编辑。

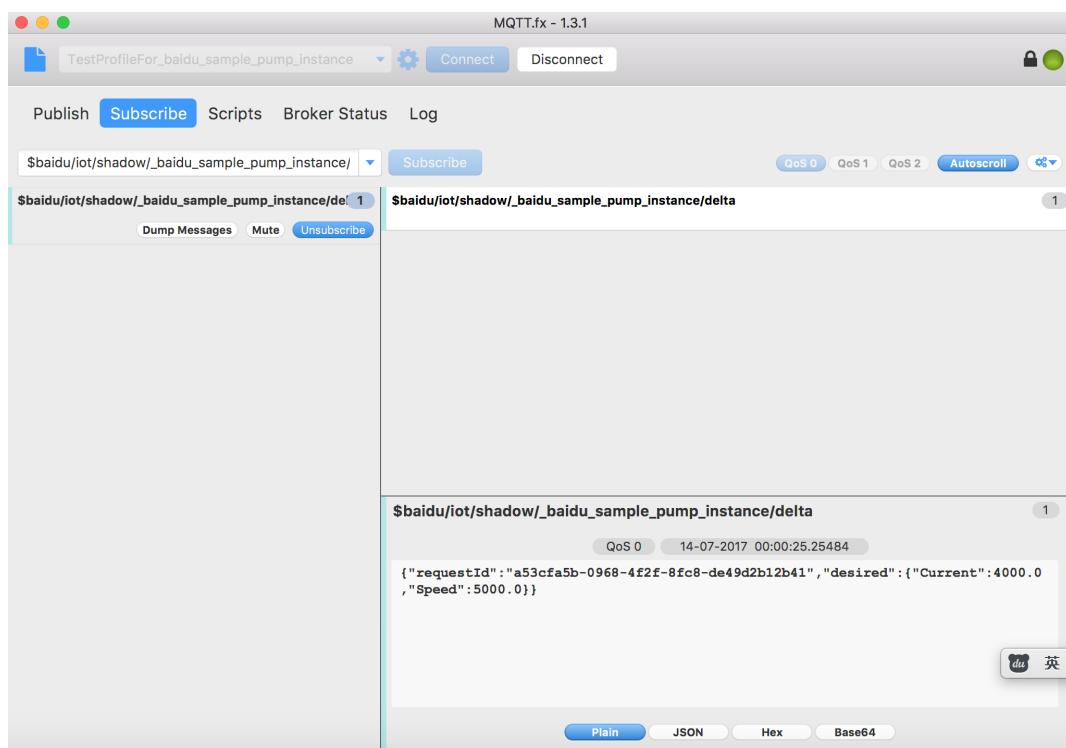
属性名称	显示名称	类型	默认值	单位	当前值	修改时间	期望值	发送时间
FrequencyIn	输入频率	number	0	Hz	300	2017-07-13 22:15:13	<input type="text"/>	2017-07-13 22:59:38
FrequencyOut	输出频率	number	0	Hz	400	2017-07-13 22:15:13	<input type="text"/>	2017-07-13 22:59:38
Current	电流	number	0	A	111	2017-07-06 14:14:54	100	2017-07-13 22:15:13
Speed	速度	number	0	rpm	1033	2017-07-06 14:14:54	2000	2017-07-13 22:15:13
Torque	输出转矩	number	0	%	41.5	2017-07-06 14:14:54	<input type="text"/>	2017-07-13 22:59:38
Power	输出功率	number	0	KW	31.9	2017-07-06 14:14:54	<input type="text"/>	2017-07-13 22:59:38
DC_bus_voltage	直流侧电压	number	0	V	543	2017-07-06 14:14:54	<input type="text"/>	2017-07-13 22:59:38

* 点击左下角“保存”后，物管理将期望值下发至设备端，用户也可以通过编辑原始数据desired中的属性实现对设备的远程控制。
 * 原始数据：在物影子详情界面点击“原始数据”，用户可以看到以下信息，即该设备影子的原始数据，用于查看设备状态或远程控制设备。



通过设备影子控制设备状态

1. 控制端可以通过MQTT协议更新设备影子中的‘desired’字段，达到控制设备的目的。物管理在接受到‘desired’字段更新后，会比较‘reported’字段和‘desired’字段之间的差异，并将diff结果发送到主题'\$baidu/iot/shadow/{device-Name}/delta'。
2. 在MQTT.fx客户端中，Subscribe主题\$baidu/iot/shadow/_baidu\sample\pump\instance/delta



3. 在控制台物影子详情中，编辑设备影子的模型数据的期望值，点击保存。

例如，当前‘reported’中的‘Current’字段为111，控制端将‘desired’中的‘Current’字段更新为4000，此时物管理会通过delta主题反控设备

The screenshot shows the Baidu IoT Cloud Model Shadow Management interface. On the left sidebar, under '物影子' (Device Shadow), there are several properties listed: FrequencyIn, FrequencyOut, Current, and Speed. The 'Current' row has its value field highlighted with a red box. The 'Speed' row also has its value field highlighted with a red box.

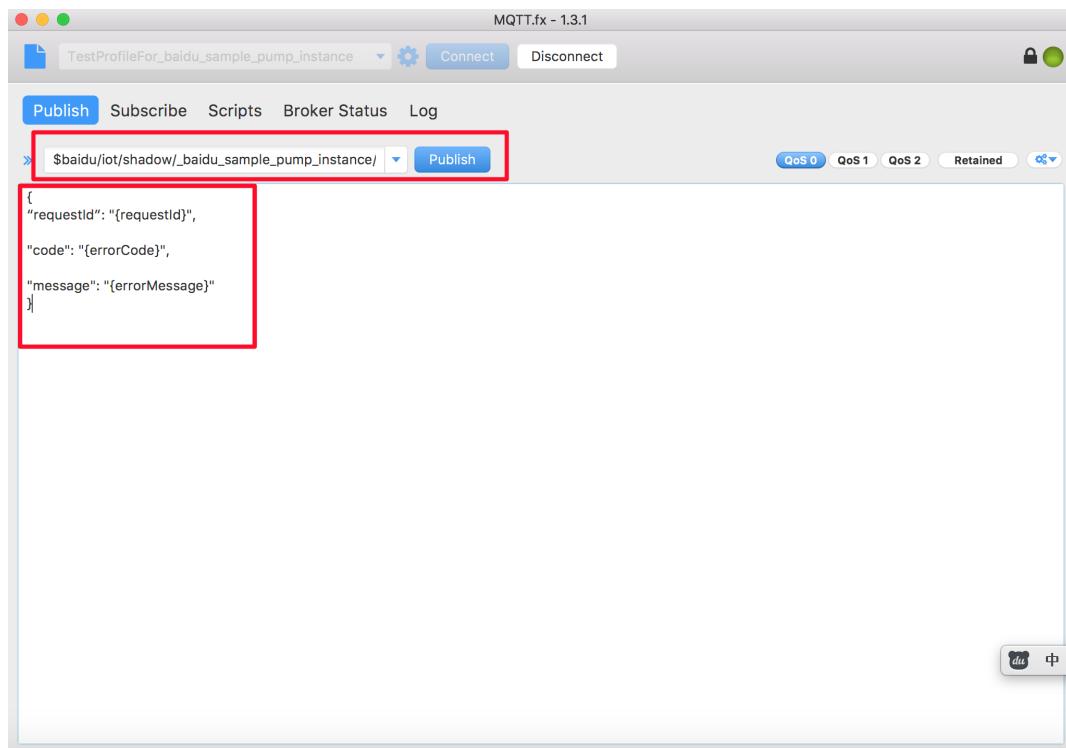
属性名称	显示名称	类型	默认值	单位	当前值	修改时间	期望值	发送时间
FrequencyIn	输入频率	number	0	Hz	400.0	2017-07-13 23:47:13	<input type="text"/>	2017-07-14 00:03:07
FrequencyOut	输出频率	number	0	Hz	400	2017-07-13 22:15:13	<input type="text"/>	2017-07-14 00:03:07
Current	电流	number	0	A	111.0	2017-07-14 00:00:23	<input type="text" value="4000.0"/>	2017-07-14 00:00:23
Speed	速度	number	0	rpm	1033.0	2017-07-14 00:00:23	<input type="text" value="5000.0"/>	2017-07-14 00:00:23

The screenshot shows the MQTT.fx 1.3.1 client interface. It is connected to a broker and subscribed to the topic '\$baidu/iot/shadow/_baidu_sample_pump_instance/delta'. A message has been received, which is highlighted with a red box. The message content is:

```

$baidu/iot/shadow/_baidu_sample_pump_instance/delta
QoS 0 14-07-2017 00:00:25.25484
{"requestId": "a53cfa5b-0968-4f2f-8fc8-de49d2b12b41", "desired": {"Current": 4000.0, "Speed": 5000.0}}
  
```

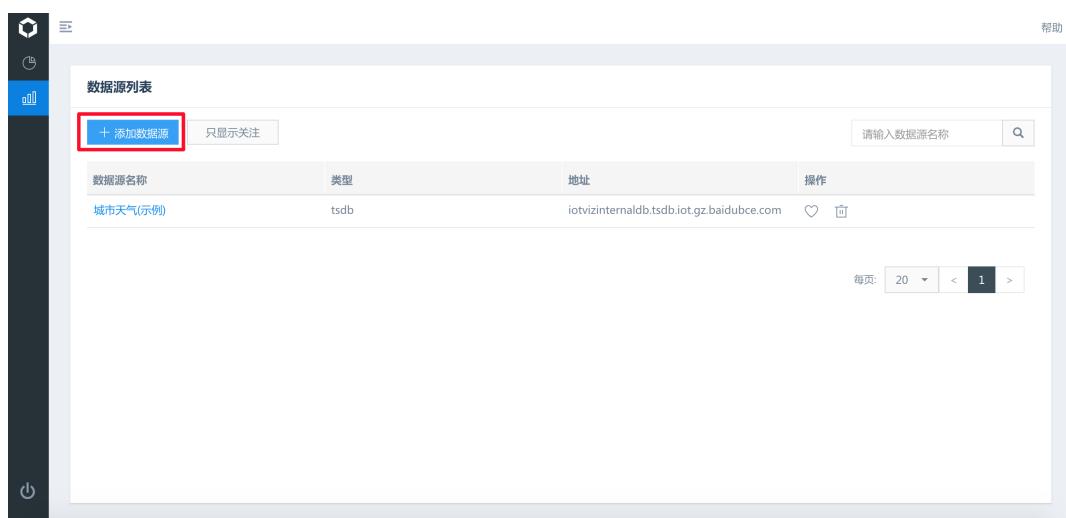
- 若设备更新状态失败，可在设备端Publish主题\$baidu/iot/shadow/myDeviceName/delta/rejected，将相关错误信息发送到物管理：



可视化展现物影子 物可视 (IoT Visualization) 使用交互式的可视化设计器可无缝获取物影子数据。有关物可视更多的信息请见产品文档。

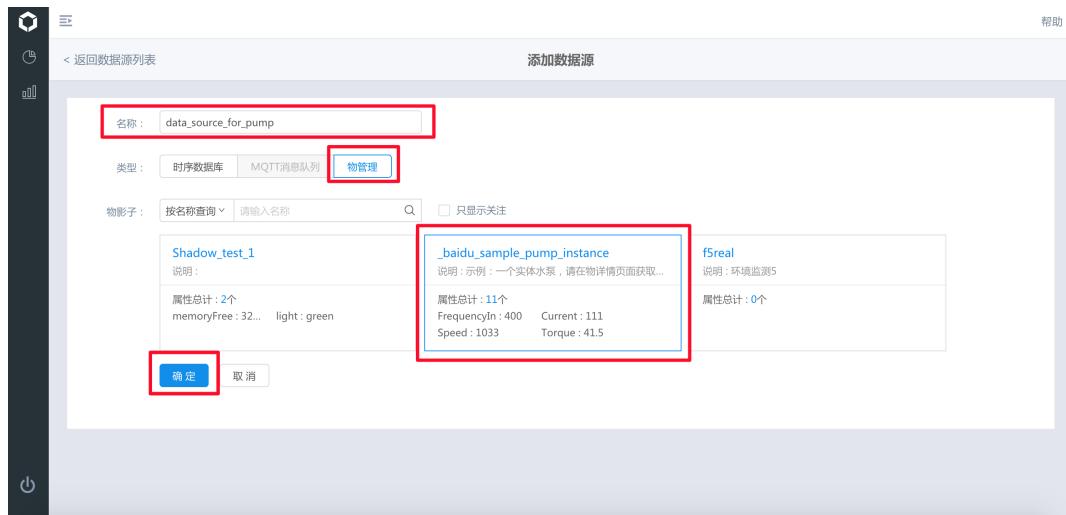
创建数据源

- 在控制台中，选择“产品服务>物可视 IoT Visualization>数据源”，进入物可视服务的数据源列表。点击“添加数据源”。



- 配置数据源信息，以接入物管理中的\baidu\sample\pump\instance为例，点击“确定”完成数据源信息配置。具体信息包括：
 - 名称：指定数据源名称，支持中英文字符、数字和下划线，支持2-32个字符。

- **类型：**物可视支持读取时序数据库、MQTT消息队列和物管理中的数据。



创建数据流

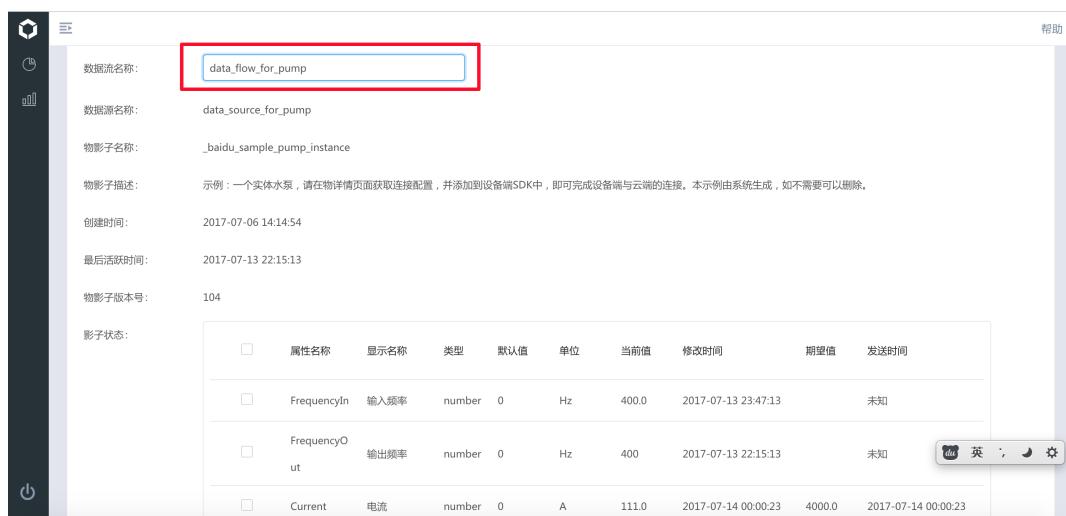
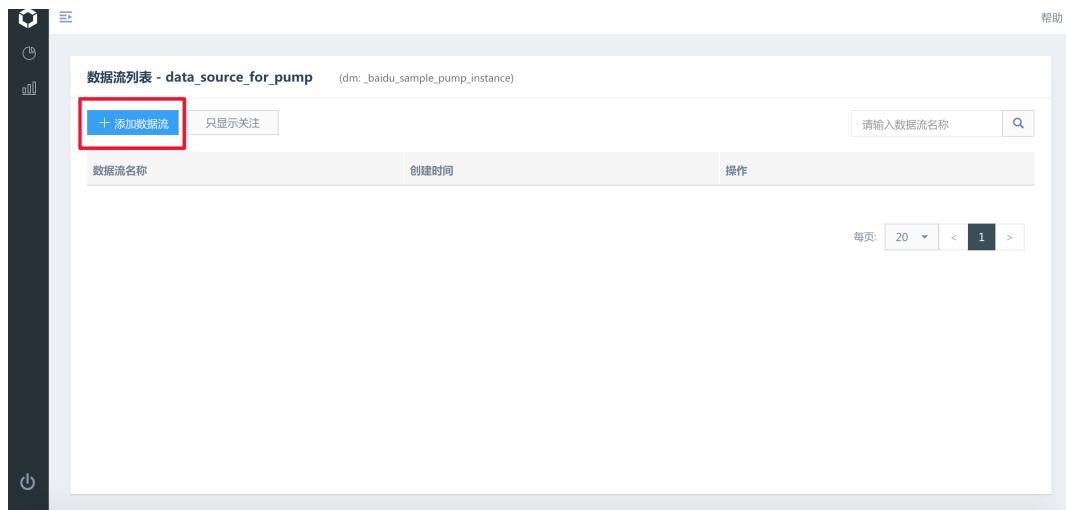
在创建数据流之前，应先完成创建数据源。

- 点击已经创建的“数据源名称”，进入该数据源下的数据流列表页面

数据源名称	类型	地址	操作
城市天气(示例)	tsdb	iotaizinternaldb.tsdb.iot.gz.baidubce.com	
data_source_for_pump	dm		

- 点击“添加数据流”，完成数据流信息配置，勾选对应的属性，点击确定。具体配置内容包括：

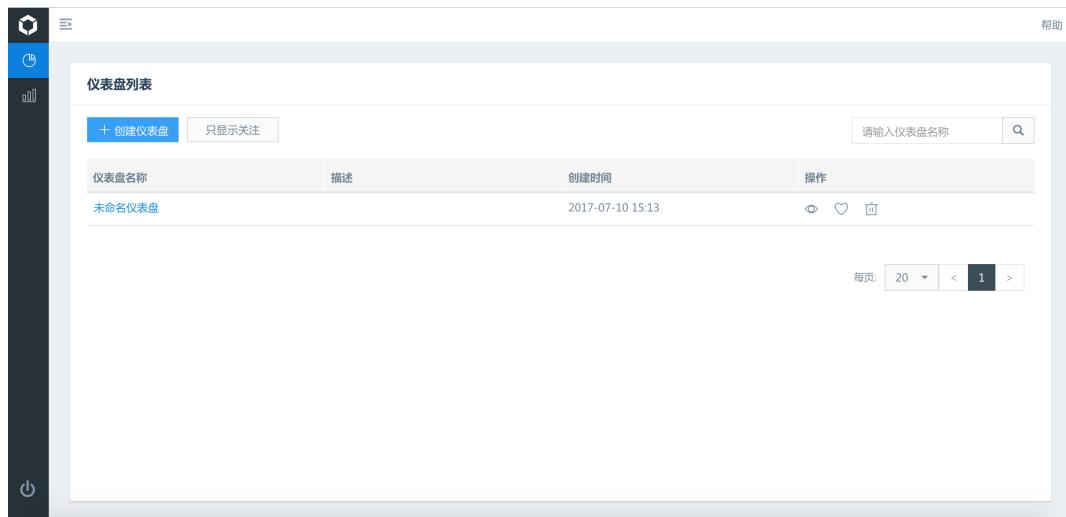
- **名称：**指定数据流名称，支持中英文字符、数字和下划线，支持2-32个字符。
- **数据源：**无需配置。
- **度量：**选择指定度量值的数据，度量值即数据的类别，如发动机的温度、转速等。
- **标签：**通过标签对数据机型过滤。
- **数据处理：**可以对一段时间的数据点做聚合，如每10分钟的和值、平均值、最大值、最小值等。



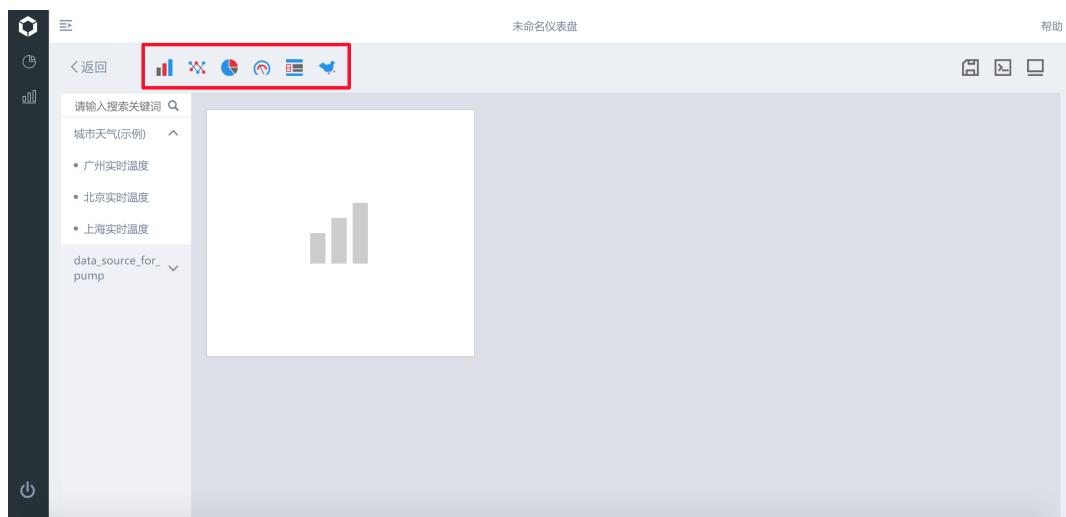
创建仪表盘

在创建仪表盘之前，应先完成创建数据源操作。

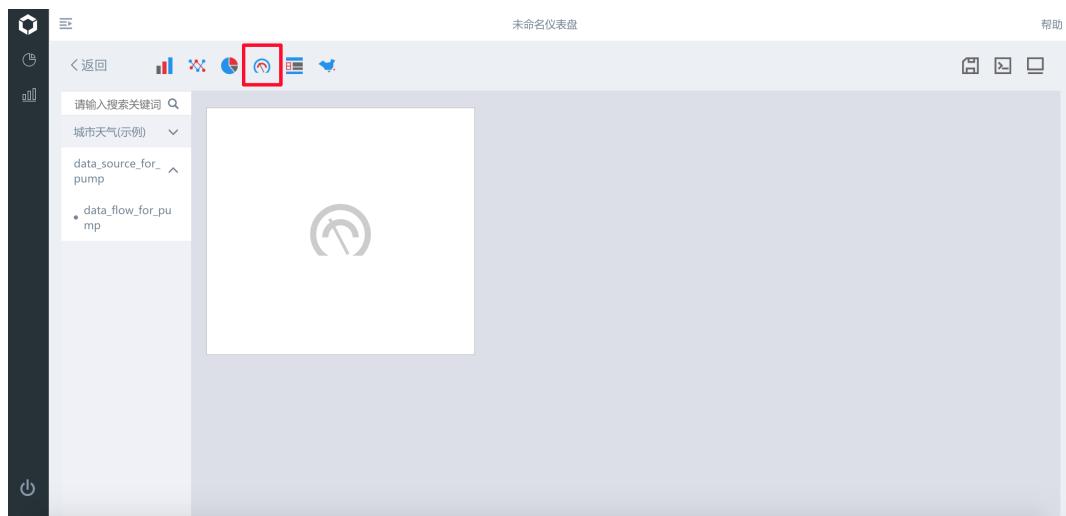
1. 在控制台中，选择“产品服务>物可视 IoT Visualization>可视化设计器>仪表盘”，进入仪表盘列表，点击“创建仪表盘”。



2. 从上方导航栏中选择图标类型。物可视当前支持的图标类型包括：柱状图、折线图、饼状图、仪表盘、指标图。

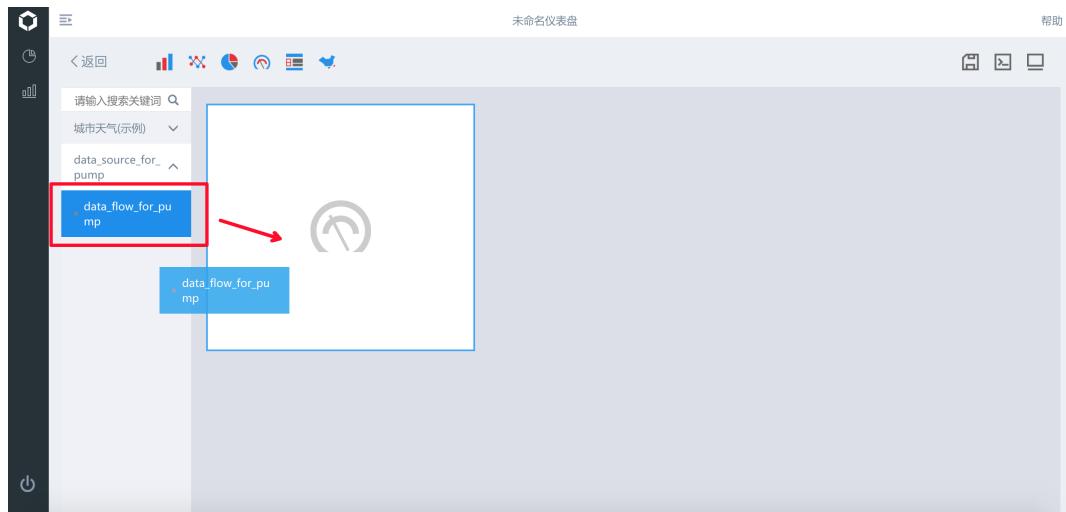


3. 点击“仪表盘”图标，生成一个空仪表盘。

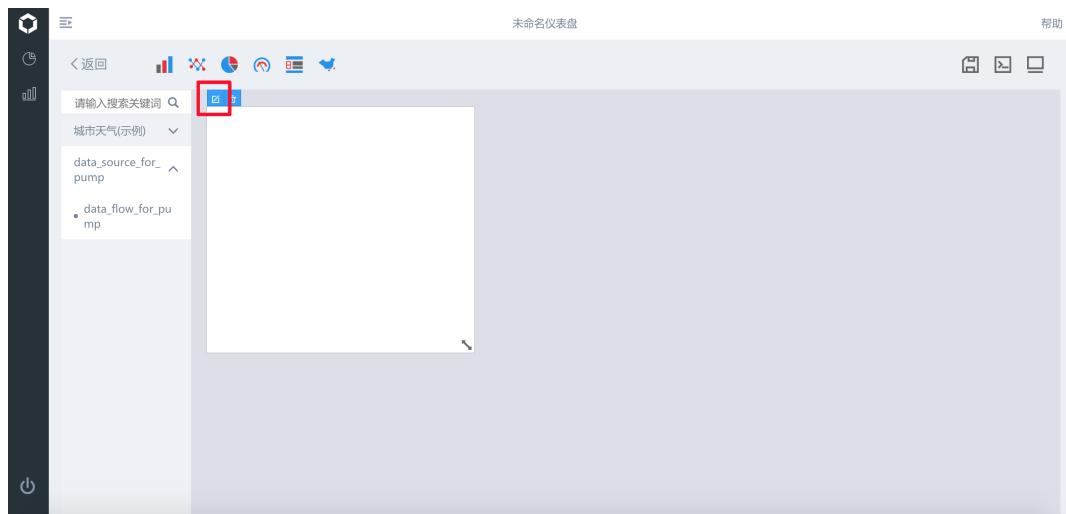


4. 从左侧列表中找到已经添加的数据源，并在该数据源中找到对应的数据流。以上一步中为示例创建的data\flow\for_pump为例，用鼠标将该数据流拖拽到空图表上。此时该空图

表将自动显示实时数据信息。



5. 生成图表后，可点击该图表左上方图标，修改相关配置信息。



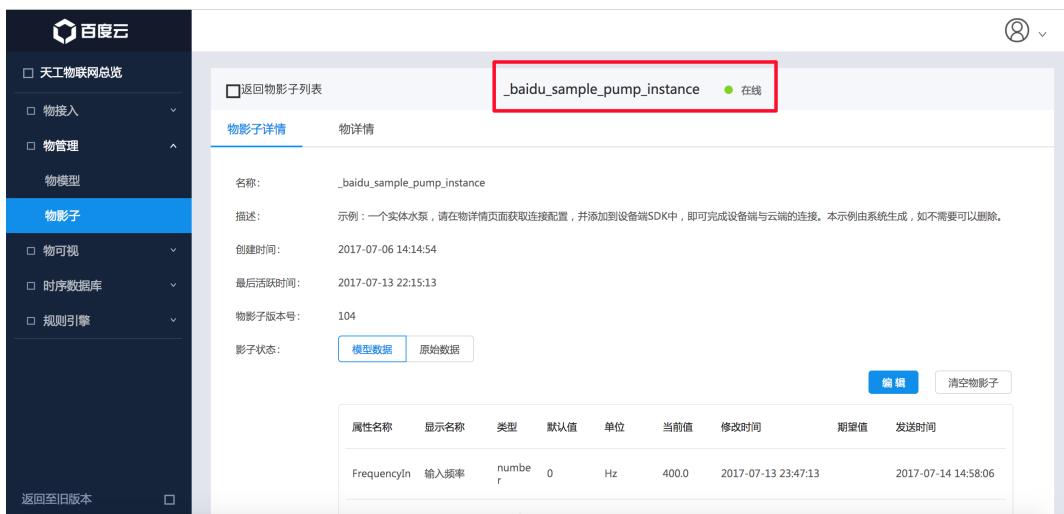
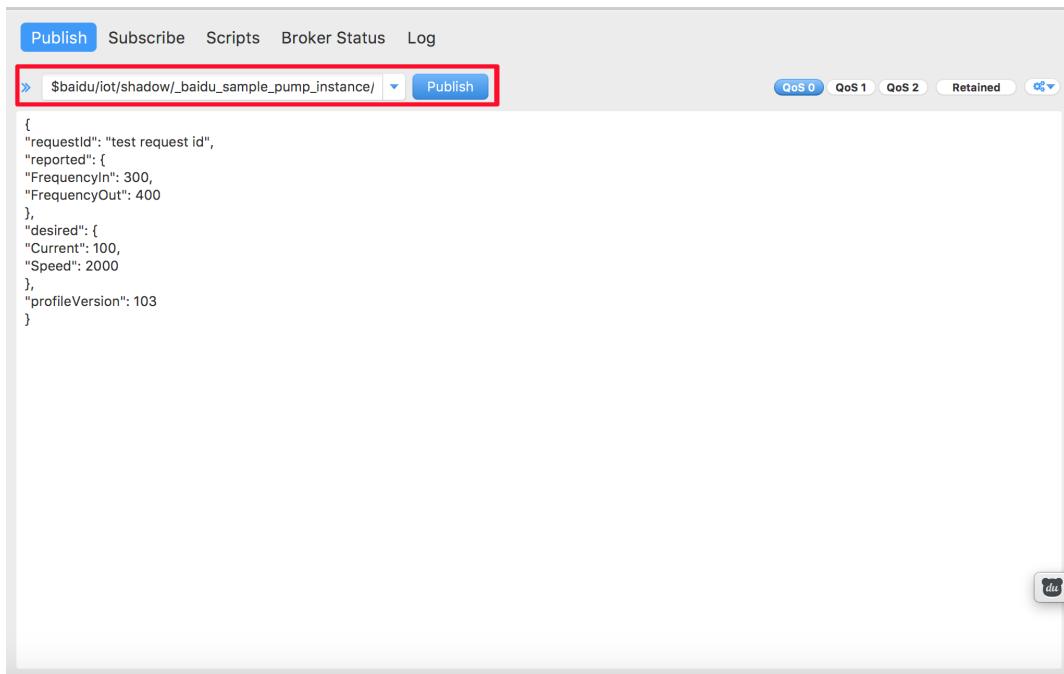
设备在线状态的说明

1. 在物管理中，我们可以看到物影子设备初始默认处于离线状态

The screenshot shows the Baidu Cloud Device Management interface. On the left, there's a sidebar with navigation items like 'IoT Device Management', 'Device Model', and 'Device Shadow'. The 'Device Shadow' item is currently selected and highlighted in blue. The main area is titled 'Device Shadow Card List' and contains a search bar and a checkbox for 'Only show followed'. There are two cards displayed: one for 'Shadow_test_1' (status: Offline) and one for 'f5real' (status: Offline). A red box highlights the status of the '_baidu_sample_pu...' card, which also shows its properties: 2 free memory, light: green, and a frequency of 400. A link to 'View more' is also present.

2. 在设备端（此处用MQTT.fx模拟设备端）用与同物影子名称相同的Client ID连接，且用该Client ID成功向\$baud/iot/shadow/myDeviceName/update发布第一条消息后，控制台物影子才能转为在线状态。本例中即物影子名称和Client ID同为\baud\sample\pump\instance，且成功向\$baud/iot/shadow/\baud\sample\pump\instancee/update发布第一条消息。

The screenshot shows the MQTT.fx 'Edit Connection Profiles' window. On the left, there's a list of connection profiles: 'IoT Device Test', 'M2M Eclipse', 'Shadow_test_1', 'Test Baidu Tiangong', 'TestProfileFor_baidu_sample_pump_instance' (which is selected and highlighted in blue), 'czwSandbox', and 'zwei_test'. The main area is titled 'Connection Profile' and shows fields for 'Profile Name' (set to 'TestProfileFor_baidu_sample_pump_instance'), 'Broker Address' (set to '4bf84731a8254b508cc8751660235f2c.m'), 'Broker Port' (set to '1884'), and 'Client ID' (set to '_baidu_sample_pump_instance'). A red box highlights the 'Client ID' field. Below these fields are tabs for 'General', 'User Credentials', 'SSL/TLS', 'Proxy', and 'Last Will and Testament'. Under the 'General' tab, there are settings for 'Connection Timeout' (30), 'Keep Alive Interval' (60), 'Clean Session' (checked), 'MQTT Version' (set to '3.1.1'), and buttons for 'Clear Publish History' and 'Clear Subscription History'. At the bottom right are 'Cancel', 'OK', and 'Apply' buttons, and a small 'du' icon.



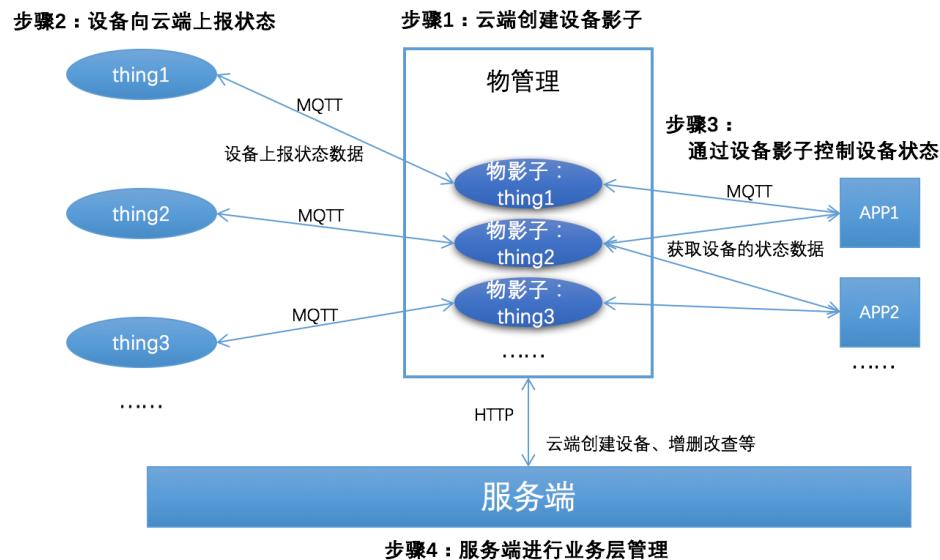
3. 若设备端使用非控制台内物影子名称的Client ID，则该设备端与控制台对应物影子的在线状态暂时无法保持一致。

3.2 基于物管理快速搭建一个应用

本文档介绍了基于物管理快速搭建一个应用的方式，帮助用户理解和使用物管理。阅读本示例前请先熟悉操作示例[端到端配置](#)。

下图为本示例解决方案概述：

基于物管理快速搭建一个应用



- 步骤1：用户在云端创建设备影子thing1, 2, 3……云端分配设备接入的地址和用户名密码。
- 步骤2：设备端的实体设备1, 2, 3……通过相应的接入地址和用户名密码连接云端，向云端上报状态。
- 步骤3：app通过接入相对应的用户名密码获取物影子信息并进行操作。
- 步骤4：用户搭建服务端，通过HTTP协议的方式，进行业务层相关操作，可以在云端创建设备，进行增删查改的操作，如哪一个APP有权限连接哪个实体设备。

操作步骤

本示例分为以下几个步骤：

1. 云端创建设备影子
2. 向云端上报状态
3. APP端通过设备影子控制设备状态
4. 服务端进行业务层管理

3.2.1 云端创建设备影子

在物管理中创建物模型 登录百度云官网，点击右上角的“管理控制台”，在控制台中，选择“产品服务>物管理 IoT Device>物模型”，在物管理中创建物模型thing_model。

物模型卡片列表

请输入模型名称

	baidu_sample_pump 说明：示例：水泵的模板。本示例由系统生成，如不需要可以删除。
	Model_test 说明：
	floor5 说明：楼层环境监控
	thing_model 说明：示例物模型

返回物模型卡片列表

thing_model

属性:	属性名称	显示名称	类型	默认值	单位
	length	长度	number	0	cm
	width	宽度	number	0	cm
	height	高度	number	0	cm

在物管理中创建物影子 选择“产品服务>物管理 IoT Device>物影子”，在物管理中创建物影子thing1, thing2, thing3……

物影子卡片列表

按名称查询

	Shadow_test_1 ● 离线 说明： 属性总计：2个 memoryFree : 3... light : green
	_baidu_sample_... ● 离线 说明：示例：一个实体水泵，请在物详情页面... 属性总计：11个 FrequencyIn : 4... Current : 111 Speed : 1033 查看更多
	f5real ● 离线 说明：环境监测5 属性总计：0个
	thing1 ● 离线 说明：设备影子1 属性总计：0个
	thing2 ● 离线 说明：设备影子2 属性总计：0个
	thing3 ● 离线 说明：设备影子3 属性总计：0个

属性名称	显示名称	类型	默认值	单位	当前值	修改时间	期望值	发送时间
length	长度	number	0	cm	2017-07-16 16:12:22	2017-07-16 16:12:22		
width	宽度	number	0	cm	2017-07-16 16:12:22	2017-07-16 16:12:22		
height	高度	number	0	cm	2017-07-16 16:12:22	2017-07-16 16:12:22		

TCP Address :
tcp://4bf84731a8254b508cc8751660235f2c.mqtt.iot.gz.baidubce.com:1883
SSL Address :
ssl://4bf84731a8254b508cc8751660235f2c.mqtt.iot.gz.baidubce.com:1884
name : 4bf84731a8254b508cc8751660235f2c/thing
key : 1Q0ntkFG0Vkk3zeQnP72JkfhiYJIIH0WnFAyY=

- 记录好设备影子的连接配置信息。
- 注意：用户除了在控制台物管理中创建物模型、物影子之外，还可以通过API接口/SDK调用的方式创建。[如何创建？](#)

3.2.2 向云端上报状态

- 将设备影子的连接配置信息配置到SDK中，实现对应的实体设备与云端连接。[如何连接？](#)
 - 使用Baidu IoT Edge SDK
 - 使用MQTT客户端SDK
- 设备端通过接入云端对应的设备影子的地址、用户名、密钥来连接云端，更新所对应的物影子状态。

3.2.3 APP端通过设备影子控制设备状态

- 用户在自己所写的程序中，可以通过接入相对应设备影子的用户名、密钥来获取物影子信息。
- 物影子与设备一一对应，但APP端可以通过配置多个物影子的连接配置信息达到一对多的关系。

3.2.4 服务端进行业务层管理

- 服务端进行业务层的管理，在云端创建设备，对设备进行增删改查等操作，比如哪一个APP有权限连接哪个实体设备。

第4章

操作指南

4.1 介绍

说明

推荐用户使用新版物管理操作平台，新版操作平台为天工物联网的统一控制台入口，用户可通过该平台无缝对接其它天工服务，便于物管理服务与其它服务集成和对接。有关旧版（即物管理V1.0）操作指导，请参看[物管理操作指南 - V1.0](#)。

为了给您提供更高效的一站式物联网开发服务，物管理升级全新版本。在新版本的物管理中，您需要了解两个概念：物模型和物影子。

- 物模型由一个或多个属性构成，您可以用来表示一类设备。
- 物影子是物理世界的物在云端的『影子』或『数字双胞胎』，这个物的属性可以从物模型中继承过来。运行时，物将监控值上报给物影子，物影子会给这个物做一个状态暂存，天工其他产品或您的应用程序可以直接使用；与此同时，您也可以通过操作物影子来控制物。

物影子将作为天工产品使用的核 心，天工产品都将会读取物影子的数据（陆续对接上线）。另外，天工物联网会提供一个统一的控制台入口，让您更便捷搭建物联网的应用。

4.2 在云端创建物影子

4.2.1 物模型

创建物模型 通过物模型可以为设备定义一套属性模板，在[创建物影子](#)时可以引用该模板，实现业务的快速部署。具体操作方法如下：

1. 登录百度云官网，点击右上角的“管理控制台”，快速进入控制台界面。

2. 选择“产品服务>物管理 IoT Device”，进入“天工物联网”控制台。
3. 选择“物管理>物模型”，进入物模型卡片列表；点击“新建物模型”进入物模型配置界面。

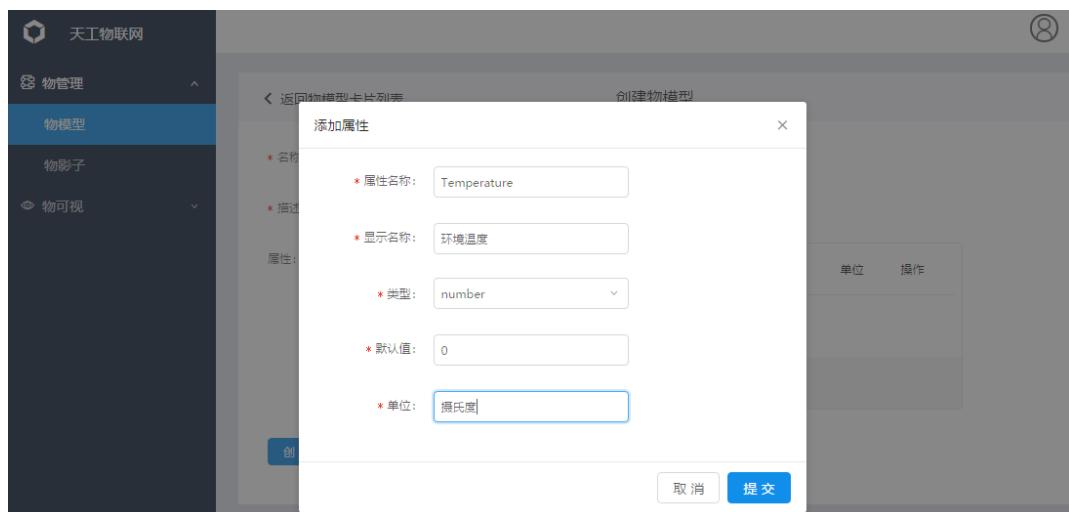


4. 填写物模型配置，包括名称、描述和属性。

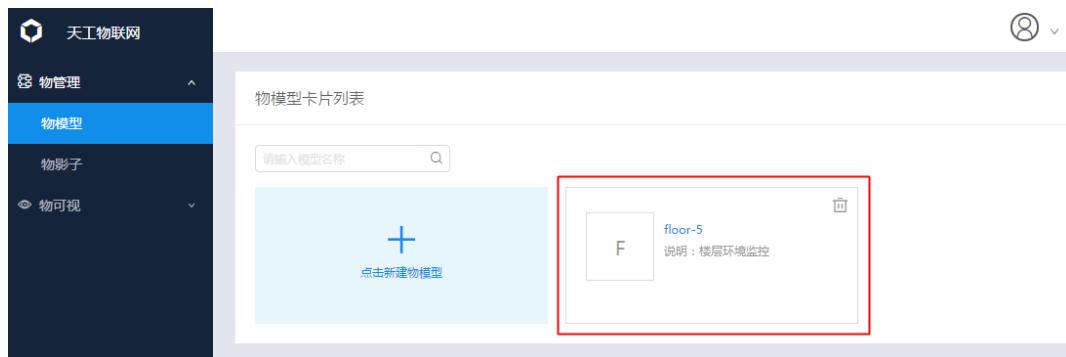
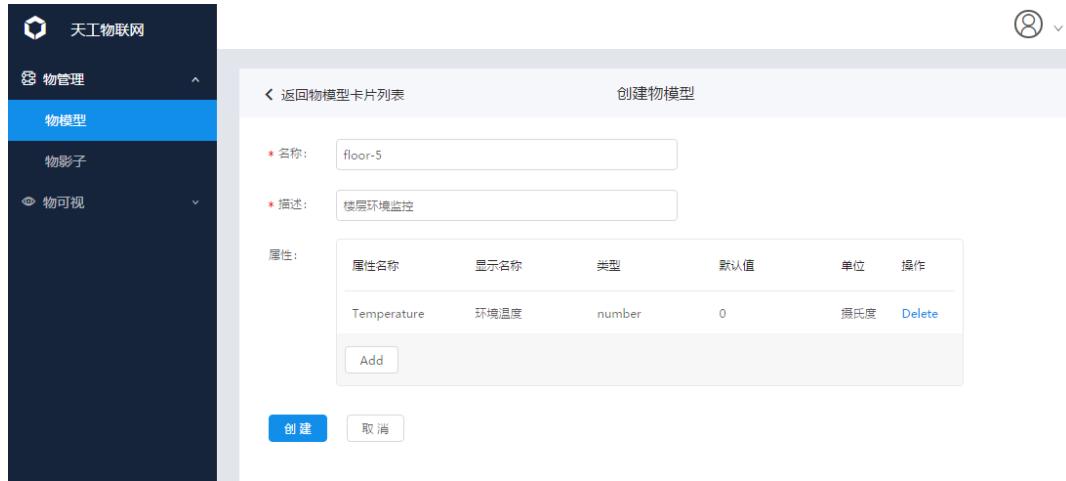
创建成功后物模型名称和属性名称无法修改，同一物模型下属性名称必须唯一。



点击“Add”为物模型新增一条属性。



完成属性配置后，点击“创建”，完成物模型创建。

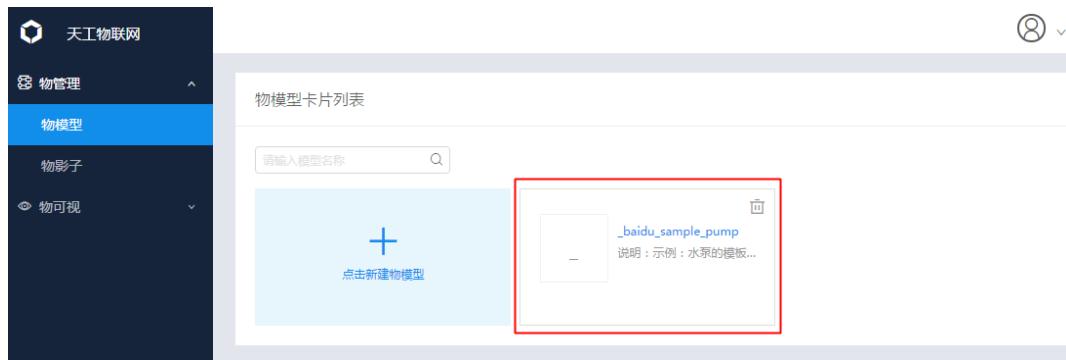


管理物模型 创建物模型以后，可以对以下信息进行编辑修改，包括：

- 物模型描述信息
- 删除或新增属性
- 属性描述、默认值、单位

具体修改方法如下：

1. 选择“产品服务>物管理 IoT Device”，进入“天工物联网”控制台。
2. 选择“物管理>物模型”，进入物模型卡片列表。
3. 选择需要修改的物管理模型卡片，以物模型示例为例，点击卡片进入详情页面。



4. 点击“编辑”进入编辑模式，此时可对物模型进行属性编辑等操作。完成操作后点击“保存”使配置生效。



4.2.2 物影子

创建物影子 在创建物影子之前，必须先[创建物模型](#)。

1. 选择“产品服务>物管理 IoT Device”，进入“天工物联网”控制台。
2. 选择“物管理>物影子”，进入物影子卡片列表；点击“新建物影子”进入物模型配置界面。



3. 填写物模型配置，包括名称、描述和物模型；点击“创建”完成物影子创建。



[查看影子详情](#) [查看设备当前状态](#)

在影子详情界面，用户可以看到以下信息

- 模型数据，通过图表展示物模型定义的所有属性，如下图所示：

物影子的列表页展示模型数据，在关联模型中存在的字段才展示在该页。如果用户 reported了不在该物影子中的属性，通过“原始数据”查看。

属性名称	显示名称	类型	默认值	单位	当前值	修改时间	期望值	发送时间
FrequencyIn	输入频率	number	0	Hz	2017-06-16 11:1	2017-06-16 11:1		6:23
FrequencyOut	输出频率	number	0	Hz	2017-06-16 11:1	2017-06-16 11:1		6:23
Current	电流	number	0	A	2017-06-16 11:1	2017-06-16 11:1		6:23
Power	功率	number	—	—	2017-06-16 11:1	2017-06-16 11:1		—

通过查看“当前值”字段，可以获取设备各属性的实时数据信息。相关字段解释如下：

- 当前值，指设备最后一次上报的该属性的值。如果没有上报则为空。
- 修改时间，指设备最后一次更新该属性的值。
- 期望值，指通过控制台或者应用程序性修改的desired值。如果没有。期望值则为空。
- 发送时间，指该属性最后一次有desired值修改的时间

- 原始数据，即设备影子的原始数据，用户可以通过原始数据查看设备状态或远程控制设备，如下图所示：



设备影子介绍

用户可通过设备影子查看实体设备的属性和状态等信息，如设备id、设备名称等。

设备影子是一个json文档，如下例所示：

```
"device": {  
    "reported": {  
        "firewareVersion": "1.0.0",  
        "light": "green"  
    }  
    "desired": {  
        "light": "red"  
    }  
    "profileVersion": 10,  
    "timestampe  
}
```

“device” 中的属性需要同步到设备实体。其中，“reported” 表示设备端通过MQTT连接汇报到物管理的设备状态。“desired” 表示控制端希望控制设备变换到的目标状态。

远程控制属性

1. 点击“编辑物影子”，编辑属性的期望值。

2. 点击“提交”后，物管理将期望值下发至设备端。

属性名称	显示名称	类型	默认值	单位	当前值	修改时间	期望值	发送时间
FrequencyIn	输入频率	num	0	Hz	<input type="text"/>	2017-06-16 11:30:53	<input type="text" value="100"/>	2017-06-16 11:30:53
FrequencyOut	输出频率	num	0	Hz	<input type="text"/>	2017-06-16 11:30:53	<input type="text"/>	2017-06-16 11:30:53
Current	电流	num	0	A	<input type="text"/>	2017-06-16 11:30:53	<input type="text"/>	2017-06-16 11:30:53
Speed	速度	num	0	rpm	<input type="text"/>	2017-06-16 11:30:53	<input type="text"/>	2017-06-16 11:30:53

用户也可以通过编辑原始数据desired中的属性实现设备的远程控制。

获取连接配置 创建物影子后，系统将自动生成与该物影子对应的连接配置，包括endpoint实例地址、设备名称和密钥信息。该信息将可用于将实体设备连接至物管理。具体操作如下：

1. 进入物详情界面。

2. 点击“连接配置”。

注意

点击“更新密钥”后，原有密钥将失效，会导致已经接入的设备断开连接。

① 更新密钥后会导致原有的连接断开，需要用新的密钥连接，请谨慎操作！

TCP Address : tcp://d3fa21a0.../468239681ffa.10.73.203.34:8061
 SSL Address : ssl://d3fa21a01...168239681ffa.10.73.203.34:61614
 name : d3fa21a01d6446f3a96
 key : ggAbHH1eHO0qqgdatx...TmJQJMrZN86JM=

4.3 将实体设备接入云端

4.3.1 使用Baidu IoT Edge SDK

说明

推荐用户优先使用百度云提供的SDK。

百度云提供的SDK封装了MQTT客户端SDK，屏蔽了MQTT客户端SDK的细节和topic信息，可以帮助用户实现业务的快速部署。

用户可以在设备端安装百度云提供的SDK，并配置[连接信息](#)，实现设备与百度云物管理的快速对接。

有关IoT Edge SDK的下载和安装，请查看：<https://github.com/baidu/iot-edge-sdk-for-iot-parser/releases/tag/v1.0.1>

4.3.2 使用MQTT客户端SDK

如果设备端已经调用了[Paho \(即MQTT Client SDK\)](#)，可以通过与特定的topic通信实现与百度云的对接。在物管理中定义了系统topic用于物管理服务和设备端基于物接入服务以及MQTT协议进行通信。

更新设备状态到设备影子 将信息推送到主题' \$baidu/iot/shadow/{deviceName}/update'，可实现将设备状态更新到设备影子。

示例：

```
pub \$baidu/iot/shadow/myDeviceName/update
{
    "requestId": "{requestId}",
    "reported": {
        "memoryFree": "32MB",
        "light": "green"
    },
    "desired": {
        "rotate": 100
    },
    "profileVersion": 5
}
```

- “requestId” 为请求的唯一标识符，每一个请求的requestId是唯一的，可随机生成。
- “reported” 为可选字段，表示设备影子的“reported” 中的相关属性需要更新。

- “desired” 为可选字段，表示设备影子的“desired” 中的相关属性需要更新。
- “profileVersion” 为可选字段，当未指定profileVersion时，物管理接收设备影子更新请求后，会将profileVersion自动加1；若指定profileVersion，物管理会检查请求中的profileVersion是否大于当前的profileVersion。只有在大于的情况下，物管理才会接受设备端的请求，更新设备影子，并将profileVersion更新到相应的版本。

更新设备影子适用于两种应用场景：

1. 设备同步状态到物管理服务。设备在状态发生变化时，将实时的状态同步到物管理服务，包括状态的自动变化以及设备反控后状态的变化。更新设备状态，通常更新“reported” 字段中的相关属性。对于反控后更新状态，设备可以用实时状态同时更新该属性的“reported” 和“desired” 中的值。
2. 通过MQTT协议反控设备状态。如果需要通过MQTT协议反控设备属性，可以通过更新“desired” 字段实现。当物管理接收到“desired” 相关属性的更新后，会diff设备影子中“reported” 和“desired” 相关字段，将diff后的结果发送到delta主题。设备端通过订阅delta主题，可将设备状态同步到“desired”的状态。状态反控后，更新设备影子，使“reported” 和“desired”的值一致。物管理对设备的反控请参考通过设备影子控制设备状态。

订阅主题获取设备影子更新成功后的结果：

```
sub \$baidu/iot/shadow/myDeviceName/update/accepted
{
    "requestId": "{requestId}",
    "reported": {
        "firewareVersion": "1.0.0",
        "light": "green"
    },
    "desired": {
        "light": "red"
    },
    "lastUpdatedTime": {
        "reported": {
            "firewareVersion": 1494904250,
            "light": 1494904250
        },
        "desired": {
            "light": 1494904250
        }
    },
    "profileVersion": 10
}
```

订阅主题获取设备影子更新失败后的结果：

```
sub \$baidu/iot/shadow/myDeviceName/update/rejected
{
    "requestId": "{requestId}",
    "code": "{errorCode}",
    "message": "{errorMessage}"
}
```

从设备影子获取设备状态，发送请求到主题' \$baidu/iot/shadow/{deviceName}/get'，可以获取该设备在设备影子中的所有状态信息。

示例：

```
pub \$baidu/iot/shadow/myDeviceName/get
{
    "requestId": "{requestId}"
}
```

订阅主题获取设备影子：

```
sub \$baidu/iot/shadow/myDeviceName/get/accepted
{
    "requestId": "{requestId}",
    "reported": {
        "firewareVersion": "1.0.0",
        "light": "green"
    },
    "desired": {
        "light": "red"
    },
    "lastUpdatedTime": {
        "reported": {
            "firewareVersion": 1494904250,
            "light": 1494904250
        },
        "desired": {
            "light": 1494904250
        }
    },
    "profileVersion": 10
}
```

同时，可以订阅获取设备影子失败的相关消息：

```
sub \$baidu/iot/shadow/myDeviceName/get/rejected
{
    "requestId": "{requestId}",
    "code": "{errorCode}",
    "message": "{errorMessage}"
}
```

通过设备影子控制设备状态 控制端可以通过MQTT协议更新设备影子中的‘desired’字段，达到反控设备的目的。物管理在接受到‘desired’字段更新后，会比较‘reported’和‘desired’之间的差异，并将diff结果发送到主题‘\$baidu/iot/shadow/{deviceName}/delta’。

例如，当前‘reported’中的‘light’字段为green，控制端将‘desired’中的‘light’字段更新为‘red’，此时物管理会通过delta主题反控设备：

```
sub \$baidu/iot/shadow/myDeviceName/delta
{
    "requestId": "{requestId}",
    "desired": {
        "light": "red"
    }
}
```

若设备更新状态失败，可将相关错误信息发送到物管理：

```
pub \$baidu/iot/shadow/myDeviceName/delta/rejected
{
    "requestId": "{requestId}",
    "code": "{errorCode}",
    "message": "{errorMessage}"
}
```

删除设备影子 支持通过MQTT主题‘\$baidu/iot/shadow/{deviceName}/delete’删除设备影子。

示例：

```
pub \$baidu/iot/shadow/myDeviceName/delete
```

```
{
  "requestId": "{requestId}"
}
```

通过订阅 ‘\$baidu/iot/shadow/{deviceName}/delete/accepted’ 可以获取设备影子删除成功后的response。

```
sub \$baidu/iot/shadow/myDeviceName/delete/accepted
{
  "requestId": "{requestId}",
  "reported": {
    "firewareVersion": "1.0.0",
    "light": "green"
  },
  "desired": {
    "light": "red"
  },
  "lastUpdatedTime": {
    "reported": {
      "firewareVersion": 1494904250,
      "light": 1494904250
    },
    "desired": {
      "light": 1494904250
    }
  },
  "profileVersion": 10
}
```

主题’ \$baidu/iot/shadow/{deviceName}/delete/rejected’ 推送删除设备影子失败后的相关信息

```
pub \$baidu/iot/shadow/myDeviceName/delete/rejected
{
  "requestId": "{requestId}",
  "code": "{errorCode}",
  "message": "{errorMessage}"
}
```

订阅设备影子的变化 可以通过主题 “\$baidu/iot/shadow/{deviceName}/update/documents” 订阅设备影子中reported字段内容的变化。物管理在收到设备影子的update请求、并成功更新后，如果reported字段内容有变（变化条件包括：增加属性、减少属性、属性

值有变化），会把reported字段中更新属性的当前值和更新前的值发送到“documents”主题。

示例：

```
sub \$baidu/iot/shadow/{deviceName}/update/documents
{
    "requestId": "{requestId}",
    "profileVersion": 10,
    "current": {
        "light": "green"
    },
    "previous": {
        "light": "red"
    }
}
```

documents topic只反映了reported字段内容中发生变化的属性状态，如在reported字段内容变化时，需要订阅设备的全量的属性状态，可以通过主题“\$baidu/iot/shadow/{deviceName}/update/snapshot”获取。该主题内容会包括shadow的reported字段的全部属性，此外还包含相应的lastUpdatedTime、profileVersion字段内容。

示例：

```
sub \$baidu/iot/shadow/{deviceName}/update/snapshot
{
    "requestId": "{requestId}",
    "profileVersion": 10,
    "reported": {
        "light": "green"
    },
    "lastUpdatedTime": {
        "reported": {
            "light": 1494904250
        }
    }
}
```

Device Profile Device Profile由Device Registry和Device Shadow两部分组成。

```
{
    "name": "test", //设备名称
```

```
"id": "098f6bcd4621d373cade4e832627b4f6",      //设备ID
"description": "测试设备",                      //设备描述
"state": "online",                            //设备状态, online/offline
"templateId": "123456",                        //设备模板ID
"templatedName": "TestTemplate",               //设备模板名称
"createTime": 1494904250,                      //创建时间
"lastActiveTime": 149490300,                   //最后一次设备影子(reported)更新时间
"attributes": {
    "region": "Shanghai"                     //设备Tag
},
"device": {                                     //设备影子
    "reported": {
        "firewareVersion": "1.0.0",
        "light": "green"
    },
    "desired": {
        "light": "red"
    },
    "lastUpdatedTime": {
        "reported": {
            "firewareVersion": 1494904250,
            "light": 1494904250
        },
        "desired": {
            "light": 1494904250
        }
    },
    "profileVersion": 10
}
}
```

4.4 物管理操作指南 - V1.0

说明

推荐用户使用新版物管理操作平台，新版操作平台为天工物联网的统一控制台入口，用户可通过该平台无缝对接其它天工服务，便于物管理服务与其它服务集成和对接。

有关旧版（即物管理V1.0）操作指导，请参看[物管理操作指南 - V1.0](#)。

第5章

API参考 - V3

5.1 目录

- 介绍
- 设备列表管理
 - 创建单个设备
 - 删除设备
 - 获取设备Profile
 - 获取设备View
 - 获取及查询影子列表
 - 获取设备接入详情
 - 更新密钥
 - 更新设备属性
 - 更新单个设备View信息
 - 更新单个设备注册表信息
 - 重置设备影子
- 设备模板管理
 - 创建模板
 - 删除模板
 - 获取模板列表
 - 获取模板
 - 更新模板
- 参数定义
 - DeviceList参数列表
 - DeviceProfile参数列表
 - DeviceBasicInfo参数列表
 - DeviceAttributes参数列表

- [DeviceView参数列表](#)
- [DeviceViewAttributes参数列表](#)
- [DeviceAccessDetail参数列表](#)
- [SchemaProperty参数列表](#)
- [Schema参数列表](#)

5.2 介绍

5.2.1 简介

百度开放云IoT物管理套件，提供用户在云端管理设备的能力。用户能够获取并控制设备状态、进行设备的批量操作以及设备诊断。一方面，设备主动向物管理中心更新状态信息；另一方面，控制端也可以通过和设备管理中心交互反控设备的行为，比如设备状态更新、OTA等。因此，设备管理中心既需要负责和设备端交互，又要负责和控制端交互。设备端的交互主要基于MQTT协议，而控制端通过HTTP通信。IoT Device Management API主要包括控制端的相关功能，以Restful API的形式提供。

V3版本引入模板的概念，用户可以通过模板和视图定义比较关注的设备属性，并将相关数据点写入百度天工时序数据库。此外，在V3版本中，我们分拆并丰富了设备主题，以帮助用户更好的处理设备数据上传以及反控等信息。

5.3 设备列表管理

5.3.1 创建单个设备

方法	API	说明
POST	/v3/iot/management/device	创建单个设备

** 请求参数 **

参数名称	参数类型	是否必须	说明
deviceName	string	必选	设备名称
description	string	必选	设备名称
schemaId	string	必选	设备名称

返回参数

一个[DeviceAccessDetail](#)对象

请求示例

```
POST /v3/iot/management/device HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
    "deviceName": "mydevice",
    "description": "device_description",
    "schemaId": "uuid"
}
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
    "tcpEndpoint": "tcp://test.baidu.iot.com",
    "sslEndpoint": "ssl://test.baidu.iot.com",
    "username": "endpointName/device_1",
    "key": "bWwCxGwaw3boV48NqsuG+XVaHpxfKdMPvmdJzNObvbY="
}
```

5.3.2 删除设备

方法	API	说明
PUT	/v3/iot/management/device?remove	删除设备

请求参数

一个[DeviceList](#)对象

返回参数

一个[DeviceList](#)对象

请求示例

```
PUT /v3/iot/management/device?remove HTTP/1.1
```

```
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8

{
    "devices": [
        "device-1",
        "device-2"
    ]
}
```

[返回示例](#)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
    "devices": [
        "device-1",
        "device-2"
    ]
}
```

5.3.3 获取设备Profile

方法	API	说明
GET	/v3/iot/management/device/{deviceName}	获取设备Profile

[返回参数](#)

一个<https://cloud.baidu.com/doc/IOTDM/DeviceProfile>对象

[请求示例](#)

```
GET /v3/iot/management/device/mydevice HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
```

[返回示例](#)

```

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
    "id": "0e357fac55b64bfa8c5c7e9fd56ee320",
    "name": "mydevice",
    "description": "my device in Shanghai",
    "createTime": 1494904250,
    "state": "online",
    "lastActiveTime": 1494904250,
    "schemaId": "uuid",
    "schemaName": "baidu_shanghai",
    "favourite": false,
    "attributes": {
        "region": "Shanghai"
    },
    "device": {
        "reported": {
            "firewareVersion": "1.0.0",
            "light": "green"
        },
        "desired": {
            "light": "red"
        },
        "lastUpdatedTime": {
            "reported": {
                "firewareVersion": 1494904250,
                "light": 1494904250
            },
            "desired": {
                "light": 1494904250
            }
        },
        "profileVersion": 10
    }
}

```

5.3.4 获取设备View

方法	API	说明
GET	/v3/iot/management/deviceView/{deviceName}	获取设备Profile和模型合并后的View

范围参数

一个DeviceView对象

请求示例

```
GET /v3/iot/management/deviceView/mydevice HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
  "id": "0e357fac55b64bfa8c5c7e9fd56ee320",
  "name": "mydevice",
  "description": "my device in Shanghai",
  "createTime": 1326789000,
  "state": "online",
  "lastActiveTime": 1326789000,
  "schemaId": "uuid",
  "schemaName": "baidu_shanghai",
  "favourite": false,
  "profileVersion": 10,
  "properties": [
    {
      "attributeName": "light",
      "showName": "灯光",
      "type": "string",
      "defaultValue": "red",
      "reportedValue": "green",
      "desiredValue": "green"
      "unit": "-",
      "reportedTime": 1326789000,
      "desiredTime": 1435860000
    }
  ]
}
```

5.3.5 获取及查询影子列表

方法	API	说明
GET	/v3/iot/management/device? pageNo=xx&pageSize=xx&orderBy =xx&&order=xx&name=xx&value=xx&favourite=xx	查询设备Profile

请求参数

参数名称	参数类型	是否必须	说明
pageNo	int	可选	表示取第几页， 默认1
pageSize	int	可选	每页返回的数目， 默认10
orderBy	String	可选	name, createTime, lastActiveTime, 默认name
order	String	可选	按升序或降序返回结果, desc / asc, 默认asc
name	String	可选	查询属性名, 默认全量
value	String	可选	查询属性名对应的值, 默认全量
favourite	String	可选	收藏, true / false / all, 默认all

返回参数

参数名称	参数类型	说明
devices	List	由DeviceProfile对象组成的列表
amount	int	总数目
pageNo	int	返回页数
pageSize	int	每页返回数目

请求示例

```
PUT /v3/iot/management/device?pageNo=1&pageSize=10&orderBy=createTime&&order=desc&name=schemaName
1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
```

[返回示例](#)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
    "amount": 20,
    "pageNo": 1,
    "pageSize": 10,
    "devices": [
        {
            "id": "0e357fac55b64bfa8c5c7e9fd56ee320",
            "name": "mydevice",
            "description": "my description",
            "state": "online",
            "createTime": 1326789000,
            "lastActiveTime": 1326789000,
            "schemaId": "uuid",
            "schemaName": "baidu_shanghai",
            "favourite": false,
            "attributes": {
                "region": "Shanghai"
            },
            "device": {
                "reported": {
                    "light": "green"
                }
            },
            "desired": {
                "light": "red"
            }
        },
        "profileVersion": 10,
        "lastUpdatedTime": {
            "reported": {
                "light": 1326789000
            },
            "desired": {
                "light": 1326789000
            }
        }
    ]
}
```

```

        }
    }
}

...
]

}

```

5.3.6 获取设备接入详情

方法	API	说明
GET	/v3/iot/management/device/{deviceName}/accessDetail	获取设备接入详情

[返回参数](#)

一个[DeviceAccessDetail](#)实例

[请求示例](#)

```

GET /v3/iot/management/device/mydevice/accessDetail HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8

```

[返回示例](#)

```

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
  "tcpEndpoint": "tcp://test.baidu.iot.com",
  "sslEndpoint": "ssl://test.baidu.iot.com",
  "username": "endpointName/device_1",
  "key": "xxxxxxxxxx"
}

```

5.3.7 更新密钥

方法	API	说明
PUT	/v3/iot/management/device/{deviceName}?updateSecretKey	更改密钥

请求参数

参数名称	参数类型	是否必须	说明
deviceName	String	必须	设备名称

返回参数

一个[DeviceAccessDetail](#)实例

请求示例

```
PUT /v3/iot/management/device/deviceName?updateSecretKey HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
  "tcpEndpoint": "tcp://test.baidu.iot.com",
  "sslEndpoint": "ssl://test.baidu.iot.com",
  "username": "endpointName/deviceName",
  "key": "bWwCxGwaw3boV48NqsuG+XVaHpxfKdMPvmdJzN0vbY="
}
```

5.3.8 更新设备属性

方法	API	说明
PUT	/v3/iot/management/device/{deviceName}?updateProfile	修改设备属性

请求参数

参数名称	参数类型	是否必须	说明
attributes	JSONObject	可选	需要更新的 attributes
device	DeviceAttributes	可选	需要更新的 device 属性

返回参数

一个DeviceProfile对象

请求示例

```
PUT /v3/iot/management/device/myDevice?updateProfile HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
  "device": {
    "desired": {
      "light": "red"
    },
    "reported": {
      "light": "green"
    }
  },
  "attributes": {
    "region": "Shanghai"
  }
}
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
  "id": "0e357fac55b64bfa8c5c7e9fd56ee320",
  "name": "mydevice",
  "description": "my device in Shanghai",
  "createTime": 1494904250,
  "state": "online",
```

```

    "lastActiveTime": 1494904250,
    "schemaId": "uuid",
    "schemaName": "baidu_shanghai",
    "favourite": false,
    "attributes": {
        "region": "Shanghai"
    },
    "device": {
        "reported": {
            "firewareVersion": "1.0.0",
            "light": "green"
        },
        "desired": {
            "light": "red"
        },
        "lastUpdatedTime": {
            "reported": {
                "firewareVersion": 1494904250,
                "light": 1494904250
            },
            "desired": {
                "light": 1494904250
            }
        },
        "profileVersion": 10
    }
}

```

5.3.9 更新单个设备View信息

方法	API	说明
PUT	/v3/iot/management/deviceView/{deviceName}?updateView	修改设备View信息

请求参数

请求名称	参数类型	是否必须	说明
reported	JsonNode	可选	需要更新的 reported
desired	JsonNode	可选	需要更新的 desired
profileVersion	int	可选	更新版本

返回参数

一个<https://cloud.baidu.com/doc/IOTDM/DeviceView>对象

请求示例

```
PUT /v3/iot/management/deviceView/myDevice?updateView HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
  "desired": {
    "light": "red"
  },
  "reported": {
    "light": "green"
  }
}
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5dddac970054
{
  "id": "0e357fac55b64bfa8c5c7e9fd56ee320",
  "name": "myDevice",
  "description": "my device in Shanghai",
  "createTime": 1326789000,
  "state": "online",
  "lastActiveTime": 1326789000,
  "schemaId": "uuid",
  "schemaName": "baidu_shanghai",
  "favourite": false,
  "profileVersion": 1,
  "properties": [
    {
      "attributeName": "light",
      "showName": "灯光",
      "type": "String",
      "defaultValue": "red",
      "reportedValue": "green",
      "desiredValue": "red",
      "unit": "-"
    }
  ]
}
```

```

    "reportedTime": 1435860000,
    "desiredTime": 1435860000
}
]
}

```

5.3.10 更新单个设备注册表信息

方法	API	说明
PUT	/v3/iot/management/device/{deviceName}?updateRegistry	更新单个设备注册表信息

请求参数

参数名称	参数类型	是否必须	说明
description	String	可选	需要更新的设备描述
schemaId	String	可选	需要更换的模板Id
favourite	boolean	可选	是否需要收藏

返回参数

一个DeviceProfile对象

请求示例

```

PUT /v3/iot/management/device/myDevice?updateRegistry HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
  "description": "new device description",
  "schemaId": "uuid",
  "favourite": true,
}

```

返回示例

HTTP/1.1 200 OK

```

Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
    "id": "0e357fac55b64bfa8c5c7e9fd56ee320",
    "name": "mydevice",
    "description": "new device description",
    "createTime": 1494904250,
    "state": "online",
    "lastActiveTime": 1494904250,
    "schemaId": "uuid",
    "schemaName": "baidu_shanghai",
    "favourite": true,
    "attributes": {
        "region": "Shanghai"
    },
    "device": {
        "reported": {
            "firewareVersion": "1.0.0",
            "light": "green"
        },
        "desired": {
            "light": "red"
        },
        "lastUpdatedTime": {
            "reported": {
                "firewareVersion": 1494904250,
                "light": 1494904250
            },
            "desired": {
                "light": 1494904250
            }
        },
        "profileVersion": 10
    }
}

```

5.3.11 重置设备影子

方法	API	说明
PUT	/v3/iot/management/device?reset	重置（清空）设备影子

请求参数

一个[DeviceList](#)对象

[返回参数](#)

一个[DeviceList](#)对象

[请求示例](#)

```
PUT /v3/iot/management/device?reset HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
  "devices": [
    "device-1",
    "device-2"
  ]
}
```

[返回示例](#)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
  "devices": [
    "device-1",
    "device-2"
  ]
}
```

5.4 设备模板管理

在新版物管理中，我们引入了模板的概念，模板定义了一类设备的schema。Schema中定义了各设备各属性的显示名称、类型、默认值等信息。可以理解为，通过模板，我们定义了设备的视图(view)。此外，我们需要支持模板的CRUD操作。IoT Device Management Schema API 主要包含管理设备模板的相关接口。

5.4.1 创建模板

方法	**API**	**说明**
POST	/v3/iot/management/schema	创建模板

请求参数

参数名称	**参数类型**	**是否必须**	**说明**
name	String	必须	模板名称
description	String	可选	模板说明
properties	List<SchemaProperty>	必须	模板属性列表

返回参数

参数名称	**参数类型**	**说明**
schemaId	String	模板ID

请求示例

```

POST /v3/iot/management/schema HTTP/1.1
Host:iotdm.gz.baidubce.com
Authorization:{authorization}
Content-Type: application/json; charset=utf-8
{
    "name": "myFirstSchema",
    "description": "description",
    "properties": [
        {
            "name": "temperature",
            "type": "number",
            "displayName": "温度",
            "defaultValue": "38.2",
            "unit": "deg"
        },
        {
            "name": "pressure",
            "type": "number",
            "displayName": "压力",
            "defaultValue": "9",
            "unit": "MPa"
        }
    ]
}

```

```
]  
}
```

[返回示例](#)

```
HTTP/1.1 200 OK  
Content-Type: application/json; charset=utf-8  
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5dddac970054  
{  
    "schemaid": "ab13ef935b84173a0f41f90c33daa87"  
}
```

5.4.2 删除模板

方法	**API**	**说明**
DELETE	/ v3/ iot/ management/ schema/{schemaId}	根据schemaId删除模板

[请求示例](#)

```
DELETE /v3/iot/management/schema/mySchema HTTP/1.1  
Host:iothdm.gz.baidubce.com  
Authorization:{authorization}
```

[返回示例](#)

```
HTTP/1.1 200 OK  
Content-Type: application/json; charset=utf-8  
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5dddac970054
```

注：删除模板，要求没有设备引用该模板。否则，会抛出异常。

5.4.3 获取模板列表

方法	**API**	**说明**
GET	/v3/iot/management/schema? pageNo=xx&pageSize=xx&order=xx&orderBy=xx&key=xx	根据条件查询模板列表

请求参数

参数名称	**参数类型**	**是否必须**	**说明**
pageNo	int	可选	获取列表在查询结果的页码， 默认为1
pageSize	int	可选	一页所包含的最大模板数量， 默认为0
order	String	可选	查询结果升序或降序排列， asc desc， 默认asc
orderBy	String	可选	排序的索引列， name createTime lastUpdatedTime， 默认name
key	String	可选	查询关键字， 模板名称

返回参数

参数名称	**参数类型**	**说明**
totalCount	int	模板总数
pageNo	int	当前页码
pageSize	int	一页所包含的最大模板数量
result	List<Schema>	模板参数列表

请求示例

```
GET /v3/iot/management/schema?pageNo=1&pageSize=200&order=asc&orderBy=name&key=myFirstSchema HTTP/  
1.1  
Host:iothdm.gz.baidubce.com  
Authorization:{authorization}
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
    "totalCount": 1,
    "pageNo": 1,
    "pageSize": 200,
    "result": [
        {
            "id": "1234939554",
            "name": "myFirstSchema",
            "description": " desc",
            "createTime": 1494904250,
            "lastUpdatedTime": 1494904250,
            "properties": [
                {
                    "name": "temperature",
                    "type": "number",
                    "displayName": "温度",
                    "defaultValue": "38.2",
                    "unit": "deg"
                },
                {
                    "name": "pressure",
                    "type": "number",
                    "displayName": "压力",
                    "defaultValue": "9",
                    "unit": "MPa"
                }
            ]
        }
    ]
}
```

5.4.4 获取模板

方法	**API**	**说明**
GET	/ v3/ iot/ management/schema/{schema_id} 根据模板ID获取模板详情	

[返回参数](#)

一个Schema实例

请求示例

```
GET /v3/iot/management/schema/1234939554 HTTP/1.1
Host:iothdm.gz.baidubce.com
Authorization:{authorization}
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
    "id": "1234939554",
    "name": "myFirstSchema",
    "description": " desc",
    "createTime": 1494904250,
    "lastUpdatedTime": 1494904250,
    "properties": [
        {
            "name": "temperature",
            "type": "number",
            "displayName": "温度",
            "defaultValue": "38.2",
            "unit": "deg"
        },
        {
            "name": "pressure",
            "type": "number",
            "displayName": "压力",
            "defaultValue": "9",
            "unit": "MPa"
        }
    ]
}
```

5.4.5 更新模板

方法	**API**	**说明**
PUT	/v3/iot/management/schema/{schema_id} 根据模板ID更新设备模板	

请求参数

参数名称	**参数类型**	**是否必须**	**说明**
description	String	可选	模板说明
properties	List<SchemaProperty>	可选	模板属性列表

请求示例

```

PUT /v3/iot/management/schema/1234939554 HTTP/1.1
Host:iothdm.gz.baidubce.com
Authorization:{authorization}
{
  "description": "new description",
  "properties": [
    {
      "name": "temperature",
      "type": "number",
      "displayName": "温度",
      "defaultValue": "38.3",
      "unit": "deg"
    },
    {
      "name": "pressure",
      "type": "number",
      "displayName": "压力",
      "defaultValue": "9",
      "unit": "MPa"
    },
    {
      "name": "speed",
      "type": "number",
      "displayName": "速度",
      "defaultValue": "5",
      "unit": "mps"
    }
  ]
}

```

[返回示例](#)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
```

5.5 参数定义

5.5.1 DeviceList参数列表

参数名称	参数类型	说明
devices	List<String>	需要进行操作的设备名称列表

5.5.2 DeviceProfile参数列表

参数名称	参数类型	说明
DeviceBasicInfo中对应项	DeviceBasicInfo	继承 DeviceBasicInfo，描述注册基本信息以及模型使用信息
attributes	JsonNode	json对象，存储设备自定义的相关属性
device	DeviceAttributes	DeviceAttributes 对象，描述设备属性相关信息

5.5.3 DeviceBasicInfo参数列表

参数名称	参数类型	说明
id	String	设备ID
name	String	设备名称
description	String	设备描述
createTime	DateTime	设备创建时间
state	String	设备状态
lastActiveTime	DateTime	设备最后一次的活跃时间
schemaId	String	设备采用模型Id

参数名称	参数类型	说明
schemaName	String	设备采用模型名称
favourite	boolean	是否收藏， 默认false

5.5.4 DeviceAttributes参数列表

参数名称	参数类型	说明
reported	JsonNode	json对象， 存储设备属性实际值
desired	JsonNode	json对象， 对出设备属性的期望值
profileVersion	Integer	设备属性版本号
lastUpdatedTime	JsonNode	与 reported 和 desired 属性一一对应， 相应的属性值为更新的时间戳

5.5.5 DeviceView参数列表

参数名称	参数类型	说明
DeviceBasicInfo中对应项	DeviceBasicInfo	继承DeviceBasicInfo， 描述注册基本信息以及模型使用信息
view	DeviceViewAttributes	DeviceViewAttributes 对象， 描述按照模板合并后对应的属性信息

5.5.6 DeviceViewAttributes参数列表

参数名称	参数类型	说明
profileVersion	Integer	设备属性版本号
properties	JsonNode	json数组， 每个数组元素都是一个模板中对应的属性

5.5.7 DeviceAccessDetail参数列表

参数名称	参数类型	说明
tcpEndpoint	String	tcp协议的物接入endpoint
sslEndpoint	String	支持ssl的物接入endpoint
username	String	endpointName/ thing-Name
key	String	物接入Thing密钥

参数名称	**参数类型**	**说明**
name	String	属性名称
displayName	String	属性显示名称
type	String	属性数据类型, string number bool
unit	String	数据单位
defaultValue	String	数据默认值

SchemaProperty参数列表

参数名称	**参数类型**	**说明**
id	String	模板ID
name	String	模板名称
description	String	模板说明
createTime	long	模板创建时间
lastUpdatedTime	long	模板更新时间
properties	List<SchemaProperty>	模板属性列表

Schema参数列表

第6章

API参考 - V2

6.1 介绍

6.1.1 简介

物管理API V1只能管理物管理服务所在实例（即物接入实例，在物管理服务中创建的设备都归属在一个指定的物接入实例下）下的设备。V2版本主要对物管理提供了multi-endpoints的管理，支持用户创建多个物接入实例，并在物接入实例的维度下管理设备。（暂不支持设备组的相关操作）

6.1.2 调用方式

概述 物管理API的设计采用了Restful风格，每个API功能（也可以称之为资源）都使用URI（Universal Resource Identifier）来唯一确定。对资源的请求方式是通过向资源对应的URI发送标准的HTTP请求，比如GET、PUT、POST等，同时，请求需要遵守签名算法，并包含约定的请求参数

通用约定

- 所有编码都采用UTF-8
- 日期格式采用yyyy-MM-dd方式，如2015-08-10
- 时间格式采用UTC格式：yyyy-MM-ddTHH:mm:ssZ, 如2015-08-20T01:24:32Z
- Content-type为application/json; charset=UTF-8
 - object类型的key必须使用双引号（“”）括起来
 - object类型的key必须使用lowerCamelCase表示

头域 (Header)	是否必须	说明
Authorization	必须	包含Access Key与请求签名
Host	必须	包含API的域名
x-bce-date	必须	表示时间的字符串，符合时间格式要求
Content-Type	可选	application/json; charset=utf-8
x-bce-content-sha256	可选	表示内容部分的SHA256签名的十六进制字符串。这里内容指HTTP Request Payload Body。即Content部分在被HTTP encode之前的原始数据

说明：

公共头域将在每个物管理API中出现，是必需的头域，其中x-bce-content-sha256头域只出现在POST和PUT请求中。

头域 (Header)	说明
Content-Type	只支持 JSON 格式， application/json; charset=utf-8
x-bce-request-id	后端生成，并自动设置到响应头域中

公共响应头

响应状态码 返回的响应状态码遵循[RFC 2616 section 6.1.1](#)

- 1xx: Informational - Request received, continuing process.
- 2xx: Success - The action was successfully received, understood, and accepted.
- 3xx: Redirection - Further action must be taken in order to complete the request.
- 4xx: Client Error - The request contains bad syntax or cannot be fulfilled.
- 5xx: Server Error - The server failed to fulfill an apparently valid request.

通用错误返回格式 当调用接口出错时，将返回通用的错误格式。Http的返回状态码为4xx或5xx，返回的消息体将包括全局唯一的请求、错误代码以及错误信息。调用方可根据错误码以及错误信息定位问题，当无法定位到错误原因时，可以发工单联系百度技术人员，并提供requestid以便于快速地帮助您解决问题。

消息体定义

参数名	类型	说明
requestId	String	请求的唯一标识
code	String	错误类型代码
message	String	错误的信息说明

错误返回示例

```
{
    "requestId": "47e0ef1a-9bf2-11e1-9279-0100e8cf109a",
    "code": "NoSuchKey",
    "message": "The resource you requested does not exist"
}
```

Code	Message	HTTP Status Code	说明
BceValidationException	[param]: [param]=[Validation criteria]	400	无效的[param]参数
MoneyNotEnough	Money not enough to complete the current request	400	余额不足以完成当前的请求操作
SignatureDoesNotMatch	The request signature we calculated does not match the signature you provided. Check your Secret Access Key and signing method. Consult the service doc- umentation for details	400	Authorization 头域 中附带的签名和服 务端验证不一致
InvalidAccessKeyId	The Access Key ID you provided does not exist in our records	403	Access Key ID不存 在
ServiceInternal Error	Service internal er- ror occurred	500	内部服务发生错误

公共错误码

签名认证 物管理API会对每个访问的请求进行身份认证，以保障用户的安全。安全认证采用Access Key与请求签名机制。Access Key由Access Key ID和Secret Access Key组成，均为字符串，由百度云官方颁发给用户。其中Access Key ID用于标识用户身份，Access Key Secret 是用于加密签名字符串和服务器端验证签名字符串的密钥，必须严格保密。

对于每个HTTP请求，用户需要使用下文所描述的方式生成一个签名字符串，并将认证字符串放在HTTP请求的Authorization头域里。

签名字符串格式

bce-auth-v{version}/{accessKeyId}/{timestamp}/{expireTime}/{signedHeaders}/{signature}

其中：

- version是正整数，目前取值为1。
- timestamp是生成签名时的时间。时间格式符合[通用约定](#)。
- expireTime表示签名有效期限，单位为秒，从timestamp所指定的时间开始计算。
- signedHeaders是签名算法中涉及到的头域列表。头域名字之间用分号（;）分隔，如host;x-bce-date。列表按照字典序排列。当signedHeaders为空时表示取默认值。
- signature是256位签名的十六进制表示，由64个小写字母组成，生成方式由如下[签名生成算法](#)给出。

签名生成算法 有关签名生成算法的具体介绍，请参看[鉴权认证机制](#)。

6.1.3 多区域选择

物管理目前仅支持“华南-广州”区域，区域的API地址为：iotdm.gz.baidubce.com。

6.2 Endpoint管理

6.2.1 创建Endpoint

方法	API	说明
POST	/v2/iot/management/endpoint?clientToken={clientToken}	创建endpoint

请求参数

参数名称	参数类型	是否必须	说明
clientToken	String	必须	用于保证接口等幂性
endpointName	String	必须	endpoint名称
newHub	String	必须	当前取值只能为 true, 表示需要创建新的物接入 endpoint

[返回参数](#)

参数名称	参数类型	说明
endpointName	String	endpoint名称
accountUuid	String	用户ID
createTime	String	创建时间

[请求示例](#)

```
POST /v2/iot/management/endpoint?clientToken=993ff7e9-018b-4246-a7ba-5ddac970054 HTTP/
1.1
Host: iothub.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
  "endpointName": "myendpoint",
  "newHub": true
}
```

[返回示例](#)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
  "endpointName": "myendpoint",
  "accountUuid": "bfeabccet3ding2048572716",
  "createTime": "2016-05-31T08:30:00"
}
```

6.2.2 删除Endpoint

方法	API	说明
DELETE	/v2/iot/management/endpoint/ {endpointName}?cleanHub=true	删除endpoint

请求参数

参数名称	参数类型	是否必须	说明
endpointName	String	必须	endpoint名称
cleanHub	String	必须	是否删除物接入中的 endpoint, 当前取值只能为true

请求示例

```
DELETE /v2/iot/management/endpoint/myendpoint?cleanHub=true HTTP/1.1
Host: iothub.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
```

6.2.3 获取endpoint列表

方法	API	说明
GET	/v2/iot/management/endpoint	获取endpoint列表

返回参数

参数名称	参数类型	说明
amount	int	用户创建的endpoint数量
endpointList	List<EndpointInResponse>	EndpointInResponse列表

请求示例

```
GET /v2/iot/management/endpoint HTTP/1.1
Host: iothub.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
  "amount": 1,
  "endpointList": [
    {
      "endpointName": "myendpoint",
      "accountUuid": "bfeabccet3ding2048572716",
      "createTime": "2016-05-31T08:30:00"
    }
  ]
}
```

6.3 设备列表管理

6.3.1 创建设备

方法	API	说明
POST	/ v2/ iot/ management/ endpoint/ {endpointName}/ device? clientToken={clientToken}	创建设备

请求参数

参数名称	参数类型	是否必须	说明
endpointName	String	必须	endpoint实例名称
clientToken	String	必须	用于保证接口等幂性
amount	int	必须	需要创建的设备个数
bootstrap	boolean	必须	是否需要在物接入中创建thing
devices	List<String>	可选	设备名称列表，若为空，后端自动生成
parent	String	可选	设备组ID
description	String	可选	设备描述

返回参数

参数名称	参数类型	说明
amount	Integer	创建的设备个数
devices	List<String>	设备名称列表
things	List<ThingDetail>	ThingDetail列表

请求示例

```

POST /v2/iot/management/endpoint/myendpoint/device?clientToken=993ff7e9-018b-4246-a7ba-5ddac970054 HTTP/1.1
Host: iothub.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
  "amount": "2",
  "devices": [
    "device-1",
    "device-2"
  ],
  "bootstrap": true
}

```

[返回示例](#)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
  "amount": "2",
  "devices": [
    "device-1",
    "device-2"
  ],
  "things": [
    {
      "endpoint": "023a72e64ecc4e52a1a988c72fa219bd",
      "thing": "device-1",
      "device": "device-1",
      "principal": "iot_dm_princepal_device_1",
      "policy": "iot_dm_policy_device_1",
      "key": "bWwCxGwaw3boV48NqsuG+XVaHpxfKdMPvmdJzNObvby="
    },
    {
      "endpoint": "023a72e64ecc4e52a1a988c72fa219bd",
      "thing": "device-2",
      "device": "device-2",
      "principal": "iot_dm_princepal_device_2",
      "policy": "iot_dm_policy_device_2",
      "key": "YmjWVckS53n19v69Q61HR9Np7Gmk97txS9LgVrlx1k="
    }
  ]
}
```

6.3.2 删除设备

方法	API	说明
PUT	/v2/iot/management/endpoint/{endpointName}/device?remove	删除设备

[请求参数](#)

参数名称	参数类型	是否必须	说明
endpointName	String	必须	endpoint名称

参数名称	参数类型	是否必须	说明
cleanThing	boolean	必须	是否删除物接入中对应的thing
deviceOperation	DeviceOperation	必须	一个DeviceOperation对象

请求示例

```
PUT /v2/iot/management/endpoint/myendpoint/device?remove HTTP/1.1
Host: iothub.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
  "clientThing": false,
  "deviceOperation": {
    "devices": [
      "device-1",
      "device-2"
    ]
  }
}
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
```

6.3.3 获取设备Profile

方法	API	说明
GET	/v2/iot/management/endpoint/{endpointName}/device/{deviceName}	获取设备Profile

请求参数

参数名称	参数类型	是否必须	说明
endpointName	String	必须	endpoint名称
deviceName	String	必须	设备名称

[返回参数](#)

一个DeviceProfile对象

[请求示例](#)

```
GET /v2/iot/management/endpoint/myendpoint/device/mydevice HTTP/1.1
Host: iothub.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
```

[返回示例](#)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
  "id": "0e357fac55b64bfa8c5c7e9fd56ee320",
  "name": "mydevice",
  "createTime": "2016-05-31T08:30:00",
  "state": "running",
  "enable": true,
  "lastActiveTime": "2016-05-31T09:00:00",
  "attributes": {
    "region": "Shanghai"
  },
  "device": {
    "reported": {
      "firewareVersion": "1.0.0",
      "light": "green"
    },
    "desired": {
      "light": "red"
    }
  },
  "profileVersion": 10,
  "timestampe": "2016-06-02T09: 00: 00"
}
```

6.3.4 设备查询

方法	API	说明
PUT	/v2/iot/management/endpoint/{endpointName}/device?query	查询设备

请求参数

请求名称	参数类型	是否必须	说明
endpointName	String	必须	endpoint名称
condition	String	必须	查询条件字符串
page	Device.Page	可选	Device.Page 对象， 查询结果分页设置

返回参数

参数名称	参数类型	说明
deviceProfiles	List<DeviceProfile>	由DeviceProfile对象组成的列表
amount	Integer	满足查询条件的设备总数

请求示例

```
PUT /v2/iot/management/endpoint/myendpoint/device?query HTTP/1.1
Host: iothub.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
  "condition": "{\"registry.reported.firewareVersion\":\"running\", \"attributes.region\": \"Shanghai\"}"
}
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
```

```
{
  "amount": 1,
  "deviceProfiles": [
    {
      "id": "0e357fac55b64bfa8c5c7e9fd56ee320",
      "name": "mydevice",
      "registryId": "0e357fac55b64bfa8c5c7e9fd56ee320",
      "state": "running",
      "enable": true,
      "createTime": "2016-05-31T08:30:00",
      "lastActiveTime": "2016-05-31T09:00:00",
      "attributes": {
        "region": "Shanghai"
      },
      "device": {
        "reported": {
          "light": "green"
        }
      },
      "profileVersion": 10,
      "timestampe": "2016-06-02T09: 00: 00"
    }
  ]
}
```

6.3.5 获取设备接入详情

方法	API	说明
GET	/v2/iot/management/endpoint/{endpointName}/device/{deviceName}/access-Detail	获取设备的接入详情

请求参数

参数名称	参数类型	是否必须	说明
endpointName	String	必须	endpoint名称
deviceName	String	必须	设备名称

返回参数

参数名称	参数类型	说明
protocol	String	设备接入协议, 目前仅支持 mqtt
deviceName	String	设备名称
endpointName	String	endpoint名称
endpoints	List<String>	设备接入的endpoint
pubTopics	List<String>	pub的topic列表
subTopics	List<String>	sub的topic列表

请求示例

```
GET /v2/iot/management/endpoint/myendpoint/device/mydevice/accessDetail HTTP/
1.1
Host: iothub.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
  "protocol": "MQTT",
  "deviceName": "mydevice",
  "endpointName": "54d83f0aeef347599eef0ec83dd6aa8f",
  "endpoints": [
    "tcp://54d83f0aeef347599eef0ec83dd6aa8f.mqtt.iot.gz.baidubce.com:1883",
    "ssl://54d83f0aeef347599eef0ec83dd6aa8f.mqtt.iot.gz.baidubce.com:1884",
    "wss://54d83f0aeef347599eef0ec83dd6aa8f.mqtt.iot.gz.baidubce.com:8884"
  ],
  "pubTopics": ["\$baidu/iot/management/mydevice"],
  "subTopics": ["\$baidu/iot/device/mydevice"]
}
```

6.3.6 更新设备属性

方法	API	说明
PUT	/v2/iot/management/endpoint/{endpointName}/device?updateProfile	修改设备属性

请求参数

参数名称	参数类型	是否必须	说明
endpointName	String	必须	endpoint名称
attributes	JSONObject	可选	需要更新的 attributes
device	DeviceAttributes	可选	需要更新的 device 属性
deviceOperation	DeviceOperation	必须	设备操作的相关参数

请求示例

```
PUT /v2/iot/management/endpoint/myendpoint/device?updateProfile HTTP/1.1
Host: iothub.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
  "device": {
    "desired": {
      "light": "red"
    }
  },
  "deviceOperation": {
    "devices": [
      "device-1",
      "device-2"
    ]
  }
}
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
```

x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054

6.3.7 更新设备注册表信息

方法	API	说明
PUT	/v2/iot/management/endpoint/{endpointName}/device?updateRegistry	修改设备注册表信息

请求参数

参数名称	参数类型	是否必须	说明
endpointName	String	必须	endpoint名称
description	String	可选	需要更新的设备描述
parent	String	可选	需要更新的设备组ID
deviceOperation	DeviceOperation	必须	设备操作相关参数

请求示例

```
PUT /v2/iot/management/endpoint/myendpoint/device?updateRegistry HTTP/1.1
Host: iothub.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
  "description": "new device description",
  "deviceOperation": {
    "devices": [
      "device-1",
      "device-2"
    ]
  }
}
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
```

x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054

6.3.8 禁用设备

方法	API	说明
PUT	/v2/iot/management/endpoint/{endpointName}/device?disable	禁用设备

请求参数

参数名称	参数类型	是否必须	说明
endpointName	String	必须	endpoint名称

请求参数

一个[DeviceOperation](#)对象

请求示例

```
PUT /v2/iot/management/endpoint/myendpoint/device?disable HTTP/1.1
Host: iothub.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
  "devices": [
    "device-1",
    "device-2"
  ]
}
```

6.3.9 启用设备

方法	API	说明
PUT	/v2/iot/management/endpoint/{endpointName}/device?enable	启用

请求参数

参数名称	参数类型	是否必须	说明
endpointName	String	必须	endpoint名称
deviceOperation	DeviceOperation	必须	一个 DeviceOperation对象

[请求示例](#)

```
PUT /v2/iot/management/endpoint/myendpoint/device?enable HTTP/1.1
Host: iothub.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
  "devices": [
    "device-1",
    "device-2"
  ]
}
```

[返回示例](#)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
```

6.4 设备操作

6.4.1 设备重启

方法	API	说明
PUT	/v2/iot/management/endpoint/{endpointName}/device?reboot	设备重启

[请求参数](#)

参数名称	参数类型	是否必须	说明
endpointName	String	必须	endpoint名称

参数名称	参数类型	是否必须	说明
deviceOperation	DeviceOperation	必须	一个 DeviceOperation 对象

[请求示例](#)

```
PUT /v2/management/endpoint/myendpoint/device?shutdown HTTP/1.1
Host: iothub.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
  "devices": [
    "device-1",
    "device-2"
  ]
}
```

[返回示例](#)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
```

6.5 参数定义

6.5.1 DeviceProfile参数列表

参数名称	参数类型	说明
id	String	设备ID
name	String	设备名称
parent	String	设备所在组ID
description	String	设备描述
createTime	DateTime	设备创建时间
state	String	设备状态
enable	bool	设备禁用、启用状态
lastActiveTime	DateTime	设备最后一次的活跃时间

参数名称	参数类型	说明
attributes	JsonNode	json对象，存储设备自定义的相关属性
device	DeviceAttributes	DeviceAttributes 对象，描述设备属性相关信息

6.5.2 DeviceAttributes参数列表

参数名称	参数类型	说明
reported	JsonNode	json对象，存储设备属性实际值
desired	JsonNode	json对象，对出设备属性的期望值
profileVersion	Integer	设备属性版本号
timestamp	Date	设备属性更新的时间戳

6.5.3 DeviceOperation参数列表

参数名称	参数类型	说明
condition	String	操作的设备组（按查询条件定义设备组）
devices	List<String>	设备列表

注：DeviceOperation中， condition、 devices为两种定义操作设备对象的方法，使用中二选一。

6.5.4 Device.Page参数列表

参数名称	参数类型	说明
orderBy	String	排序索引列， name createTime lastActiveTime
order	String	按升序或降序返回结果， desc asc
pageNo	Integer	需要返回的页数
pageSize	Integer	每一页包含的设备信息数量

6.5.5 ThingDetail参数列表

参数名称	参数类型	说明
endpoint	String	endpoint名称
thing	String	thing名称
device	String	设备名称
principal	String	用于物管理的身份
policy	String	用于物管理测策略
key	String	用于物管理的key

6.5.6 EndpointInResponse参数列表

参数名称	参数类型	说明
endpointName	String	endpoint名称
accountUuid	String	用户id
createTime	String	创建时间

第7章

API参考 - V1

7.1 介绍

7.1.1 简介

注意

物管理API V1只能管理物管理服务所在实例（即物接入实例，在物管理服务中创建的设备都归属在一个指定的物接入实例下）下的设备。如果需要管理其他物接入服务实例下的设备，请查看[API参考 - V2](#)。

百度云IoT物管理套件，提供用户在云端管理设备的能力。用户能够获取并控制设备状态、进行设备的批量操作以及设备诊断。一方面，设备主动向物管理中心更新状态信息；另一方面，控制端也可以通过和设备管理中心交互反控设备的行为，比如设备状态更新、OTA、关机以及重启等。因此，设备管理中心既需要负责和设备端交互，又要负责和控制端交互。设备端的交互主要基于MQTT协议，而控制端通过HTTP通信。IoT Device Management API主要包括控制端的相关功能，以Restful API的形式提供。

7.1.2 调用方式

概述 物管理API的设计采用了Restful风格，每个API功能（也可以称之为资源）都使用URI（Universal Resource Identifier）来唯一确定。对资源的请求方式是通过向资源对应的URI发送标准的HTTP请求，比如GET、PUT、POST等，同时，请求需要遵守签名算法，并包含约定的请求参数

通用约定

- 所有编码都采用UTF-8
- 日期格式采用yyyy-MM-dd方式，如2015-08-10
- 时间格式采用UTC格式：yyyy-MM-ddTHH:mm:ssZ, 如2015-08-20T01:24:32Z

- Content-type为application/json; charset=UTF-8
 - object类型的key必须使用双引号（“”）括起来
 - object类型的key必须使用lowerCamelCase表示

头域 (Header)	是否必须	说明
Authorization	必须	包含Access Key与请求签名
Host	必须	包含API的域名
x-bce-date	必须	表示时间的字符串，符合时间格式要求
Content-Type	可选	application/json; charset=utf-8
x-bce-content-sha256	可选	表示内容部分的SHA256签名的十六进制字符串。这里内容指HTTP Request Payload Body。即Content部分在被HTTP encode之前的原始数据

说明：

公共头域将在每个物管理API中出现，是必需的头域，其中x-bce-content-sha256头域只出现在POST和PUT请求中。

头域 (Header)	说明
Content-Type	只支持JSON格式，application/json; charset=utf-8
x-bce-request-id	后端生成，并自动设置到响应头域中

公共响应头

响应状态码 返回的响应状态码遵循[RFC 2616 section 6.1.1](#)

- 1xx: Informational - Request received, continuing process.
- 2xx: Success - The action was successfully received, understood, and accepted.
- 3xx: Redirection - Further action must be taken in order to complete the request.
- 4xx: Client Error - The request contains bad syntax or cannot be fulfilled.
- 5xx: Server Error - The server failed to fulfill an apparently valid request.

通用错误返回格式 当调用接口出错时，将返回通用的错误格式。Http的返回状态码为4xx或5xx，返回的消息体将包括全局唯一的请求、错误代码以及错误信息。调用方可根据错误码以及错误信息定位问题，当无法定位到错误原因时，可以发工单联系百度技术人员，并提供requestId以便于快速地帮助您解决问题。

消息体定义

参数名	类型	说明
requestId	String	请求的唯一标识
code	String	错误类型代码
message	String	错误的信息说明

错误返回示例

```
{
  "requestId": "47e0ef1a-9bf2-11e1-9279-0100e8cf109a",
  "code": "NoSuchKey",
  "message": "The resource you requested does not exist"
}
```

Code	Message	HTTP Status Code	说明
BceValidationException	[param]: [param]=[Validation criteria]	400	无效的[param]参数
MoneyNotEnough	Money not enough to complete the current request	400	余额不足以完成当前的请求操作
SignatureDoesNotMatch	The request signature we calculated does not match the signature you provided. Check your Secret Access Key and signing method. Consult the service doc- umentation for details	400	Authorization 头域 中附带的签名和服 务端验证不一致
InvalidAccessKeyId	The Access Key ID you provided does not exist in our records	403	Access Key ID不存 在

Code	Message	HTTP Status Code	说明
ServiceInternal Error	Service internal error occurred	500	内部服务发生错误

公共错误码

签名认证 物管理API会对每个访问的请求进行身份认证，以保障用户的安全。安全认证采用Access Key与请求签名机制。Access Key由Access Key ID和Secret Access Key组成，均为字符串，由百度云官方颁发给用户。其中Access Key ID用于标识用户身份，Access Key Secret 是用于加密签名字符串和服务器端验证签名字符串的密钥，必须严格保密。

对于每个HTTP请求，用户需要使用下文所描述的方式生成一个签名字符串，并将认证字符串放在HTTP请求的Authorization头域里。

签名字符串格式

bce-auth-v{version}/{accessKeyId}/{timestamp}/{expireTime}/{signedHeaders}/{signature}

其中：

- version是正整数，目前取值为1。
- timestamp是生成签名时的时间。时间格式符合[通用约定](#)。
- expireTime表示签名有效期限，单位为秒，从timestamp所指定的时间开始计算。
- signedHeaders是签名算法中涉及到的头域列表。头域名字之间用分号（;）分隔，如host;x-bce-date。列表按照字典序排列。当signedHeaders为空时表示取默认值。
- signature是256位签名的十六进制表示，由64个小写字母组成，生成方式由如下[签名生成算法](#)给出。

签名生成算法 有关签名生成算法的具体介绍，请参看[鉴权认证机制](#)。

7.1.3 多区域选择

物管理目前仅支持“华南-广州”区域，区域的API地址为：iotdm.gz.baidubce.com。

7.2 设备列表管理

7.2.1 创建设备

方法	API	说明
POST	/v1/iot/management/device?clientToken={clientToken}	创建设备

请求参数

参数名称	参数类型	是否必须	说明
clientToken	String	必须	用于保证接口等幂性
amount	int	必须	需要创建的设备个数
bootstrap	boolean	必须	是否需要在物接入中创建thing
devices	List<String>	可选	设备名称列表，若为空，后端自动生成
parent	String	可选	设备组ID
description	String	可选	设备描述

返回参数

参数名称	参数类型	说明
amount	Integer	创建的设备个数
devices	List<String>	设备名称列表
things	List<ThingDetail>	ThingDetail列表

请求示例

```
POST /v1/iot/management/device?clientToken=993ff7e9-018b-4246-a7ba-5ddac970054 HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
    "amount": "2",
    "devices": [
        "device-1",
        "device-2"
    ]
}
```

```
        "device-2"  
    ],  
    "bootstrap":true  
}
```

[返回示例](#)

```
HTTP/1.1 200 OK  
Content-Type: application/json; charset=utf-8  
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054  
{  
    "amount": "2",  
    "devices": [  
        "device-1",  
        "device-2"  
    ],  
    "things": [  
        {  
            "endpoint": "023a72e64ecc4e52a1a988c72fa219bd",  
            "thing": "device-1",  
            "device": "device-1",  
            "principal": "iot_dm_principal_device_1",  
            "policy": "iot_dm_policy_device_1",  
            "key": "bLwCxGwaw3boV48NqsuG+XVaHpxfKdMPvmdJzNObvbY="  
        },  
        {  
            "endpoint": "023a72e64ecc4e52a1a988c72fa219bd",  
            "thing": "device-2",  
            "device": "device-2",  
            "principal": "iot_dm_principal_device_2",  
            "policy": "iot_dm_policy_device_2",  
            "key": "YmjWVckS53n19v69Q61HR9Np7Gmk97txS9LgVrlx1k="  
        }  
    ]  
}
```

7.2.2 删除设备

方法	API	说明
PUT	/v1/iot/management/device?remove	删除设备

请求参数

参数名称	参数类型	是否必须	说明
cleanThing	boolean	必须	是否删除物接入中对应的thing
deviceOperation	DeviceOperation	必须	一个 DeviceOperation对象

请求示例

```
PUT /v1/iot/management/device?remove HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
    "cleanThing": false,
    "deviceOperation": {
        "devices": [
            "device-1",
            "device-2"
        ]
    }
}
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
```

7.2.3 获取设备Profile

方法	API	说明
GET	/v1/iot/management/device/{deviceName}	获取设备Profile

返回参数

一个[DeviceProfile对象](#)

请求示例

```
GET /v1/iot/management/device/mydevice HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
    "id": "0e357fac55b64bfa8c5c7e9fd56ee320",
    "name": "mydevice",
    "createTime": "2016-05-31T08:30:00",
    "state": "online",
    "enable": true,
    "lastActiveTime": "2016-05-31T09:00:00",
    "attributes": {
        "region": "Shanghai"
    },
    "device": {
        "reported": {
            "firewareVersion": "1.0.0",
            "light": "green"
        },
        "desired": {
            "light": "red"
        },
        "profileVersion": 10,
        "timestampe": "2016-06-02T09:00:00"
    }
}
```

7.2.4 设备查询

方法	API	说明
PUT	/v1/iot/management/device?query	查询设备

请求参数

请求名称	参数类型	是否必须	说明
condition	String	必须	查询条件字符串
page	Device.Page	可选	Device.Page 对象，查询结果分页设置

返回参数

参数名称	参数类型	说明
deviceProfiles	List<DeviceProfile>	由DeviceProfile对象组成的列表
amount	Integer	满足查询条件的设备总数

请求示例

```
PUT /v1/iot/management/device?query HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
    "condition": "{\"registry.reported.firewareVersion\": \"running\", \
\"attributes.region\": \"Shanghai\"}"
}
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
    "amount": 1,
    "deviceProfiles": [
        {
            "id": "0e357fac55b64bfa8c5c7e9fd56ee320",
            "name": "mydevice",
            "registryId": "0e357fac55b64bfa8c5c7e9fd56ee320",
            "state": "online",
            "enable": true,
            "createTime": "2016-05-31T08:30:00",
            "lastReportTime": "2016-05-31T08:30:00"
        }
    ]
}
```

```

        "lastActiveTime": "2016-05-31T09:00:00",
        "attributes": {
            "region": "Shanghai"
        },
        "device": {
            "reported": {
                "light": "green"
            }
        },
        "profileVersion": 10,
        "timestampe": "2016-06-02T09: 00: 00"
    }
}
]
}

```

7.2.5 获取设备接入详情

方法	API	说明
GET	/v1/iot/management/device/{deviceName}/accessDetail	获取设备的接入详情

返回参数

参数名称	参数类型	说明
protocol	String	设备接入协议，目前仅支持mqtt
deviceName	String	设备名称
endpointName	String	endpoint名称
endpoints	List<String>	设备接入的endpoint
pubTopics	List<String>	pub的topic列表
subTopics	List<String>	sub的topic列表

请求示例

```

GET /v1/iot/management/device/mydevice/accessDetail HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json: charset=utf-8

```

[返回示例](#)

```

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
    "protocol": "MQTT",
    "deviceName": "mydevice",
    "endpointName": "54d83f0aeeef347599eef0ec83dd6aa8f",
    "endpoints": [
        "tcp://54d83f0aeeef347599eef0ec83dd6aa8f.mqtt.iot.gz.baidubce.com:1883",
        "ssl://54d83f0aeeef347599eef0ec83dd6aa8f.mqtt.iot.gz.baidubce.com:1884",
        "wss://54d83f0aeeef347599eef0ec83dd6aa8f.mqtt.iot.gz.baidubce.com:8884"
    ],
    "pubTopics": ["\$baidu/iot/management/mydevice"],
    "subTopics": ["\$baidu/iot/device/mydevice"]
}

```

7.2.6 更新设备属性

方法	API	说明
PUT	/v1/iot/management/device?updateProfile	修改设备属性

[请求参数](#)

参数名称	参数类型	是否必须	说明
attributes	JSONObject	可选	需要更新的 attributes
device	DeviceAttributes	可选	需要更新的 device 属性
deviceOperation	DeviceOperation	必须	设备操作的相关参数

[请求示例](#)

```

PUT /v1/iot/management/device?updateProfile HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}

```

```
Content-Type: application/json; charset=utf-8
{
  "device": {
    "desired": {
      "light": "red"
    }
  },
  "deviceOperation": {
    "devices": [
      "device-1",
      "device-2"
    ]
  }
}
```

[返回示例](#)

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054

7.2.7 更新设备注册表信息

方法	API	说明
PUT	/v1/iot/management/device?updateRegistry	修改设备注册表信息

[请求参数](#)

参数名称	参数类型	是否必须	说明
description	String	可选	需要更新的设备描述
parent	String	可选	需要更新的设备组ID
deviceOperation	DeviceOperation	必须	设备操作相关参数

[请求示例](#)

PUT /v1/iot/management/device?updateRegistry HTTP/1.1

```

Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
  "description": "new device description",
  "deviceOperation": {
    "devices": [
      "device-1",
      "device-2"
    ]
  }
}

```

[返回示例](#)

```

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054

```

7.2.8 禁用设备

方法	API	说明
PUT	/v1/iot/management/device?disable	禁用设备

[请求参数](#)

一个DeviceOperation对象

[请求示例](#)

```

PUT /v1/iot/management/device?disable HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
  "devices": [
    "device-1",
    "device-2"
  ]
}

```

[返回示例](#)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
```

7.2.9 启用设备

方法	API	说明
PUT	/v1/iot/management/device?enable	启用

[请求参数](#)

一个DeviceOperation对象

[请求示例](#)

```
PUT /v1/iot/management/device?enable HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
    "devices": [
        "device-1",
        "device-2"
    ]
}
```

[返回示例](#)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
```

7.3 设备分组管理

7.3.1 创建设备组

方法	API	说明
POST	/v1/iot/management/group?clientToken={clientToken}	创建设备组

请求参数

参数名称	参数类型	是否必须	说明
clientToken	String	必须	用于保证接口幂等性
uri	String	必须	设备组的绝对路径
description	String	可选	设备组描述

返回参数

一个DeviceGroup实例

请求示例

```
POST /v1/iot/management/group?clientToken=a0be3889-5f74-47f1-8a71-857066266ed0 HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
    "uri": "baidu-beijing"
}
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
    "id": "2f7fb58a-1134-478c-b127-12675e74bde7",
    "name": "Beijing",
    "uri": "baidu-beijing",
    "device": 0,
    "parent": "12345678-1134-478c-b127-12675e74bde7"
    "createTime": "2016-06-14T09:00:00"
}
```

7.3.2 删除设备组

方法	API	说明
DELETE	/v1/iot/management/group/{groupId}	删除设备组

请求示例

```
DELETE /v1/iot/management/group/2f7fb58a-1134-478c-b127-12675e74bde7 HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5dddac970054
```

7.3.3 修改设备组设置

方法	API	说明
PUT	/v1/iot/management/group/{groupId}	修改设备组

请求参数

参数名称	参数类型	是否必须	说明
name	String	可选	设备组名称
parent	String	可选	设备组父节点 (group)
description	String	可选	设备组描述

请求示例

```
PUT /v1/iot/management/group/2f7fb58a-1134-478c-b127-12675e74bde7 HTTP/1.1
```

```

Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
    "name": "newGroup"
}

```

[返回示例](#)

```

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054

```

7.3.4 查询设备组信息

方法	API	说明
GET	/v1/iot/management/group/{groupId} HTTP/1.1	查询设备组信息

[返回参数](#)

一个DeviceGroup对象

[请求示例](#)

```

GET /v1/iot/management/group/2f7fb58a-1134-478c-b127-12675e74bde7 HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8

```

[返回示例](#)

```

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
    "id": "2f7fb58a-1134-478c-b127-12675e74bde7",
    "name": "Beijing",
    "uri": "baidu-beijing",
    "device": 0,
}

```

```

    "parent": "12345678-1134-478c-b127-12675e74bde7",
    "createTime": "2016-06-14Z09:00:00"
}

```

7.3.5 查询指定设备组子节点列表

方法	API	说明
GET	/v1/iot/management/group/{groupId}/children	查询group类型设备组的子节点列表

[返回参数](#) |

参数名称	参数类型	说明
groups	List<DeviceGroup>	子节点列表

[请求示例](#)

```

GET /v1/iot/management/group/2f7fb58a-1134-478c-b127-12675e74bde7/children HTTP/
1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8

```

[返回示例](#)

```

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
{
  "groups": [
    {
      "id": "12345678-1134-478c-b127-12675e74bde7",
      "name": "Shanghai",
      "uri": "/Baidu/Shanghai",
      "device": 0,
      "parent": "2f7fb58a-1134-478c-b127-12675e74bde7",
      "createTime": "2016-06-14Z09:00:00"
    }
  ]
}

```

7.3.6 查询根目录下的设备组

方法	API	说明
GET	/ v1/ iot/ management/ group?root	查询根目录下的设备组

[返回参数](#)

参数名称	参数类型	说明
groups	List< DeviceGroup >	子节点列表

7.3.7 查询device设备组

方法	API	说明
GET	/ v1/ iot/ management/ group?device	查询device类型的设备组

[返回参数](#)

参数名称	参数类型	说明
groups	List< DeviceGroup >	子节点列表

注：device设备组是指在group层级结果的叶子节点以及拥有至少一个device的非叶子节点。

7.4 设备操作

7.4.1 设备重启

方法	API	说明
PUT	/ v1/ iot/ management/ device?reboot	设备重启

[请求参数](#)

一个[DeviceOperation](#)对象

请求示例

```
PUT /v1/management/device?shutdown HTTP/1.1
Host: iotdm.gz.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
{
  "devices": [
    "device-1",
    "device-2"
  ]
}
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 993ff7e9-018b-4246-a7ba-5ddac970054
```

7.5 附录

7.5.1 DeviceProfile参数列表

参数名称	参数类型	说明
id	String	设备ID
name	String	设备名称
parent	String	设备所在组ID
description	String	设备描述
createTime	DateTime	设备创建时间
state	String	设备状态
enable	bool	设备禁用、启用状态
lastActiveTime	DateTime	设备最后一次的活跃时间
attributes	JsonNode	json对象，存储设备自定义的相关属性
device	DeviceAttributes对象	DeviceAttributes，描述设备属性相关信息

7.5.2 DeviceAttributes参数列表

参数名称	参数类型	说明
desired	JsonNode	json对象，对出设备属性的期望值
profileVersion	Integer	设备属性版本号
reported	JsonNode	json对象，存储设备属性实际值
timestamp	Date	设备属性更新的时间戳

7.5.3 DeviceGroup参数列表

参数名称	参数类型	说明
id	String	节点ID
name	String	节点名称
uri	String	节点uri
device	Integer	属于该设备组的设备个数
parent	String	父节点设备组ID
description	String	节点描述
createTime	DateTime	节点创建时间

7.5.4 DeviceOperation参数列表

参数名称	参数类型	说明
condition	String	操作的设备组（按查询条件定义设备组）
devices	List<String>	设备列表

注：DeviceOperation中，condition、devices为两种定义操作设备对象的方法，使用中二选一。

7.5.5 Device.Page参数列表

参数名称	参数类型	说明
orderBy	String	排序索引列, name createTime lastActiveTime
order	String	按升序或降序返回结果, desc asc
pageNo	Integer	需要返回的页数
pageSize	Integer	每一页包含的设备信息数量

7.5.6 ThingDetail参数列表

参数名称	参数类型	说明
device	String	设备名称
endpoint	String	endpoint名称
key	String	用于物管理的key
policy	String	用于物管理测策略
principal	String	用于物管理的身份
thing	String	thing名称

第8章

Java SDK文档

8.1 V3

8.1.1 概述

本文档主要介绍IoT Device SDK的安装和使用。在使用本文档前，您需要先了解IoT Device的一些基本知识，并已经开通了Iot Device服务。若您还不了解Iot Device，可以参考[产品描述和操作指南](#)。

8.1.2 安装SDK工具包

运行环境

Java SDK工具包可在jdk1.6、jdk1.7、jdk1.8环境下运行。

安装步骤

1. 在[官方网站](#)下载Java SDK压缩工具包。
2. 将下载的**bce-java-sdk-version.zip**解压后，复制到工程文件夹中。
3. 在Eclipse右键“工程 -> Properties -> Java Build Path -> Add JARs”。
4. 添加SDK工具包**lib/bce-java-sdk-version.jar**和第三方依赖工具包**third-party/*.jar**。
其中，**version**为版本号。

SDK目录结构

```
com.baidubce
    └── auth           //BCE签名相关类
```

```

    └── http                                //BCE的Http通信相关类
    └── internal                            //SDK内部类
    └── model                               //BCE公用model类
    └── services
        └── iotdm                         //物管理服务相关类
            └── model                      //物管理内部model, 如Request

或Response
    └── IotDmV3Client.class                //物管理客户端入口类
    └── util                                //BCE公用工具类
    └── BceClientConfiguration.class        //对BCE的HttpClient的配置
    └── BceClientException.class           //BCE客户端的异常类
    └── BceServiceException.class          //与BCE服务端交互后的异常类
    └── ErrorCode.class                   //BCE通用的错误码
    └── Region.class                      //BCE提供服务的区域

```

8.1.3 创建IotDmV3Client

用户可以参考如下代码创建一个IotDmV3Client:

HTTP Client

```

String ACCESS_KEY_ID = <your-access-key-id>; // 用户的Access Key ID
String SECRET_ACCESS_KEY = <your-secret-access-key>; // 用户的Secret Access Key
String ENDPOINT = "iotdm.gz.baidubce.com";

// 创建配置
BceClientConfiguration config = new BceClientConfiguration()
.withCredentials(new DefaultBceCredentials(ACCESSKEY, SECRETKEY))
.withEndpoint(ENDPOINT);

// 初始化一个IotDmV3Client
IotDmV3Client client = new IotDmV3Client(config);

```

HTTPS Client

```

String ACCESS_KEY_ID = <your-access-key-id>; // 用户的Access Key ID
String SECRET_ACCESS_KEY = <your-secret-access-key>; // 用户的Secret Access Key
String ENDPOINT = "iotdm.gz.baidubce.com";

// 创建配置
BceClientConfiguration config = new BceClientConfiguration()
.withProtocol(Protocol.HTTPS) // 使用HTTPS协议

```

```
.withCredentials(new DefaultBceCredentials(ACCESSKEY, SECRETKEY))
.withEndpoint(ENDPOINT);

// 初始化一个IotDmV3Client
IotDmV3Client client = new IotDmV3Client(config);
```

在代码中，变量ACCESS_KEY_ID与SECRET_ACCESS_KEY是系统分配给用户的，均为字符串，用于标识用户，为访问物管理做签名认证。其中ACCESS_KEY_ID对应控制台中的“Access Key ID”，SECRET_ACCESS_KEY对应控制台的“Access Key Secret”，获取方式请参考[获取AK/SK](#)。

参数说明

BceClientConfiguration中有更多的配置项，可配置如下参数：

参数	说明
connectionTimeo utInMillis	建立连接的超时时间（单位：毫秒）
localAddress	本地地址
maxConnections	允许打开的最大HTTP连接数
protocol	连接协议类型
proxyDomain	访问NTLM验证的代理服务器的Windows域名
proxyHost	代理服务器主机地址
proxyPassword	代理服务器验证的密码
proxyPort	代理服务器端口
proxyPreemptive AuthenticationEnabled	是否设置用户代理认证
proxyUsername	代理服务器验证的用户名
proxyWorkstation	NTLM代理服务器的Windows工作站名称
retryPolicy	连接重试策略
socketBufferSizeInBytes	Socket缓冲区大小
socketTimeoutInMillis	通过打开的连接传输数据的超时时间（单位：毫秒）
userAgent	用户代理，指HTTP的User-Agent头

8.1.4 设备管理

创建设备 创建设备可以参考代码如下：

```
String deviceName = "device_name";      // 设置创建的设备名称
String schemaId = "schema_id";          // 设置绑定的模型ID
String description = "description";     // 设置对该设备的描述

CreateDeviceRequest request = new CreateDeviceRequest()
    .withDeviceName(deviceName)
    .withSchemaId(schemaId)
    .withDescription(description);

DeviceAccessDetailResponse response = client.createDevice(request);

String tcpEndpoint = response.getTcpEndpoint(); // tcp endpoint地址
String sslEndpoint = response.getSslEndpoint(); // ssl endpoint地址
String username = response.getUsername();        // 账户名，用于接入IoT hub服务
String key = response.getKey();                  // 密钥，用于接入IoT hub服务
```

删除设备 删除设备可以参考代码如下：

```
List<String> devices = Arrays.asList("device_name_1", "device_name_2"); // 设置
需要删除的设备列表

DeviceListRequest request = new DeviceListRequest().
    withDevices(devices);

DeviceListResponse response = client.removeDevices(request);

List<String> deletedDevices = response.getDevices(); // 删除设备列表
```

获取设备Profile 获取设备Profile可以参考代码如下：

```
String deviceName = "device_name";      // 设置需要查询的设备名称

DeviceProfileResponse response = client.getDeviceProfile(deviceName);

String id = response.getId();           // 设备Id
String name = response.getName();       // 设备名称
String description = response.getDescription(); // 设备描述
String state = response.getState();     // 设备当前状态
String schemaId = response.getSchemaId(); // 设备使用的模版ID
String schemaName = response.getSchemaName(); // 设备使用的模板名称
Long createTime = response.getCreateTime(); // 设备的创建时间
```

```

Long lastActiveTime = response.getLastActiveTime(); // 设备最后一次上报内容的时间
Boolean favourite = response.getFavourite(); // 设备的收藏信息
JsonNode attributes = response.getAttributes(); // 设备在服务端设置的属性
DeviceAttributes device = response.getDevice(); // 设备在设备端的属性

```

获取设备View 获取设备View可以参考代码如下：

```

String deviceName = "device_name"; // 设置需要查询的设备名称

DeviceViewResponse response = client.getDeviceView(deviceName);

String id = response.getId(); // 设备Id
String name = response.getName(); // 设备名称
String description = response.getDescription(); // 设备描述
String state = response.getState(); // 设备当前状态
String schemaId = response.getSchemaId(); // 设备使用的模版ID
String schemaName = response.getSchemaName(); // 设备使用的模板
名称
Long createTime = response.getCreateTime(); // 设备的创建时间
Long lastActiveTime = response.getLastActiveTime(); // 设备最后一次上
报内容的时间
Boolean favourite = response.getFavourite(); // 设备的收藏信息

int profileVersion = response.getProfileVersion(); // 设备影子的版本号
List<DeviceViewAttributes> device = response.getProperties(); // 设备所使用模板的
属性

```

获取及查询影子列表 获取及查询影子列表可以参考代码如下：

```

int pageNo = 1; // 设置需要获取所有查询结果的第几页
int pageSize = 10; // 设置每页返回的最大个数
String orderBy = "name"; // 设置排序的索引列，支持name createTime/
lastActiveTime
String order = "asc"; // 设置按照升序、降序排列结果，支持asc/desc
String name = "schemaName"; // 设置需要查询的属性名
// 支持设备名字查询(name)/模型名字查询(schemaName)/
服务端属性查询(attributes.***)/设备端属性查询(device.reported.***)
String value = "my_schema_name"; // 设置需要查询的属性值，对于设备端属性查询，如果相应
属性值为字符串类型，需要用""将字符串包裹
String favourite = "all"; // 设置收藏选择，支持all/true/false

DeviceProfileListResponse response = client.getDeviceProfiles(pageNo, pageSize, orderBy, order, name)

```

```
int amount = response.getAmount(); // 满足查询条件的设备数目  
int pageNo = response.getPageNo(); // 当前页页码  
int pageSize = response.getPageSize(); // 返回的每页的最大个数  
List<DeviceProfile> devices = response.getDevices(); // 当前页设备详情列表
```

获取设备接入详情 获取设备接入详情可以参考代码如下：

```
String deviceName = "device_name"; // 设置需要查询的设备名称  
  
DeviceAccessDetailResponse response = client.getDeviceAccessDetail(deviceName);  
  
String tcpEndpoint = response.getTcpEndpoint(); // tcp endpoint地址  
String sslEndpoint = response.getSslEndpoint(); // ssl endpoint地址  
String username = response.getUsername(); // 账户名，用于接入IoT hub服务  
String key = response.getKey(); // 调用此方法密钥默认返回"xxxxxxxxx",  
请在创建设备时妥善保存
```

更新密钥 更新密钥可以参考代码如下：

```
String deviceName = "device_name"; // 设置需要更新的设备名称  
  
DeviceAccessDetailResponse response = client.updateDeviceSecretKey(deviceName);  
  
String tcpEndpoint = response.getTcpEndpoint(); // tcp endpoint地址  
String sslEndpoint = response.getSslEndpoint(); // ssl endpoint地址  
String username = response.getUsername(); // 账户名，用于接入IoT hub服务  
String key = response.getKey(); // 更新密钥后会导致原有的连接断开，需  
要用新的密钥连接。
```

更新设备属性 更新设备属性可以参考代码如下：

```
String deviceName = "device_name"; // 设置需要更新的设备名称  
  
// 设置需要更新的服务端属性  
ObjectNode attributes = new ObjectMapper().createObjectNode();  
attributes.put("regionTag", "Shanghai");  
  
// 设置需要更新的设备影子属性期望值  
ObjectNode desired = new ObjectMapper().createObjectNode();
```

```
desired.put("light", "red");

// 设置需要更新的设备端属性
DeviceAttributes device = new DeviceAttributes()
    .withDesired(desired);

UpdateDeviceProfileRequest request = new UpdateDeviceProfileRequest()
    .withAttributes(attributes)
    .withDevice(device);

DeviceProfileResponse response = client.updateDeviceProfile(deviceName, request);
```

更新设备View 更新设备View可以参考代码如下：

```
String deviceName = "device_name";      // 设置需要更新的设备名称

// 设置需要更新的设备端属性当前汇报值
ObjectNode reported = new ObjectMapper().createObjectNode();
reported.put("light", "red");

// 设置需要更新的设备端属性期望值
ObjectNode desired = new ObjectMapper().createObjectNode();
desired.put("light", "red");

UpdateDeviceViewRequest request = new UpdateDeviceViewRequest()
    .withReported(reported)
    .withDesired(desired);

DeviceViewResponse response = client.updateDeviceView(deviceName, request);
```

更新设备注册表信息 更新设备注册表信息可以参考代码如下：

```
String deviceName = "device_name";      // 设置需要更新的设备名称

String schemaId = "new_schema_id";        // 设置变更的模板ID
String description = "new_description";    // 设置变更的设备描述
boolean favourite = true;                 // 设置变更的收藏选项

UpdateDeviceRegistryRequest request = new UpdateDeviceRegistryRequest()
    .withDescription(description)
    .withSchemaId(schemaId)
    .withFavourite(favourite);
```

```
DeviceProfileResponse response = client.updateDeviceRegistry(deviceName, request);
```

重置设备影子 重置设备影子可以参考代码如下：

```
List<String> devices = Arrays.asList("device_name_1", "device_name_2"); // 设置  
需要重置的设备列表  
  
DeviceListRequest request = new DeviceListRequest().  
    withDevices(devices);  
  
DeviceListResponse response = client.resetDevices(request);
```

8.1.5 模板管理

创建模板 创建模板可以参考代码如下：

```
String schemaName = "schema_name"; // 设置创建的模板名字  
String description = "description"; // 设置创建的模板描述  
  
String propertyName = "property_name"; // 设置  
属性名称  
String displayName = "display_name"; // 设置属性  
显示名称  
String unit = "unit"; // 设置属性单位  
String defaultValue = "default_value"; // 设置  
属性默认值  
SchemaProperty.PropertyType type = SchemaProperty.PropertyType.STRING; // 设置  
属性类型，支持数字/布尔值/字符串  
// 设置模板属性  
SchemaProperty property = new SchemaProperty()  
    .withName(propertyName)  
    .withDisplayName(DISPLAY_NAME)  
    .withUnit(UNIT)  
    .withDefaultValue(defaultValue)  
    .withType(type);  
  
// 设置模板属性列表  
List<SchemaProperty> properties = Arrays.asList(property);  
SchemaCreateRequest request = new SchemaCreateRequest()  
    .withName(schemaName)
```

```
.withDescription(description)  
.withProperties(properties);  
  
SchemaCreateResponse response = client.createSchema(request);  
  
String schemaId = response.getSchemaId(); // 创建的模板ID
```

删除模板 删除模板可以参考代码如下：

```
String schemaId = "schema_id"; // 设置需要删除的模板ID  
  
client.deleteSchema(schemaId);
```

更新模板 更新模板可以参考代码如下：

```
String schemaId = "schemaId"; // 设置需要更新的模板ID  
  
String description = "new_description" // 设置变更的模板描述  
  
String propertyName = "new_property_name"; // 设置变更的属性名称  
String displayName = "new_display_name"; // 设置变更的属性显示名称  
String unit = "new_unit"; // 设置变更的属性单位  
String defaultValue = "new_default_value"; // 设置变更的属性默认值  
SchemaProperty.PropertyType type = SchemaProperty.PropertyType.STRING; // 设置变更的属性类型，支持数字/布尔值/字符串  
// 设置变更的模板属性  
SchemaProperty property = new SchemaProperty()  
    .withName(propertyName)  
    .withDisplayName(DISPLAY_NAME)  
    .withUnit(UNIT)  
    .withDefaultValue(defaultValue)  
    .withType(type);  
  
// 设置变更的模板属性列表  
List<SchemaProperty> properties = Arrays.asList(property);  
  
SchemaUpdateRequest request = new SchemaUpdateRequest()  
    .withDescription(description)
```

```
.withProperties(properties);

client.updateSchema(schemaId, request);
```

获取模板详情 获取模板详情可以参考代码如下：

```
String schemaId = "schema_id";      // 需要查询的模板ID

SchemaResponse response = client.getSchema(schemaId);

String id = response.getId();          // 模板ID
String name = response.getName();       // 模板的名称
String description = response.getDescription(); // 模板的描述
long createTime = response.getCreateTime(); // 模板的创建时间
long lastUpdatedTime = response.getLastUpdatedTime(); // 模板最后一次更新时间
List<SchemaProperty> properties = response.getProperties(); // 模板的属性列表
```

获取及查询模板列表 获取及查询模板列表可以参考代码如下：

```
int pageNo = 1;                      // 设置需要获取所有查询结果的第几页
int pageSize = 10;                    // 设置每页返回的最大个数
String order = "desc";                // 设置按照升序、降序排列结果，支持asc/desc
String orderBy = "createTime";        // 设置排序的索引列，支持name createTime/
lastUpdatedTime
String key = "schema_name";           // 查询关键字，支持模板名称/模板描述

SchemaListResponse response = client.getschemas(pageNo, pageSize, orderBy, order, key);

int totalCount = response.getTotalCount(); // 满足查询条件的模版数目
int pageNo = response.getPageNo();        // 当前页页码
int pageSize = response.getPageSize();     // 返回的每页的最大个数
List<Schema> result = response.getResult(); // 当前页模板详情列表
```

8.1.6 版本说明

v0.10.21 首次发布

8.2 旧版

8.2.1 概述

本文档主要介绍IoT Device SDK的安装和使用。在使用本文档前，您需要先了解IoT Device的一些基本知识，并已经开通了Iot Device服务。若您还不了解Iot Device，可以参考[产品描述](#)和[操作指南](#)。

8.2.2 安装SDK工具包

运行环境

Java SDK工具包可在jdk1.6、jdk1.7、jdk1.8环境下运行。

安装步骤

1. 在[官方网站](#)下载Java SDK压缩工具包。
2. 将下载的**bce-java-sdk-version.zip**解压后，复制到工程文件夹中。
3. 在Eclipse右键“工程->Properties->Java Build Path->Add JARs”。
4. 添加SDK工具包**lib/bce-java-sdk-version.jar**和第三方依赖工具包**third-party/*.jar**。
其中，**version**为版本号。

SDK目录结构

```
com.baidubce
    ├── auth                                //BCE签名相关类
    ├── http                                 //BCE的Http通信相关类
    ├── internal                            //SDK内部类
    ├── model                               //BCE公用model类
    ├── services
        |   └── iotdm                         //物管理服务相关类
        |       └── model                     //物管理内部model，如Request
或Response
    |   └── IotDmClient.class                //物管理客户端入口类
    ├── util                                //BCE公用工具类
    ├── BceClientConfiguration.class         //对BCE的HttpClient的配置
    ├── BceClientException.class            //BCE客户端的异常类
    ├── BceServiceException.class           //与BCE服务端交互后的异常类
    ├── ErrorCode.class                      //BCE通用的错误码
    └── Region.class                        //BCE提供服务的区域
```

8.2.3 创建IotDmClient

用户可以参考如下代码创建一个IotDmClient:

HTTP Client

```
String ACCESS_KEY_ID = <your-access-key-id>; // 用户的Access Key ID
String SECRET_ACCESS_KEY = <your-secret-access-key>; // 用户的Secret Access Key
String ENDPOINT = "iotdm.gz.baidubce.com";

// 创建配置
BceClientConfiguration config = new BceClientConfiguration()
.withCredentials(new DefaultBceCredentials(ACCESSKEY, SECRETKEY))
.withEndpoint(ENDPOINT);

// 初始化一个IotDmClient
IotDmClient client = new IotDmClient(config);
```

HTTPS Client

```
String ACCESS_KEY_ID = <your-access-key-id>; // 用户的Access Key ID
String SECRET_ACCESS_KEY = <your-secret-access-key>; // 用户的Secret Access Key
String ENDPOINT = "iotdm.gz.baidubce.com";

// 创建配置
BceClientConfiguration config = new BceClientConfiguration()
.withProtocol(Protocol.HTTPS) // 使用HTTPS协议
.withCredentials(new DefaultBceCredentials(ACCESSKEY, SECRETKEY))
.withEndpoint(ENDPOINT);

// 初始化一个IotDmClient
IotDmClient client = new IotDmClient(config);
```

在代码中，变量ACCESS_KEY_ID与SECRET_ACCESS_KEY是系统分配给用户的，均为字符串，用于标识用户，为访问物管理做签名认证。其中ACCESS_KEY_ID对应控制台中的“Access Key ID”，SECRET_ACCESS_KEY对应控制台的“Access Key Secret”，获取方式请参考[获取AK/SK](#)。

参数说明

BceClientConfiguration中有更多的配置项，可配置如下参数：

参数	说明
connectionTimeo utInMillis	建立连接的超时时间 (单位: 毫秒)
localAddress	本地地址
maxConnections	允许打开的最大HTTP连接数
protocol	连接协议类型
proxyDomain	访问NTLM验证的代理服务器的Windows域名
proxyHost	代理服务器主机地址
proxyPassword	代理服务器验证的密码
proxyPort	代理服务器端口
proxyPreemptive AuthenticationEnabled	是否设置用户代理认证
proxyUsername	代理服务器验证的用户名
proxyWorkstation	NTLM代理服务器的Windows工作站名称
retryPolicy	连接重试策略
socketBufferSizeInBytes	Socket缓冲区大小
socketTimeoutIn Millis	通过打开的连接传输数据的超时时间 (单位: 毫秒)
userAgent	用户代理, 指HTTP的User-Agent头

8.2.4 设备列表管理

创建设备 用户可以参考如下代码创建设备：

```
CreateDevicesRequest request = new CreateDevicesRequest()
.withAmount(1) // 需要创建的设备数量
.withBootstrap(false) // 是否需要物接入中创建对应的thing以及接入身份

// uuid为一个32位随机UUID, 标识一个请求, 用于服务端在异常情况下重
CreateDevicesResponse response = client.createDevices(request, UUID.randomUUID().toString());

// 获取创建的设备数量
int amount = response.getAmount();
// 获取设备名称列表
List<String> devices = response.getDevices();
// 获取在物接入中创建的thing和身份列表 (需要将请求中bootstrap参数设为true)
List<ThingDetail> things = response.getThings();
```

删除设备 用户可以参考如下代码删除设备：

```
List<String> devices = Arrays.asList("device1", "device2");
RemoveDeviceRequest request = new RemoveDeviceRequest()
.withDeviceOperation(new DeviceOperation().withDevices(devices))
.withCleanThing(false); // 是否需要删除物接入中对应的thing

client.removeDevices(request);
```

获取设备相关信息 用户可以参考如下代码获取设备相关信息：

```
DeviceProfileResponse response = client.getDeviceProfile("mydevice");
// 获取设备id
String id = response.getId();
// 获取设备名称
String name = response.getName();
// 获取设备所在设备组ID
String parent = response.getParent();
// 获取设备描述
String description = response.getDescription();
// 获取设备状态
String state = response.getState();
// 获取设备在服务端的属性
JsonNode attributes = response.getAttributes();
// 获取设备在设备端的属性
DeviceAttributes deviceAttributes = response.getDevices();
```

获取设备信息列表 用户可以参考如下代码获取设备信息列表：

```
DeviceQueryResponse response = client.getDeviceProfiles(
New DeviceQueryRequest().withCondition("{}"));

// 获取设备信息列表
List<DeviceProfile> profiles = response.getDeviceProfiles();
```

获取设备接入相关信息 用户可以参考如下代码获取设备接入topic以及endpoint相关信息：

```
DeviceAccessDetail detail = client.getDeviceAccessDetail("mydevice");

// 获取通信协议，当前为mqtt
String protocol = detail.getProtocol();
// 获取设备接入的IoT Hub endpoint
List<String> endpoints = detail.getEndpoints();
// 获取设备pub到物管理服务的topic信息
List<String> pubTopics = detail.getPubTopics();
// 获取设备sub物管理服务的topic信息
List<String> subTopics = detail.getSubTopics();
```

更新设备Profile 用户可以参考如下代码更新设备Profile：

```
ObjectNode attributes = new ObjectMapper().createObjectNode();
attributes.put("user tag", "student");

UpdateDeviceProfileRequest request = new UpdateDeviceProfileRequest()
.withAttributes(attributes)
.withDeviceOperation(new DeviceOperation.withDevices(Arrays.asList("mydevice")));
client.updateDeviceProfile(request);
```

更新设备注册表信息 用户可以参考如下代码更新设备注册表信息：

```
UpdateDeviceRegistryRequest request = new UpdateDeviceRegistryRequest()
.withDescription("test my device")
.withDeviceOperation(new DeviceOperation().withDevices(Arrays.asList("mydevice")));
client.updateDeviceRegistry(request);
```

8.2.5 设备操作

禁用设备 用户可以参考如下代码禁用设备：

```
DeviceOperationRequest request = new DeviceOperationRequest()
.withDevices(Arrays.asList("mydevice"));
client.disableDevices(request);
```

恢复设备 用户可以参考如下代码恢复设备：

```
DeviceOperationRequest request = new DeviceOperationRequest()  
.withDevices(Arrays.asList("mydevice"));  
client.enableDevices(request);
```

重启设备 用户可以参考如下代码重启设备：

```
DeviceOperationRequest request = new DeviceOperationRequest()  
.withDevices(Arrays.asList("mydevice"));  
client.rebootDevices(request);
```

8.2.6 设备组列表管理

创建设备组

用户可以参考如下代码创建设备组：

```
CreateGroupRequest request = new CreateGroupRequest().withUri("China-beijing");  
CreateGroupResponse response = client.createGroup(request);  
  
// 获取设备组id  
String id = response.getId();  
// 获取设备组名称  
String name = response.getName();  
// 获取设备组uri  
String uri = response.getUri();  
// 获取该组设备数量  
int amount = response.getDevices();  
// 获取父设备组id  
String parent = response.getParent();  
// 获取设备组描述信息  
String description = response.getDescription();  
// 获取设备创建时间  
Date time = getCreateTime();
```

删除设备组

```
client.removeGroup("groupId"); // 通过设备组ID删除设备组
```

获取设备组

```
GroupInfoResponse response = client.getGroup("groupId");
```

获取子设备组

```
GroupListResponse response = client.getClientGroups("groupId");
```

```
// 获取子设备组列表
```

```
List<DeviceGroup> groups = response.getGroups();
```

获取根设备组

```
GroupListResponse response = client.getRootGroups();
```

获取拥有设备的设备组

```
GroupListResponse response = client.getDeviceGroup();
```

8.2.7 版本说明

v0.10.13 首次发布。

第9章

常见问题

9.1 物管理常见问题

9.1.1 物管理中是否能够批量创建设备？

物管理中可以通过调用open API来批量创建设备，查看<https://cloud.baidu.com/doc/IOT-DM/API文档>。

9.1.2 物管理与物接入的关系是什么？

物管理中对设备的反控、更新设备影子等功能是通过物接入来实现的。当用户使用物管理创建设备时，会在物接入中创建一个供物管理使用的实例，以32位数字和字母命名。系统会为每个物管理设备分配主题，用户需在该实例下创建Thing和身份来绑定某个物管理设备的Topic，即可实现对设备的操作。

9.1.3 编辑设备影子中，reported字段和desired字段代表什么意思？

reported字段代表设备向物管理云端汇报的信息；desired字段代表控制台向物管理云端发送的信息，物管理将这些信息发送给设备端。

9.1.4 编辑设备影子中，profileVersion是什么意思？

用于版本控制，每次更新设备影子时的版本号要大于上一个版本号。

9.1.5 设备上传到云端的字段会覆盖设备影子吗？

设备上传到云端的字段，在设备影子中只会更新该字段的值，设备影子中没有被更新的字段仍保留原值；从控制端传到云端的字段，物管理认为这是控制端行为，会覆盖设备影子的原有字段。

9.1.6 物管理服务中的设备与物接入服务中的设备是否一一对应？

否。物管理中的设备与用户的实体设备一一对应，在物管理中的创建设备相当于在云端为实体设备创建一个镜像。

物接入中一个设备可以对应多个实体设备，所有实体设备共享相同的接入用户名和密码，并具有相同的主题收发权限。

9.1.7 物管理服务可管理多少个实体设备？

最多可管理10000台。用户可以通过提交工单申请更高额度。

9.1.8 设备影子中“lastActiveTime”字段是什么意思？

lastActiveTime为最后一次活跃时间，记录设备最近一次和云端交互的时间。若一个设备从未与云端交互，则该时间默认为格林威治时间（1970年1月1日）。