
IOTDM 文档

2018-03-22



百度云
cloud.baidu.com

目录

1 产品介绍	1
1.1 产品概述	1
1.2 产品功能	1
1.3 产品优势	2
1.4 名词解释	2
2 产品定价	4
3 快速入门	5
3.1 下载天工IoT Edge SDK	5
3.2 快速操作	5
3.2.1 1.在物管理控制台或API创建物模型和物影子，系统默认提供水泵的物模型。	5
3.2.2 2.创建物影子后，可以在物影子->物详情中找到『连接配置』的按钮。可以得到设备的接入地址。	6
3.2.3 3.将上面得到的连接配置信息，更新到IoT Edge SDK提供的sample代码中。	6
3.2.4 4.运行设备端程序，即可在云端收到相应的值。	7
3.2.5 5.数据存储到TSDB：在物详情的存储配置，可以选择上报即存储、满足某条件存储到TSDB。	9
3.2.6 6.当使用IoT Edge SDK发送数据时，到TSDB实例中可以看到数据已经写入了TSDB。此场景默认的用法是，设备通过IoT Edge SDK以一定频率向物影子发送数据，然后将数据写入TSDB。所以，在控制台上更新物影子的操作无法将数据写入TSDB。	9
3.2.7 7.在物可视中通过可视化的方式查看。	9

4 操作指南	10
4.1 在云端创建物影子	10
4.1.1 物模型	10
创建物模型	10
管理物模型	12
4.1.2 物影子	13
创建物影子	13
查看影子详情	14
获取连接配置	16
4.2 物影子操作	17
4.2.1 使用Baidu IoT Edge SDK	17
4.2.2 使用MQTT客户端SDK	17
4.2.3 更新设备状态到设备影子	17
4.2.4 从设备影子获取设备状态	20
4.2.5 通过设备影子控制设备状态	21
4.2.6 删除设备影子	21
4.2.7 订阅设备影子的变化	22
4.2.8 订阅设备快照	23
4.2.9 Device Profile	23
4.3 数据存储TSDB	24
4.3.1 开启存储配置	24
4.3.2 存储TSDB格式	25
4.4 权限管理	25
4.4.1 介绍	25
4.4.2 创建权限	26
4.4.3 创建超级权限	28
4.5 设备在线状态	29
4.6 设备相互通信	30

4.7 MQTT开源SDK接入指南	30
4.8 物管理的数据消费	30
5 应用场景	31
5.1 端到端配置示例	31
5.1.1 操作步骤	31
设备接入云端	31
设备向云端上报状态	35
通过设备影子控制设备状态	45
可视化展现物影子	47
设备在线状态的说明	51
5.2 基于物管理快速搭建物联网应用	53
5.2.1 简介	53
5.2.2 1.云端创建设备影子	54
5.2.3 2.设备端发消息更新云端的物影子	54
5.2.4 3.设备端与云端通信(非更新物影子)	55
5.2.5 4.应用服务和手机APP端消费设备的数据	55
6 API参考 - V3	57
6.1 目录	57
6.2 介绍	58
6.2.1 简介	58
6.2.2 多区域选择	58
6.3 设备列表管理	59
6.3.1 创建单个设备	59
6.3.2 删除设备	60
6.3.3 获取设备Profile	61
6.3.4 获取设备View	62
6.3.5 获取及查询影子列表	63

6.3.6	获取设备接入详情	65
6.3.7	更新密钥	66
6.3.8	更新设备属性	67
6.3.9	更新单个设备View信息	69
6.3.10	更新单个设备注册表信息	70
6.3.11	重置设备影子	72
6.4	设备模板管理	73
6.4.1	创建模板	73
6.4.2	删除模板	74
6.4.3	获取模板列表	75
6.4.4	获取模板	77
6.4.5	更新模板	78
6.5	物管理数据写入TSDB	79
6.5.1	创建规则	79
6.5.2	获取规则详情	82
6.5.3	修改规则	84
6.5.4	删除规则	88
6.5.5	禁用一条规则	88
6.5.6	启用一条规则	89
6.5.7	参数定义	89
	DeviceRuleRequest	90
	DeviceRuleSource	90
	DeviceRuleDestination	91
	DeviceRuleResponse	92
	DeviceRuleSourceDetail	93
	DeviceRuleDestinationDetail	94
6.6	权限管理	94
6.6.1	简介	94

6.6.2	设备权限组管理	94
	创建权限组	94
	删除权限组	95
	获取权限组列表	96
	获取权限组详情	98
	权限组中更改设备	99
	更新权限组注册信息	100
	获取权限组接入信息	101
	更新权限组密钥	102
	获取及查询设备列表	103
6.6.3	参数定义	105
6.7	参数定义	106
6.7.1	DeviceList参数列表	106
6.7.2	DeviceProfile参数列表	106
6.7.3	DeviceBasicInfo参数列表	106
6.7.4	DeviceAttributes参数列表	106
6.7.5	DeviceView参数列表	107
6.7.6	DeviceViewAttributes参数列表	107
6.7.7	DeviceAccessDetail参数列表	107
	SchemaProperty参数列表	108
	Schema参数列表	109
6.8	更新历史	109
6.9	API参考-V1	109
6.10	API参考-V2	109
7	Java SDK文档	110
7.1	V3	110
7.1.1	概述	110

7.1.2	安装SDK工具包	110
7.1.3	创建IotDmV3Client	111
7.1.4	设备管理	113
创建设备	113	
删除设备	113	
获取设备Profile	113	
获取设备View	114	
获取及查询影子列表	114	
获取设备接入详情	115	
更新密钥	115	
更新设备属性	115	
更新设备View	116	
更新设备注册表信息	116	
重置设备影子	117	
7.1.5	模板管理	117
创建模板	117	
删除模板	118	
更新模板	118	
获取模板详情	119	
获取及查询模板列表	119	
7.1.6	版本说明	119
v0.10.21	119	
7.2	旧版	120
7.2.1	概述	120
7.2.2	安装SDK工具包	120
7.2.3	创建IotDmClient	121
7.2.4	设备列表管理	122
创建设备	122	

删除设备	123
获取设备相关信息	123
获取设备信息列表	123
获取设备接入相关信息	123
更新设备Profile	124
更新设备注册表信息	124
7.2.5 设备操作	124
禁用设备	124
恢复设备	124
重启设备	125
7.2.6 设备组列表管理	125
7.2.7 版本说明	126
v0.10.13	126
8 IoT Edge SDK	127
9 常见问题	128
9.1 物管理常见问题	128
9.1.1 物管理中是否能够批量创建设备 ?	128
9.1.2 物管理服务可管理多少个实体设备 ?	128
9.1.3 新版物管理提供了哪些topic?	128
9.1.4 可以使用MQTT的SDK连接物管理吗?	128
9.1.5 物管理如何获得设备在线状态?	128
9.1.6 物接入和物管理如何选择?	129
9.1.7 物影子有什么作用?	129
9.1.8 编辑设备影子中, reported字段和desired字段代表什么意思 ?	129
9.1.9 编辑设备影子中, profileVersion是什么意思 ?	129
9.1.10 设备上传到云端的字段会覆盖设备影子吗 ?	129
9.1.11 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 ?	129

第1章 产品介绍

1.1 产品概述

物联网时代，万物互联已经是势不可挡的趋势，世界每一个物都是可以互联并产生价值的。

物管理旨在云端构建一个真实的物理世界，实现在云端的设备管理：包括设备注册、设备鉴权、设备实时状态存储、设备影子、设备状态管理等。适用于工业联网设备、智能家居、车联网等场景。

物管理可以无缝连接天工平台和百度云的各类服务，如时序数据库、规则引擎、物可视等。助力构建完整的物联网服务。

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。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](#)。

1.2 产品功能

设备管理

在云端实现设备注册管理、设备接入、设备鉴权、设备实时状态查询、设备反控、设备检索、OTA等。

物影子

接受设备上报的消息，在云端保存设备的最新状态；并且可以通过物影子实现反控。

物模型

通过构建单个设备模型，实现与时序数据库等天工产品无缝对接；提供楼宇等多个场景的模型构建工具。

端SDK

提供端上SDK，不需要理解接入协议，建立端与云的双向连接，实现物影子的状态更新、反控、OTA等。

1.3 产品优势

易管理

覆盖设备全生命周期管理，实时监控设备在线状态。

易连接

仅需三步即可将设备连接到云端：云端创建设备、下载连接地址和密钥，设备端安装SDK并配置连接地址即可完成数据上传。

开放兼容

接入协议支持开源MQTT和HTTP，适配更多设备连接，更多的协议接入将陆续开放。

1.4 名词解释

物影子

物影子反映物理世界中的一个物，是物在云端的『影子』或『数字双胞胎』。运行时，物将监控值上报给物影子，物影子会用一个json文档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通过MQTT或HTTP访问。

同时，物影子也提供反控功能。

如下提供一个水泵物影子的示例：

```
{  
  "reported": {  
    "FrequencyIn": "70",  
    "Current": "3",  
    "Speed": "1.2",  
    "Torque": "12",  
    "DC_bus_voltage": 89,  
    "light": "red"  
  },  
  "desired": {  
    "light": "red"  
  },  
  "profileVersion": 10,  
  "timestamp": 1499598239827  
}
```

- reported字段，代表设备上报的最新状态。服务端能通过MQTT或HTTP从reported字段中拿到物影子的最新状态。

- **desired**字段，代表控制端期望设备变换到的目标状态。设备端通过MQTT从**desired**字段中拿到某个属性的期望值（如“light”：“red”），就收到了控制端期望执行的操作，硬件即可执行相关操作。硬件执行相关操作后，应该把对应的值上报到**reported**字段上。用户可以通过判断**repoeted**与**desired**的差别来判断是否反控成功。
- **profileVersion**字段，代表物影子的版本号。上传的**profileVersion**字段，只有大于原值，物管理才会接受更新物影子的请求。如果没有该字段，则系统会自动+1。

其他详细操作请参考[操作指南文档](#)。

物模型

物模型由一个或多个属性构成，您可以用他来表示一类设备。有了物模型的定义，可以更方便的基于模型的接口来处理物影子数据，以及反控。也可以通过直接配置模型的属性实现上传的数据转发至TSDB。不需要写复杂的SQL语句。

第2章 产品定价

在物管理推出单独计费策略之前，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[物接入产品定价](#)。

物管理单独计费策略时不会影响使用户使用，请放心使用。

第3章 快速入门

3.1 下载天工IoT Edge SDK

下载天工IoT Edge SDK，IoT Edge SDK包含了物管理的C语言客户端、序列化和反序列化、物影子操作等组件。涵盖了实现设备上云时在断线缓存、数据安全等多种场景，更多的服务会在未来开放。

地址：<https://github.com/baidu/iot-edge-c-sdk>

按照以下路径找到物管理的sample，这里面封装了更新物影子等操作。

[iot-edge-c-sdk / iothub_client / samples / iotdm_client_sample / iotdm_client_sample.c](#)

3.2 快速操作

3.2.1 1. 在物管理控制台或API创建物模型和物影子，系统默认提供水泵的物模型。

物模型卡片列表

请输入模型名称

请点击新建物模型

_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。

物影子卡片列表

按名称查询

请输入名称

111

说明：fc

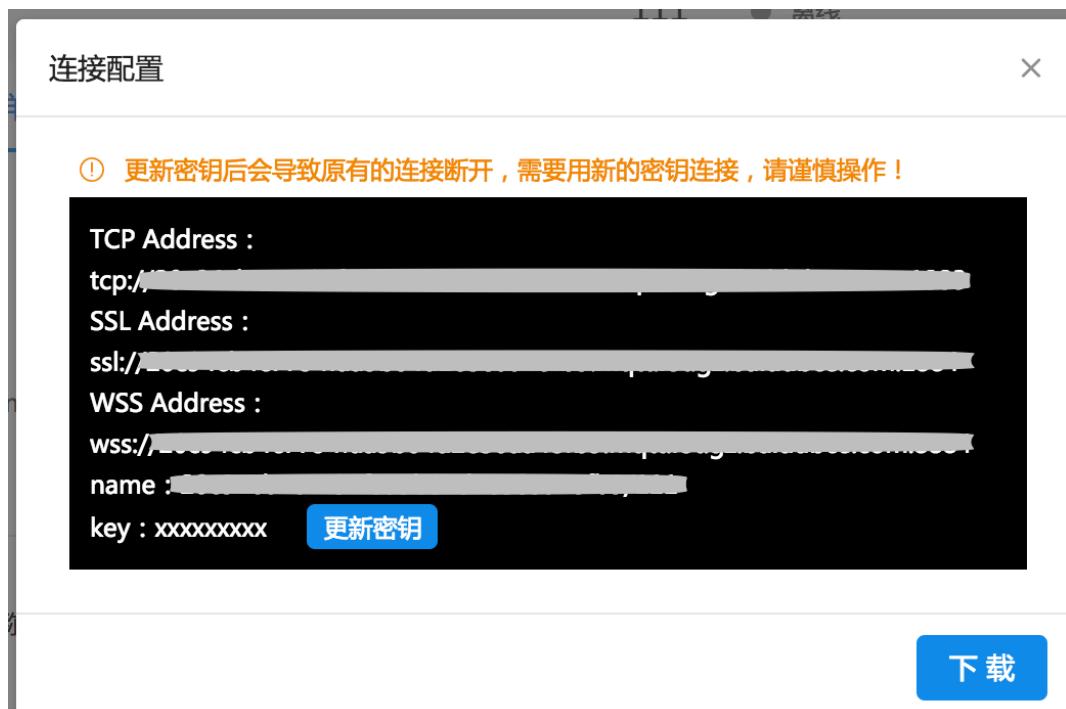
属性总计：0个

AAAAA

说明：

属性总计：0个

3.2.2 2. 创建物影子后，可以在物影子->物详情中找到『连接配置』的按钮。可以得到设备的接入地址。



3.2.3 3. 将上面得到的连接配置信息，更新到 IoT Edge SDK 提供的 sample 代码中。

```
// 2. "ssl://xxxxxx.mqtt.iot.xx.baidubce.com:1884"  
#define ADDRESS "tcp://xxxxxx.mqtt.iot.xx.baidubce.com:1883"  
  
// The device name you created in device management service.  
#define DEVICE "xxxxxx"  
  
// The username you can find on the device connection configuration web,  
// and the format is like "xxxxxx/xxxxx"  
#define USERNAME "xxxxxx/xxxxxx"  
  
// The key (password) you can find on the device connection configuration web.  
#define PASSWORD "xxxxxx"
```

SDK 代码中编写每个属性的上传值，如下参考。

```
// Sample: use 'serializer' to encode device model to binary and update the device shadow.  
BaiduSamplePump* pump = CREATE_MODEL_INSTANCE(BaiduIotDeviceSample, BaiduSamplePump);  
pump->FrequencyIn = 1;  
pump->FrequencyOut = 2;  
pump->Current = 3;  
pump->Speed = 4;  
pump->Torque = 5;  
pump->Power = 6;  
pump->DC_Bus_voltage = 7;  
pump->Output_voltage = 8;  
pump->Drive_temp = 9;
```

3.2.4 4.运行设备端程序，即可在云端收到相应的值。

属性名称	显示名称	类型	默认值	单位	当前值	修改时间
FrequencyIn	输入频率	number	0	Hz	1	2017-12-21 22:34:12
Current	电流	number	0	A	3	2017-12-21 22:34:12
Speed	速度	number	0	rpm	4	2017-12-21 22:34:12
Torque	输出转矩	number	0	%	5	2017-12-21 22:34:12
Power	输出功率	number	0	KW	6	2017-12-21 20:50:53
DC_bus_voltage	直流侧电压	number	0	V	7	2017-12-21 20:50:53
Output_voltage	输出电压	number	0	V	8	2017-12-21 20:50:53
Drive_temp	变频器温度	number	0	C	9	2017-12-21 20:50:53

物影子的原始json数据如下：

模型数据

原始数据

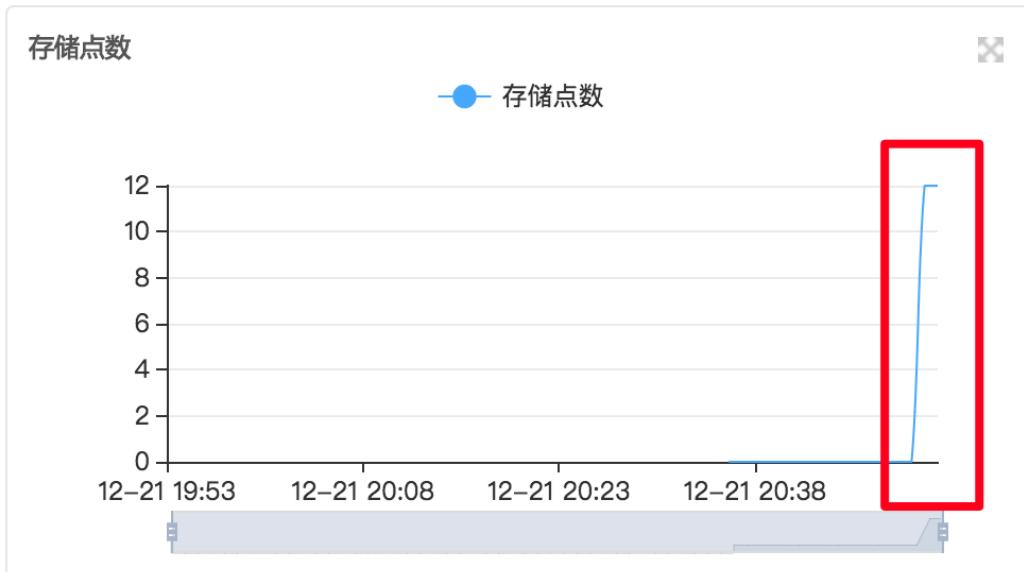
Shadow State :

```
1  {
2      "desired": {
3          "rotate": 100,
4          "FrequencyOut": 23,
5          "light": "red"
6      },
7      "reported": {
8          "memoryFree": "32MB",
9          "light": "red",
10         "FrequencyIn": 1,
11         "Current": 3,
12         "Speed": 4,
13         "Torque": 5,
14         "Power": 6,
15         "DC_bus_voltage": 7,
16         "Output_voltage": 8,
17         "Drive_temp": 9
18     }
19 }
```

3.2.5 5. 数据存储到TSDB：在物详情的存储配置，可以选择上报即存储、满足某条件存储到TSDB。

属性名称	显示名称	类型	默认值	单位	存储配置
FrequencyIn	输入频率	number	0	Hz	<input type="button" value="上报满足..."/> <input type="button" value=">= 12"/>
Current	电流	number	0	A	<input type="button" value="上报即存..."/> <input type="button" value="阀值"/>
Speed	速度	number	0	rpm	<input type="button" value="上报即存..."/> <input type="button" value="阀值"/>
Torque	输出转矩	number	0	%	<input type="button" value="上报即存..."/> <input type="button" value="阀值"/>
Power	输出功率	number	0	KW	<input type="button" value="上报即存..."/> <input type="button" value="阀值"/>

3.2.6 6. 当使用IoT Edge SDK发送数据时，到TSDB实例中可以看到数据已经写入了TSDB。此场景默认的用法是，设备通过IoT Edge SDK以一定频率向物影子发送数据，然后将数据写入TSDB。所以，在控制台上更新物影子的操作无法将数据写入TSDB。



3.2.7 7. 在物可视中通过可视化的方式查看。

参考：[可视化展现物影子](#)

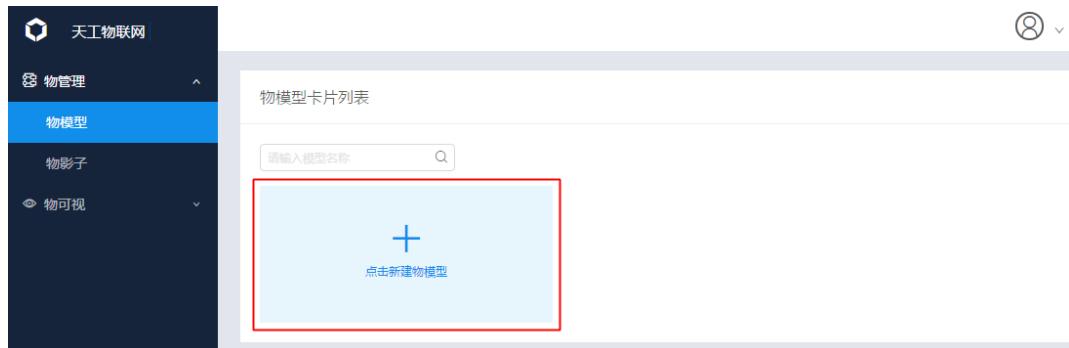
第4章 操作指南

4.1 在云端创建物影子

4.1.1 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。

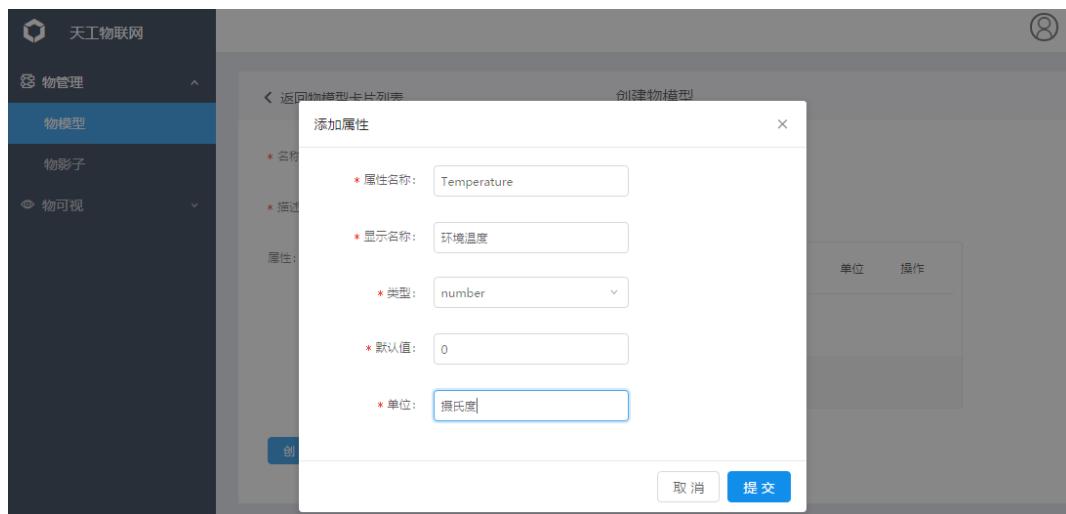


4. 填写物模型配置，包括名称、描述和属性。

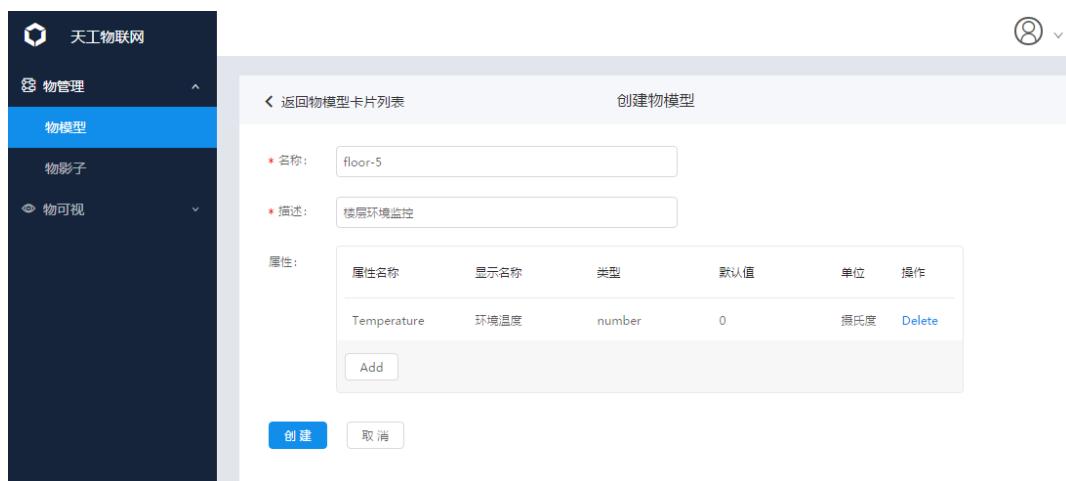
创建成功后物模型名称和属性名称无法修改，同一物模型下属性名称必须唯一。



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。



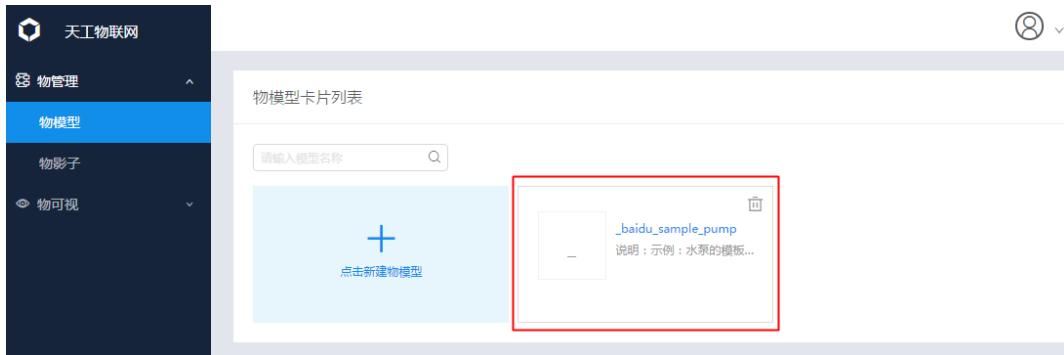


管理物模型 创建物模型以后，可以对以下信息进行编辑修改，包括：

- 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。

属性名称	显示名称	类型	默认值	单位
FrequencyIn	输入频率	number	0	Hz
FrequencyOut	输出频率	number	0	Hz
Current	电流	number	0	A
Speed	速度	number	0	rpm

4.1.2 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、选择物模型和是否开启存储配置。[数据存储TSDB](#)

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了不在该物详情中的属性，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
					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。

The screenshot shows the 'Thing Shadow' management interface. On the left, there's a sidebar with '物管理' (Thing Management) and '物影子' (Thing Shadow) selected. The main area has tabs for '模型数据' (Model Data) and '原始数据' (Raw Data). A table lists four properties: 'Frequen cyIn' (输入频率), 'Frequen cyOut' (输出频率), 'Current' (电流), and 'Speed' (速度). Each row includes columns for '属性名称' (Property Name), '显示名称' (Display Name), '类型' (Type), '默认值' (Default Value), '单位' (Unit), '当前值' (Current Value), '修改时间' (Last Modified Time), '期望值' (Desired Value), and '发送时间' (Send Time). The 'Speed' row's '期望值' column contains '100', which is 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
Frequen cyIn	输入频率	num ber	0	Hz	<input type="text"/>	2017-06-16 11:30:53	<input type="text" value="100"/>	2017-06-16 11:30:53
Frequen cyOut	输出频率	num ber	0	Hz	<input type="text"/>	2017-06-16 11:30:53	<input type="text"/>	2017-06-16 11:30:53
Current	电流	num ber	0	A	<input type="text"/>	2017-06-16 11:30:53	<input type="text"/>	2017-06-16 11:30:53
Speed	速度	num ber	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。

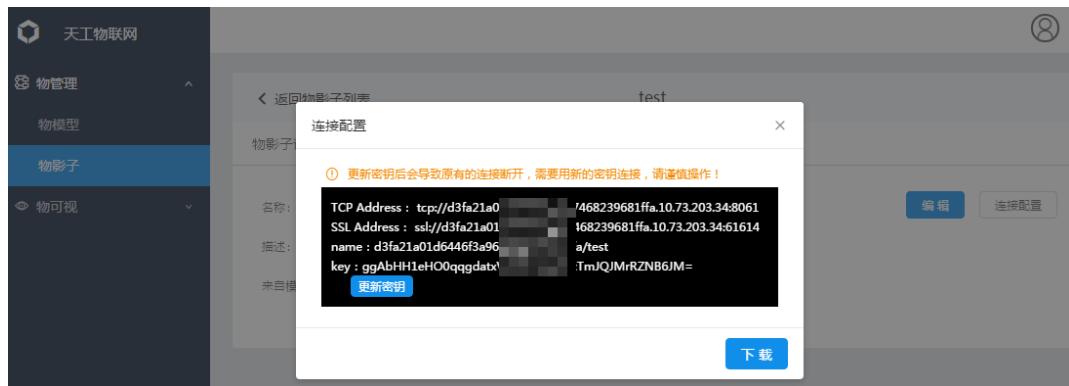
The screenshot shows the 'Thing Detail' interface. It includes fields for '名称' (Name: floor5), '描述' (Description: 环境监测5), '来自模型' (From Model: floor5), and a '存储配置' (Storage Configuration) switch set to 'ON'. Below these are sections for '物影子' (Thing Shadow) and '连接配置' (Connection Configuration). The '连接配置' section is highlighted with a red box. A table shows a single entry for 'Temperature' (环境温度) with type 'number', default value '0', unit '摄氏度', storage configuration '上报且满足条件存储 > 20', and last stored time 'N/A'.

属性名称	显示名称	类型	默认值	单位	存储配置	最后一次存储时间
Temperature	环境温度	number	0	摄氏度	上报且满足条件存储 > 20	N/A

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。



4.2 物影子操作

4.2.1 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>

4.2.2 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。

4.2.3 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,  
"lastUpdatedTime": {  
    "reported": {  
        "light": 1494904250  
    },  
    "desired": {  
        "rotate": 1494904250  
    }  
}  
}
```

- “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，代表物影子中设备上报的最新状态。服务端能通过MQTT或HTTP从reported字段中拿到物影子的最新状态。
- desired为可选字段，代表控制端期望设备变换到的目标状态。设备端通过MQTT从desired字段中拿到某个属性的期望值（如“light”：“red”），就收到了控制端期望执行的操作，硬件即可执行相关操作。硬件执行相关操作后，应该把对应的值上报到reported字段上。用户可以通过判断repoeted与desired的差别来判断是否反控成功。
- “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。
- “lastUpdatedTime” 为可选字段，“lastUpdatedTime.reported” 和 “lastUpdatedTime.desired” 中的时间表示属性（“reported” 和 “desired”）的更新时间，如果没有相应字段，则更新时间由系统时间决定。注意只有当相应位置的属性键值对存在于本次请求中，且请求更新时间为非负整数（毫秒为单位），相应的时间更新有效，无效的更新时间会被替换为系统时间。此外，若本次请求中属性的更新时间早于系统中该属性已存储的更新时间，则该属性的本次更新时间判断为过时，不予更新。

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}"
}
```

4.2.4 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}"
}

```

4.2.5 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}"
}

```

4.2.6 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

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

4.2.7 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"
    }
}
```

4.2.8 订阅设备快照

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
        }
    }
}
```

4.2.9 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/unknown
"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
}
}
```

4.3 数据存储TSDB

4.3.1 开启存储配置

物管理支持将数据存储到时序数据库TSDB。可以在[创建物影子](#)时，开启存储配置。同时，也可以在[物详情获取连接配置](#)中，开启与关闭存储配置。

开启存储配置：

- 可修改存储配置，选择不存储，上报即存储和上报满足条件存储三种方式。
- 可将数据存储到时序数据库中（TSDB），即需要选择相应的TSDB实例。

- 配置生效后，物管理会在收到物影子更新时，将更新的数据按配置条件写入时序数据库，这里需要保证时序数据库无欠费等异常情况。



4.3.2 存储TSDB格式

TSDB中表示每个数据点的格式如下：

TSDB实例	metric	timestamp	value	tag1 (显示名称)	tag2 (物影子名称)	tag3 (物模型名称)
需提前创建	属性名如 temperature	属性的更新时间如 1499071119722	该属性本次更新值	“desc” = “温度”	“deviceName” = “pantone”	“template_name” = “template_pantone”

4.4 权限管理

4.4.1 介绍

在物管理中设置权限管理，可以使得一个客户端的连接配置具有访问多个设备的能力。

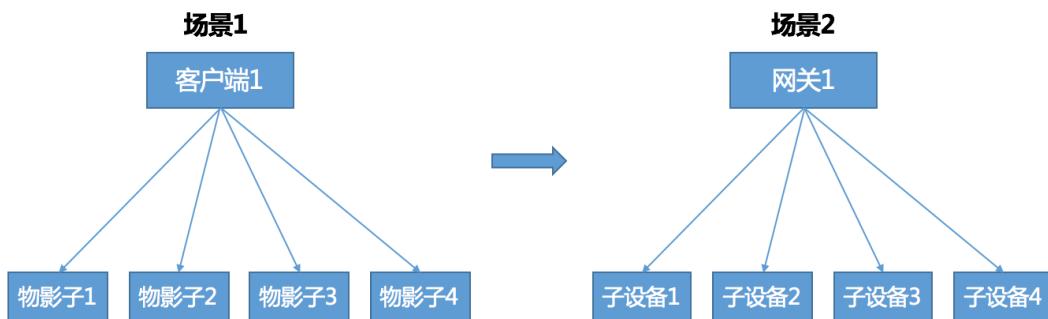
物管理权限分为普通权限和超级权限。

- 普通权限：由用户自行选择添加物影子，上限100个。
- 超级权限：默认包含用户名下所有的物影子。

如图所示：

- 给客户端1新建权限1，权限1内包括物影子1、2、3、4；

2. 权限1的用户名与密钥可以向物影子1、2、3、4的topic发送消息；
3. 根据权限1起一个连接就可以向物影子1、2、3、4对应的设备通信。



注意：

1. 用户通过服务端连接物管理，订阅所有/部分设备的物影子变化状态，可以用一个mqtt客户端订阅，不需要启用n个客户端。
2. 网关设备连接n个子设备，需要有一个MQTT客户端可以对网关和子设备都有访问权限，如更新shadow/获取delta等。设置权限管理后，则可避免一个网关启用n个连接来更新n个子设备的物影子。
3. 一个设备最多允许属于10个权限组。

4.4.2 创建权限

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>高级设置”，接下来进行权限管理设置。
4. 点击“创建权限”，填写基本信息，分别为名称和描述。



5. 选择物影子，支持多选，目前上限100个。

创建普通权限

基本信息 —— 2 添加影子

按影子查询 请输入影子名称

影子名称

<input type="checkbox"/> agaggga
<input type="checkbox"/> teaset
<input type="checkbox"/> 333333
<input type="checkbox"/> aaaaaaaaaaa
<input checked="" type="checkbox"/> floor5
<input type="checkbox"/> czhuahggagagae
<input type="checkbox"/> asdfasdfasdfsad
<input type="checkbox"/> eeeeeeee

已选择 1 / 100 项

6. 点击“确定”，完成权限设置，并生成配置信息。

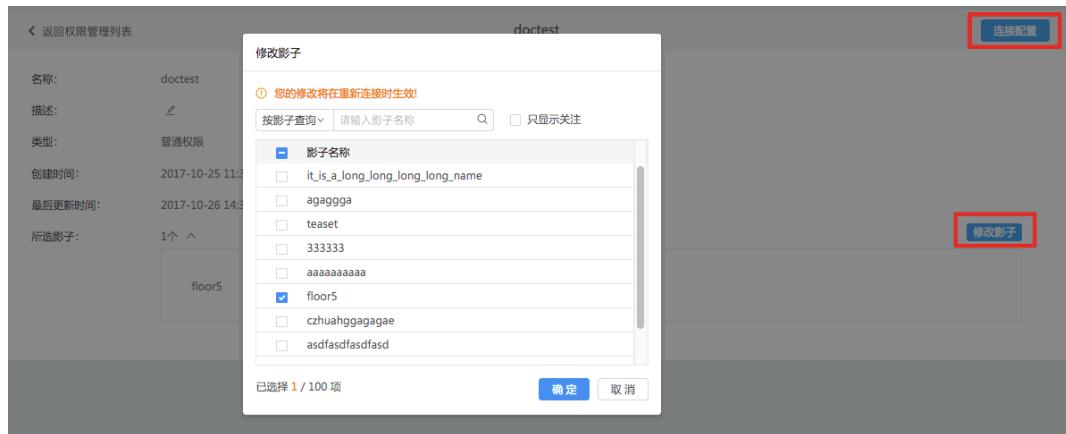
提示

- ✓ 创建成功！请将连接信息配置到SDK中，实现设备与云端连接。[如何连接](#)
为了安全考虑，请合理保管以上密钥，密钥丢失无法找回，只能在物详情中重新生成。

TCP Address :
tcp://001ccb00097061d7fa209660ud59df1cc0.107.51.205.9.101021
SSL Address :
ssl://001ccb00097061d7fa209660ud59df1cc0.107.51.205.9.101021
name : 051ccb00097061d7fa209660ud59df1cc0.107.51.205.9.101021
key : 051ccb00097061d7fa209660ud59df1cc0.107.51.205.9.101021

7. 点击权限名称，进入权限管理列表。

- 可以编辑权限描述；
- 点击“连接配置”，可以查看配置信息。
- 点击“修改影子”，可以重新添加物影子。



4.4.3 创建超级权限

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>高级设置”，接下来进行权限管理设置。
4. 点击“创建超级权限”。



5. 超级权限具有访问所有设备的权限，请妥善保管好用户名和秘钥，谨慎使用。

- 超级权限不支持权限内物影子的增加和减少。
- 在创建超级权限后，新创建的影子会被默认添加到超级权限内。

权限名称	描述	类型	影子总数	操作
doctest1		超级权限	42	删除
doctest		普通权限	1	删除

6. 点击权限名称，进入权限管理列表，可以编辑权限描述；点击“连接配置”，可以查看配置信息。

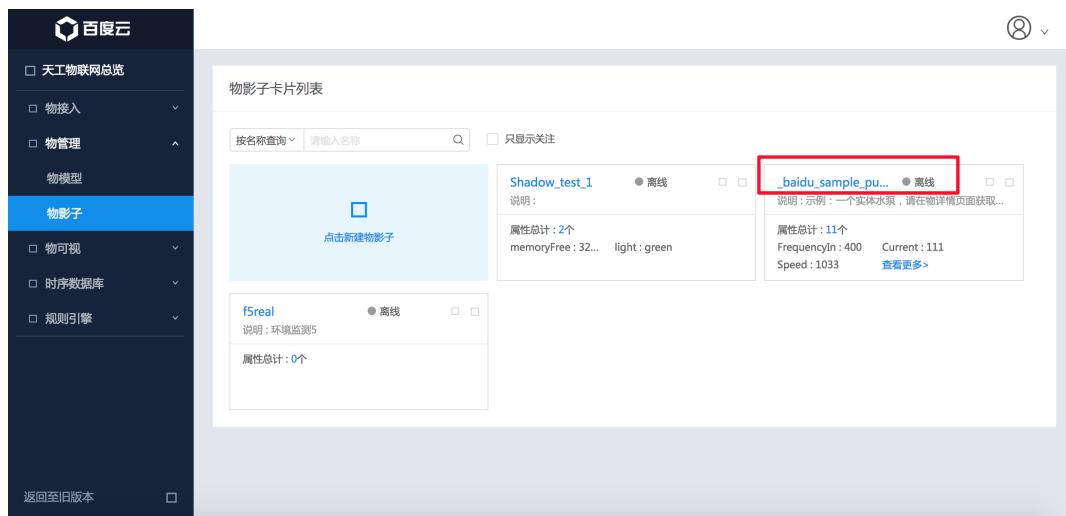


4.5 设备在线状态

注意

使用IoT Edge SDK,当使用SDK向云端发送过一次数据后，物影子即标志为『在线』。如果设备由于网络原因或自身关机等原因断开连接，物影子即显示离线，可以通过API获取。
使用MQTT开源SDK，如果使用MQTT开源SDK，请将MQTT客户端的clientId设置为物影子名称，物管理则可以监测在线和离线状态。

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



2. 使用IoT Edge SDK,当使用SDK向云端发送过一次数据后，物影子即标志为『在线』。
如果设备由于网络原因或自身关机等原因断开连接，物影子即显示离线，可以通过API获取。
3. 使用MQTT开源SDK，如果使用MQTT开源SDK，请将MQTT客户端的clientId设置为物影子名称，物管理则可以监测在线和离线状态。

4.6 设备相互通信

设备可以通过更新物影子来上传当前状态，在有些场景下，设备还需要相互通信。物管理默认给每个设备绑定有主题\$baidu/iot/general/#的订阅和发布权限。

例如，设备A发布消息到主题\$baidu/iot/general/a，设备B订阅主题\$baidu/iot/general/a，则设备B就能收到设备A的消息。

4.7 MQTT开源SDK接入指南

物管理兼容开源MQTT的设计，目前IoT Edge SDK推出了C语言版本，如果IoT Edge SDK无法安装到您的设备上，也可以使用开源MQTT SDK，接入信息从物详情->连接配置中获取，将连接配置中提供的信息填写到MQTT客户端，其中，clientID填写物影子名称。

4.8 物管理的数据消费

1. 通过存储配置到TSDB中，基于TSDB的接口消费数据。
2. 物可视中展示数据。见[物可视中展示数据](#)
3. 服务端消费数据。服务端可以通过API获取设备当前状态，也可以通过MQTT客户端连接到物管理，sub相应的topic，获取数据。相应的topic定义参见：[操作指南->物影子操作](#)。

第5章 应用场景

5.1 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。

5.1.1 操作步骤

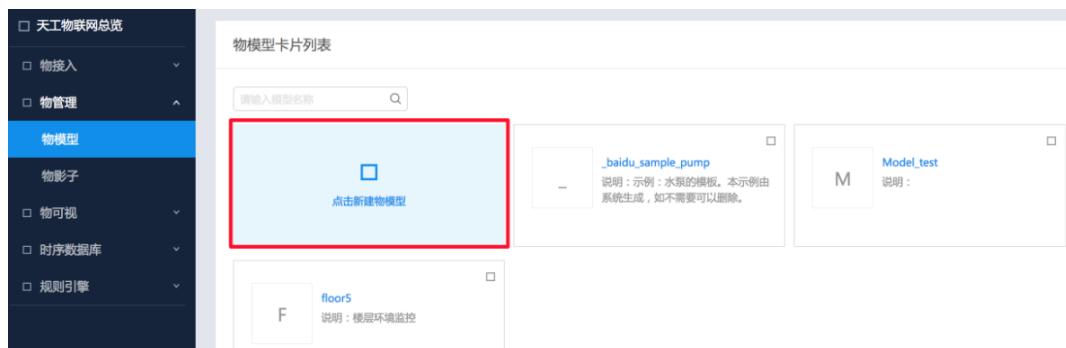
本示例分成以下几步：

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：

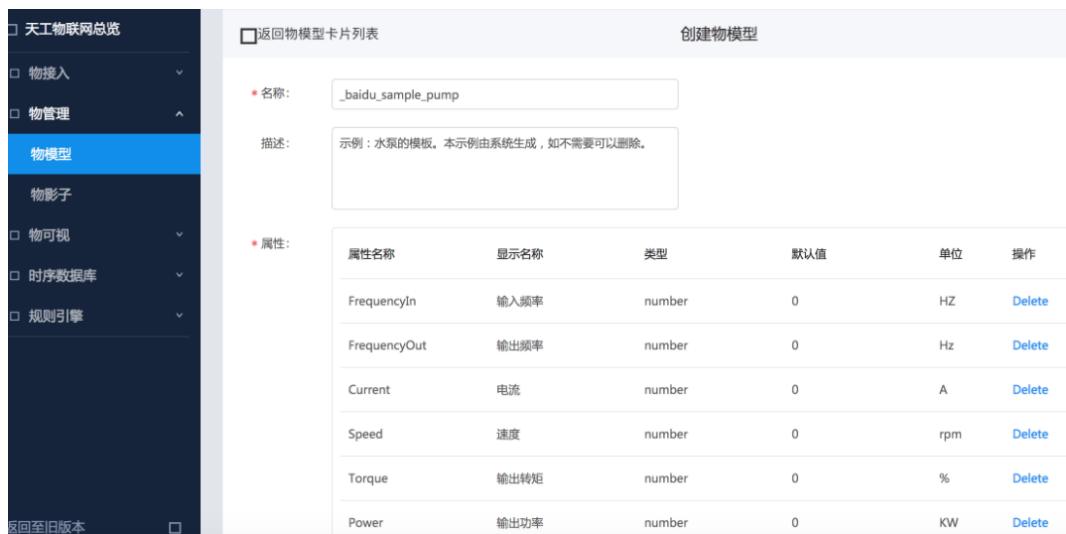
1. 登录百度云官网，点击右上角的“管理控制台”，快速进入控制台界面。在控制台中，选择“产品服务>物管理 IoT Device>物模型”，在物管理中创建物模型。
2. “点击新建物模型”进入物模型配置界面。



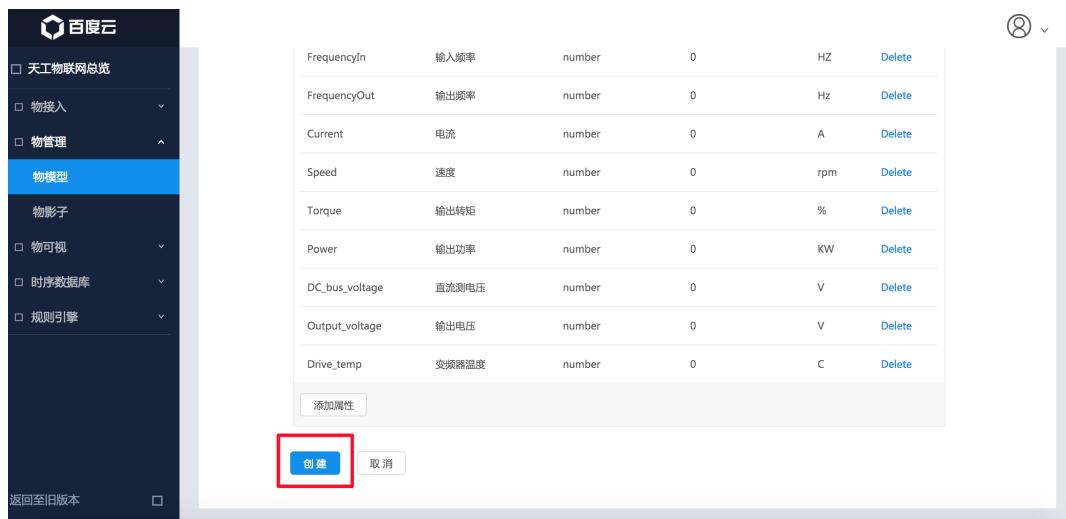


3. 填写物模型配置，包括名称、描述和属性。点击添加属性为物模型新增一条属性。

“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。



- 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。

- 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。
- “点击新建物影子”进入物影子配置界面。

The screenshot shows the 'Thing Shadow Card List' interface. On the left is a sidebar with navigation items like 'Thing Model', 'Thing Shadow' (which is selected and highlighted in blue), and 'Thing View'. The main area displays a list of thing shadows. One item, 'Shadow_test_1', is shown with its status as 'Online' (green dot). Another item, 'f5real', is shown as 'Offline' (grey dot). At the bottom of the list, there's a note: '说明: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。' Below the list, there are summary statistics: '属性总计: 2个', 'memoryFree: 32...', 'light: green', '属性总计: 0个', 'FrequencyIn: 20', 'Current: 111', and 'Speed: 1033'. A large red box surrounds the 'Create New Thing Shadow' button at the top of the list.

- 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。

The screenshot shows the 'Create Thing Shadow' dialog. The left sidebar is identical to the previous screenshot. The main dialog has a title bar with 'Return to Thing Shadow List' and 'Create Thing Shadow'. It contains three input fields: 'Name' (set to '.baidu_sample_pump_instance'), 'Description' (with a placeholder note about connecting to a device pump), and 'Selected Model' (set to '.baidu_sample_pump'). At the bottom are 'Create' and 'Cancel' buttons.

- 点击“创建”完成物影子创建。

- 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”。



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。



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>

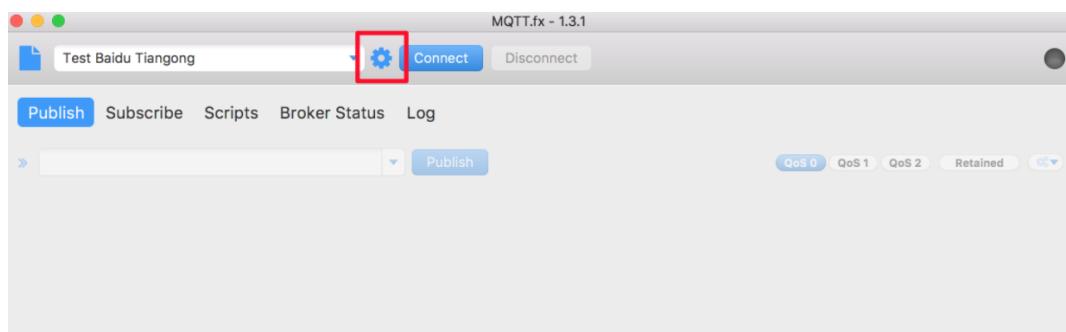
2. 接入方式2：使用MQTT客户端SDK

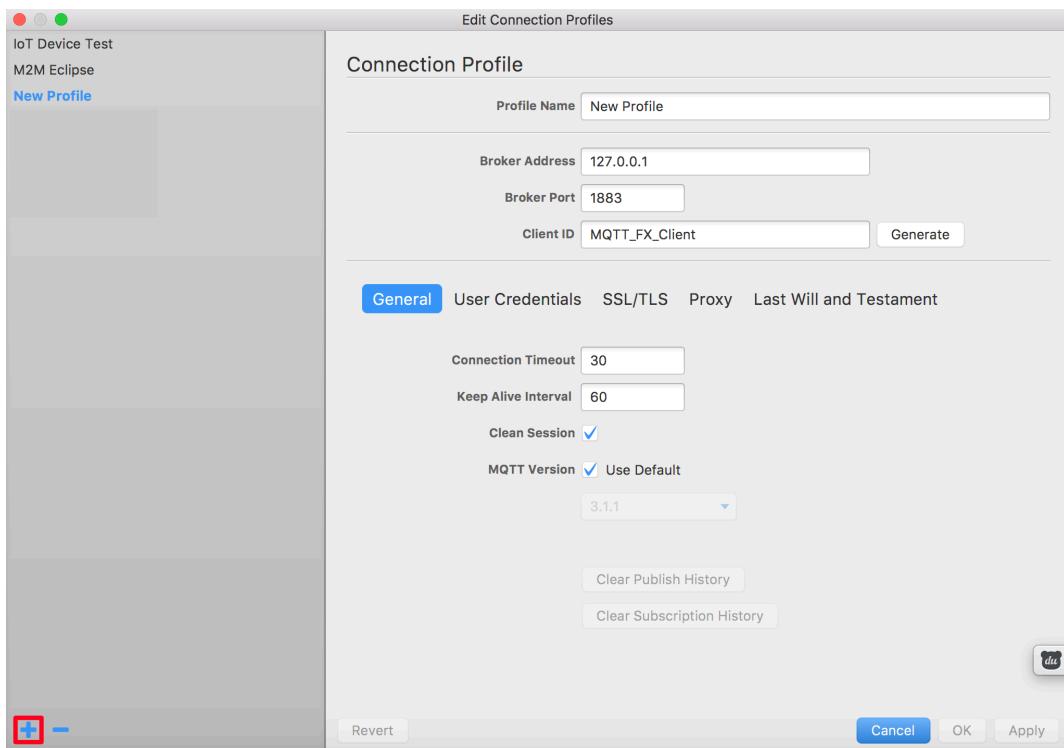
如果设备端已经调用了[Paho \(即MQTT Client SDK\)](#)，可以通过与特定的topic通信实现与百度云的对接。在物管理中定义了系统topic用于物管理服务和设备端基于物接入服务以及MQTT协议进行通信。

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

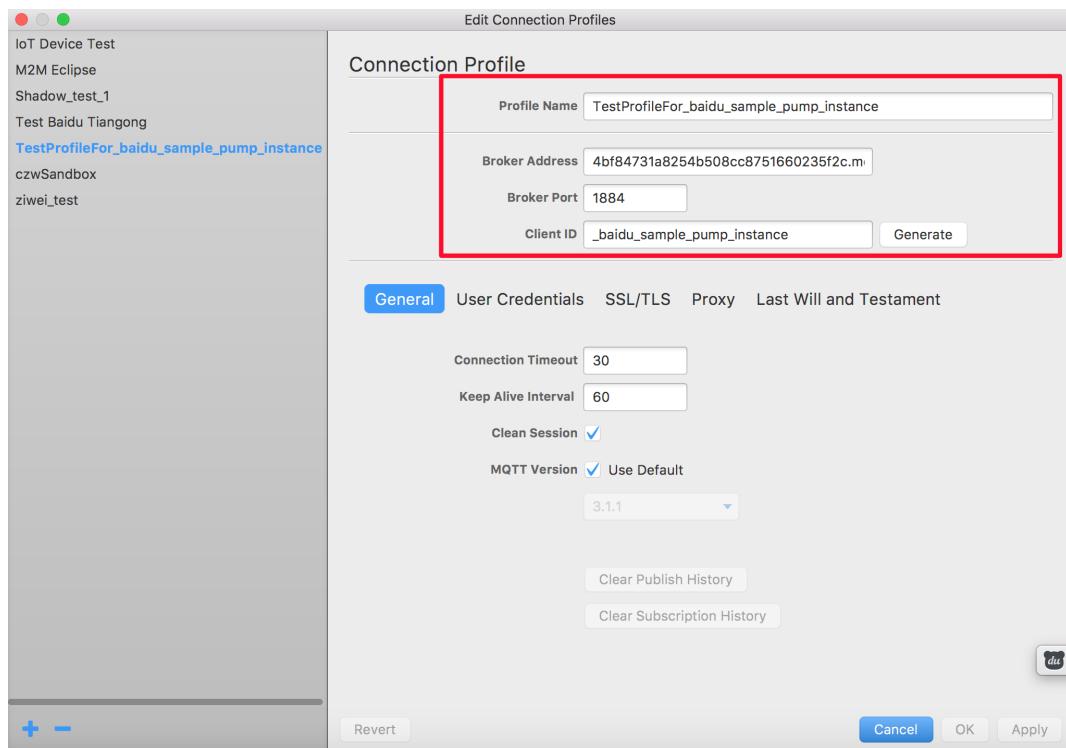
1. 打开MQTT.fx，选择设置项进入设置页面，点击左下角“+”按键，创建一个新的Connection Profile配置文件。



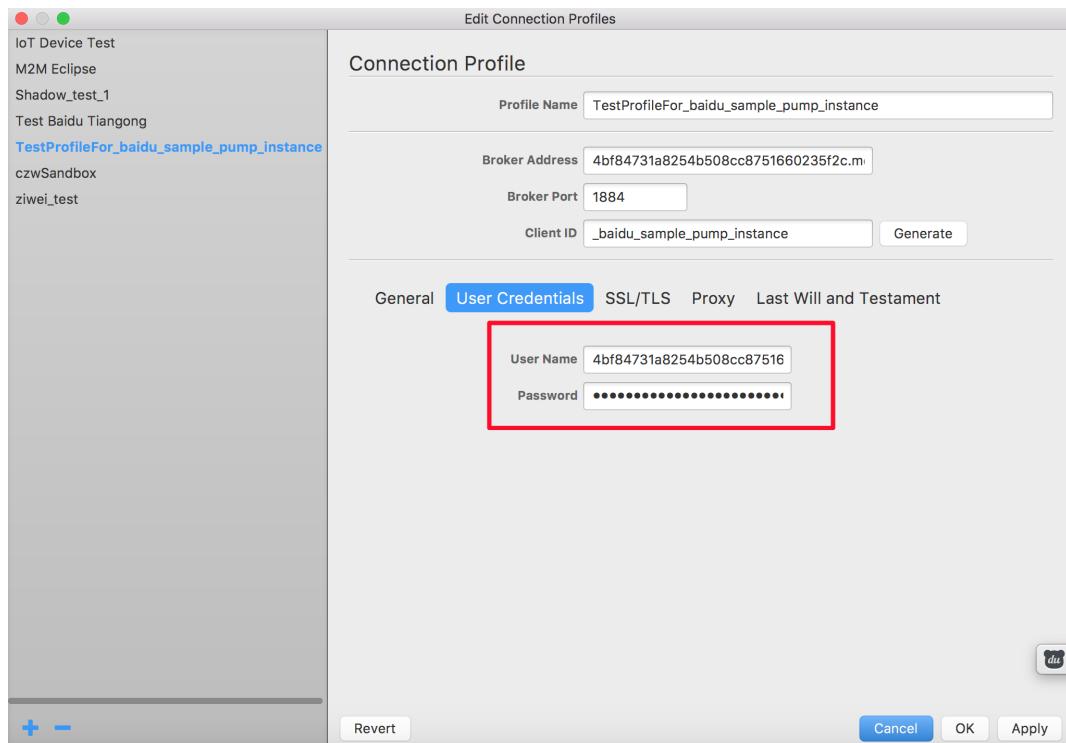


2. 根据物影子示例_baidu_sample_pump_instance的“连接配置”填写Connection Profile的相关信息

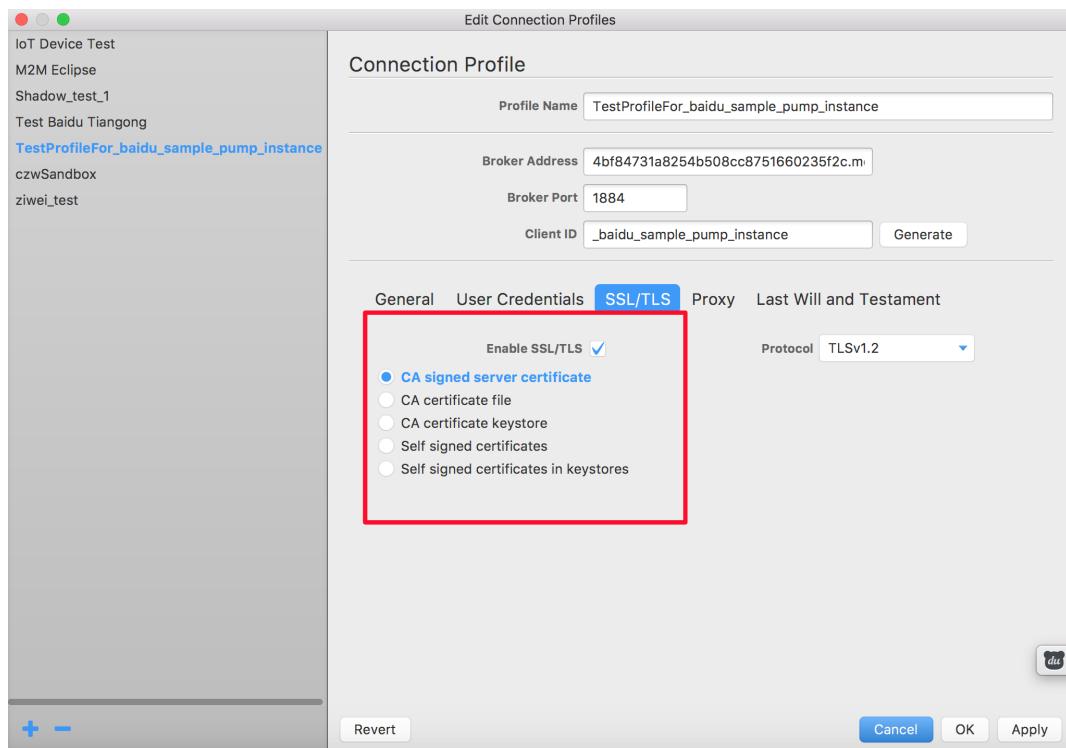
- * **Profile Name:** 该配置文件名，用户自定义。
- * **Broker Address:** 物影子连接配置中Address的hostname，例如"4bf84731a8254b508cc8751660235f2c.mqtt.iot.g
- * **Broker Port:** ssl加密连接方式，端口使用1884；tcp不加密连接，端口使用1883。
- * **Client ID:** 客户端ID。
- * 支持“a-z”, “0-9”, “_”, “-”字符，且不能大于128bytes，UTF8编码，在MQTT.fx客户端中支持随机生成。
- * 在同一个实例下，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。

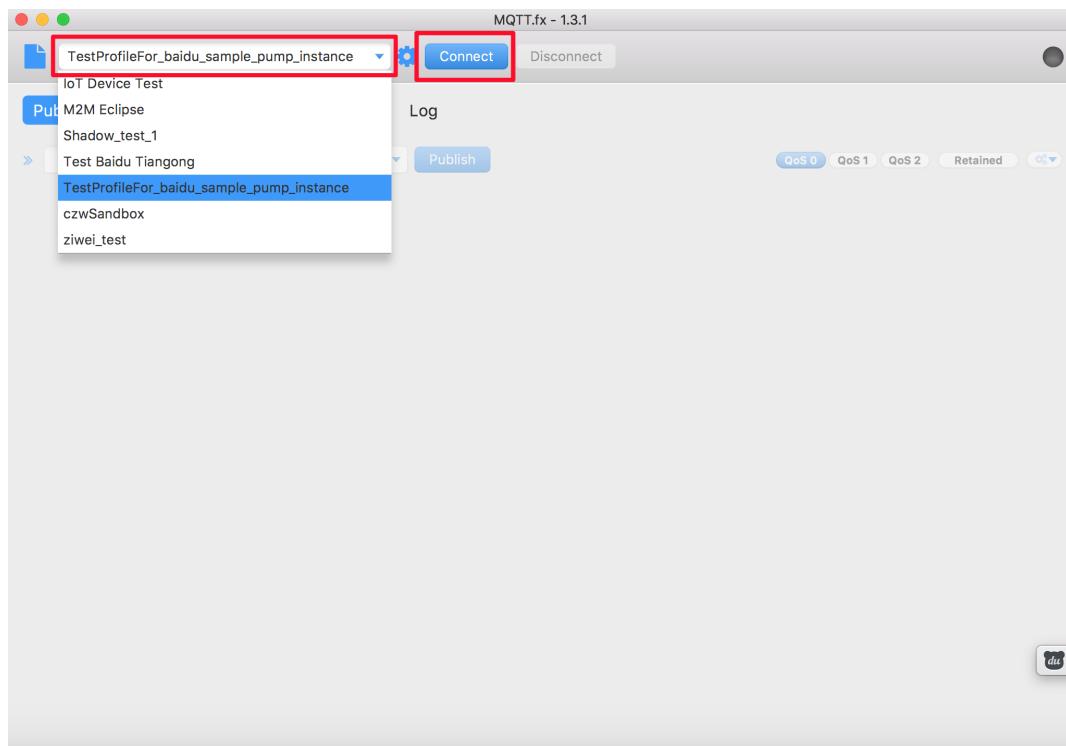


b.选了SSL/TLS，如果您选择SSL安全认证方式连接物影子，需要配置SSL/TLS安全认证，勾选Enable SSL/TLS，选择CA signed server certificate认证。

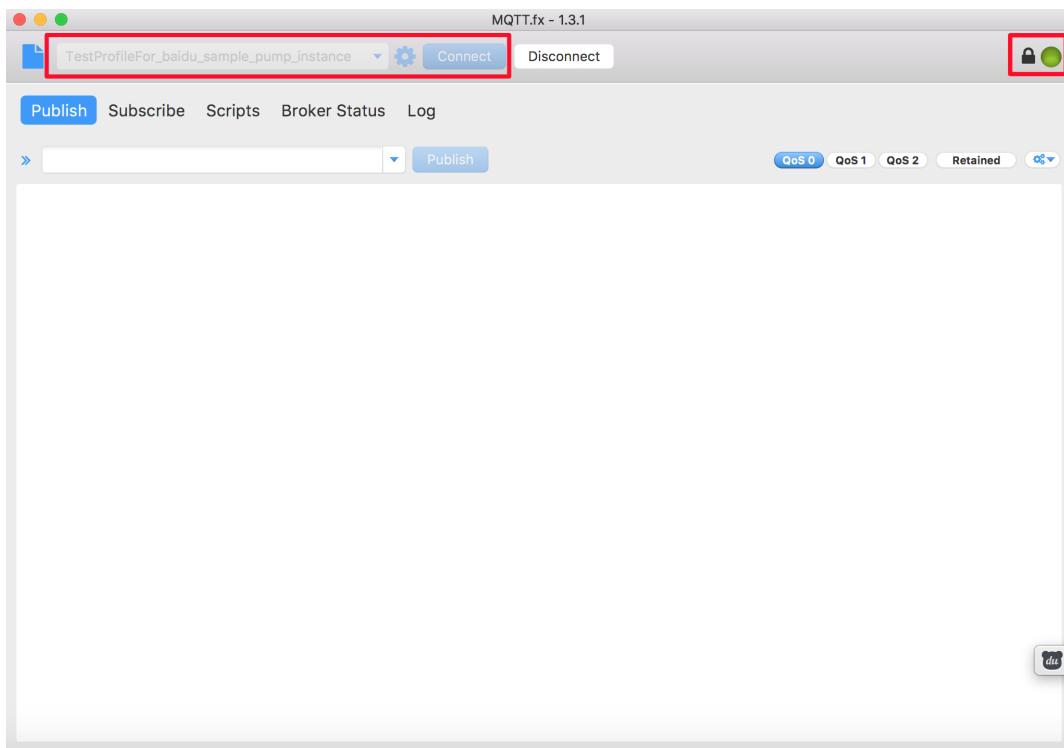


若您选择TCP连接，无需配置SSL安全认证，忽略本步骤即可。

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。



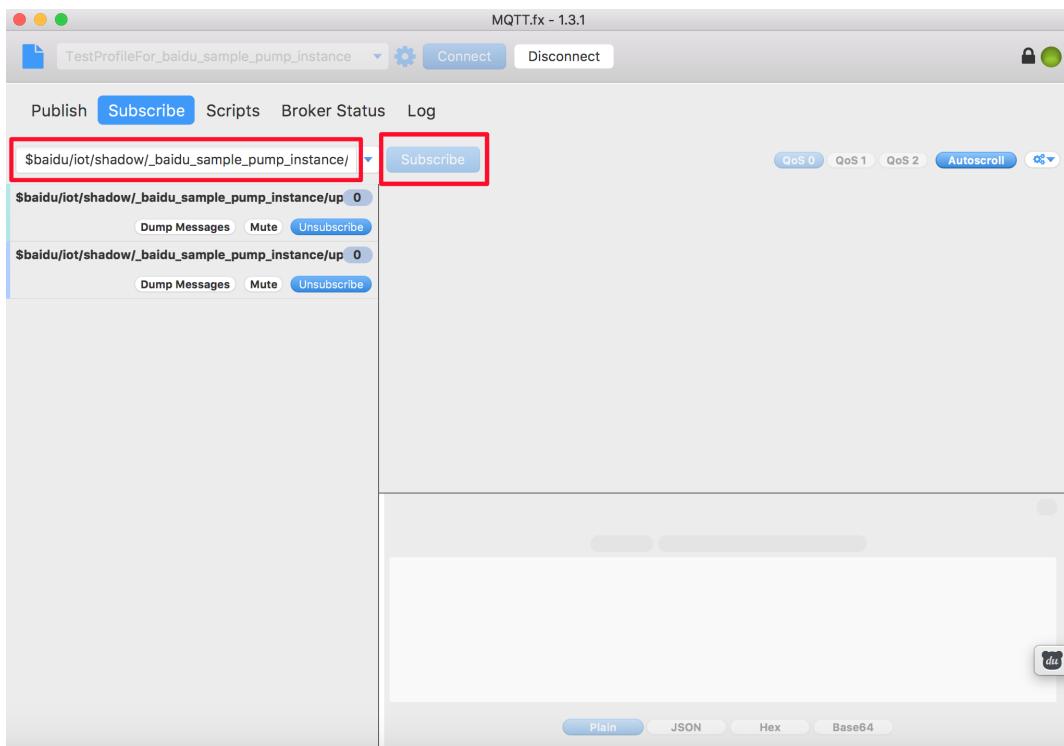
d.左侧下拉框与连接按钮置灰，右侧状态灯变绿，表示连接成功。



更新设备状态到设备影子

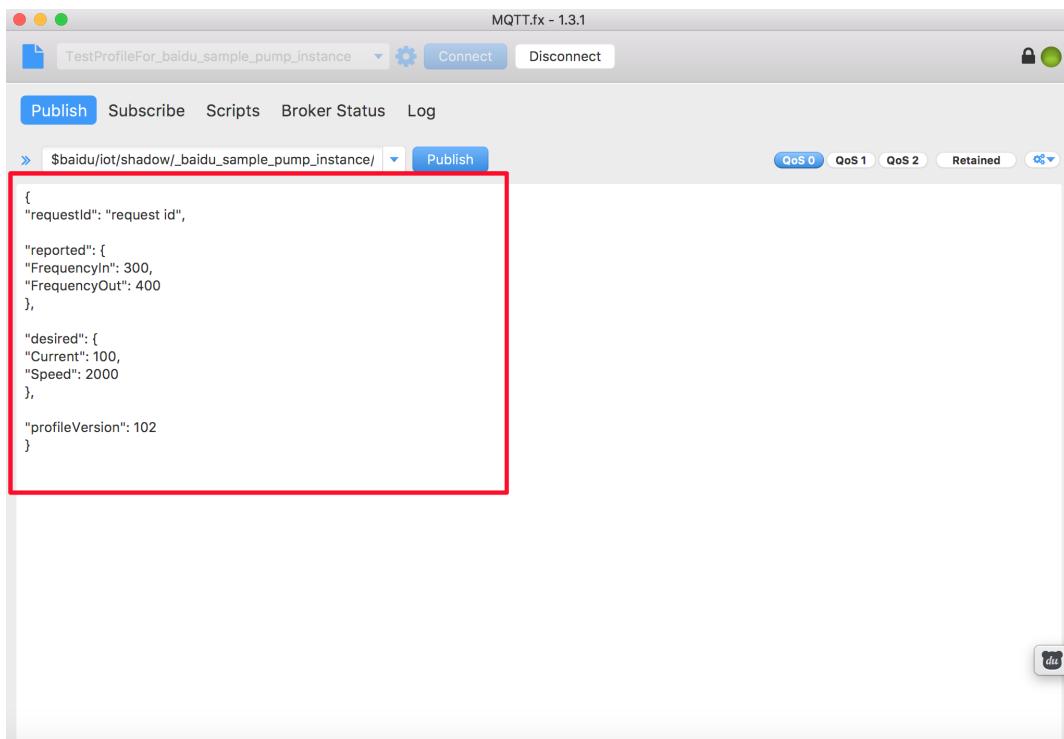
1. 订阅主题：获取设备影子更新状态情况

- 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：

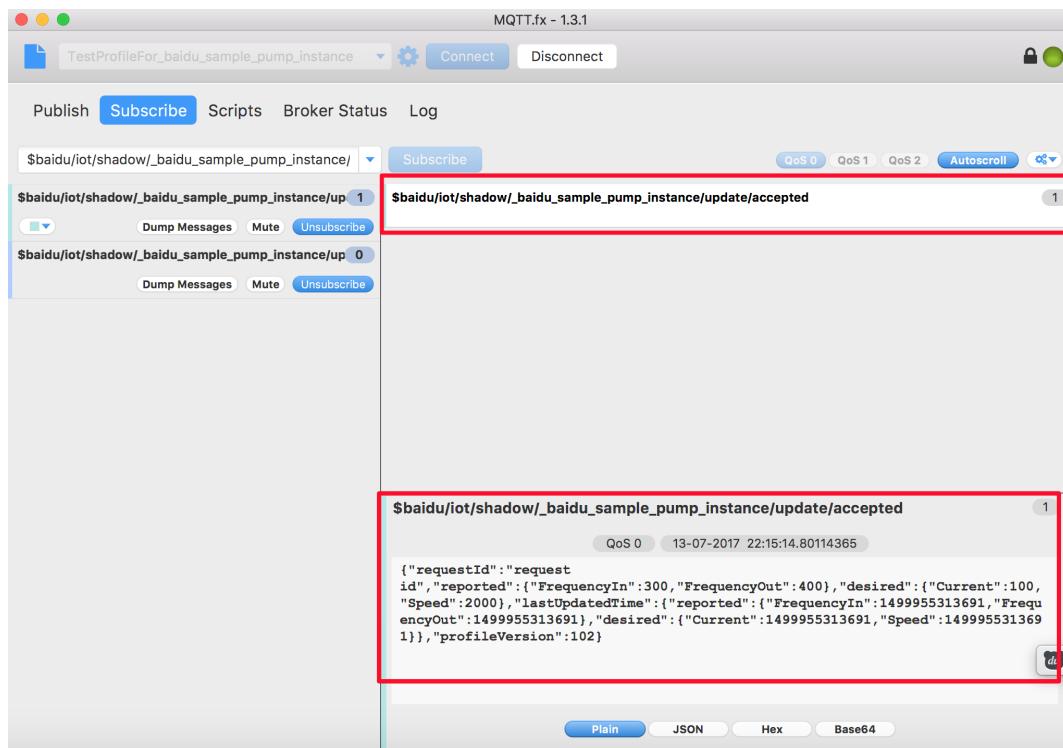
- * "requestId"为请求的唯一标识符，每一个请求的requestId是唯一的，可随机生成。
- * "reported"为可选字段，表示设备影子的"reported"中的相关属性需要更新。
- * "desired"为可选字段，表示设备影子的"desired"中的相关属性需要更新。
- * "profileVersion"为可选字段，当未指定profileVersion时，物管理接收设备影子更新请求后，会将profileVersion自动加1；若指定profileVersion，物管理会检查请求中的profileVersion是否大于当前的profileVersion。只有在大于的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

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。

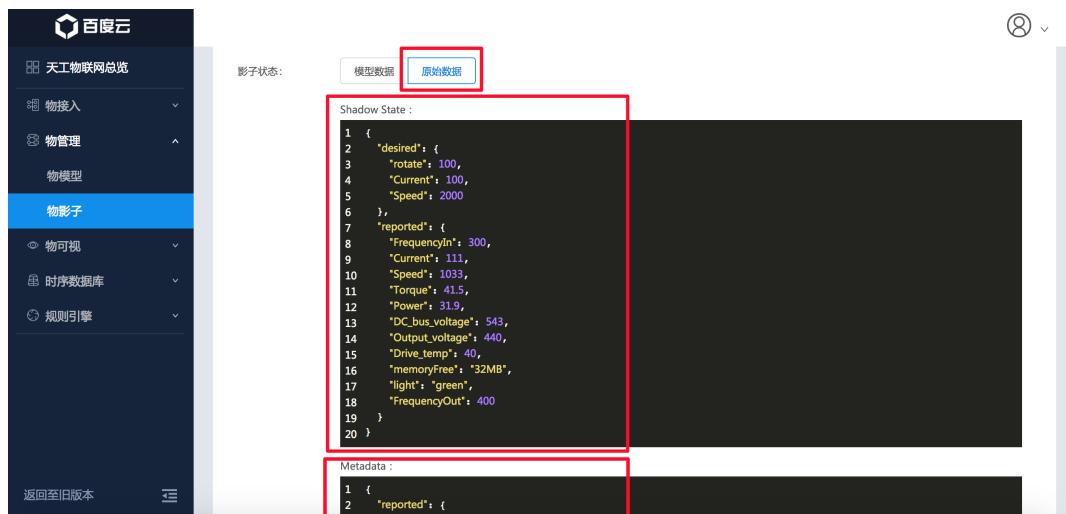
- 期望值：指通过控制台或者应用程序性修改的desired值。如果没有。期望值则为空。
- 发送时间：指该属性最后一次有desired值修改的时间

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。

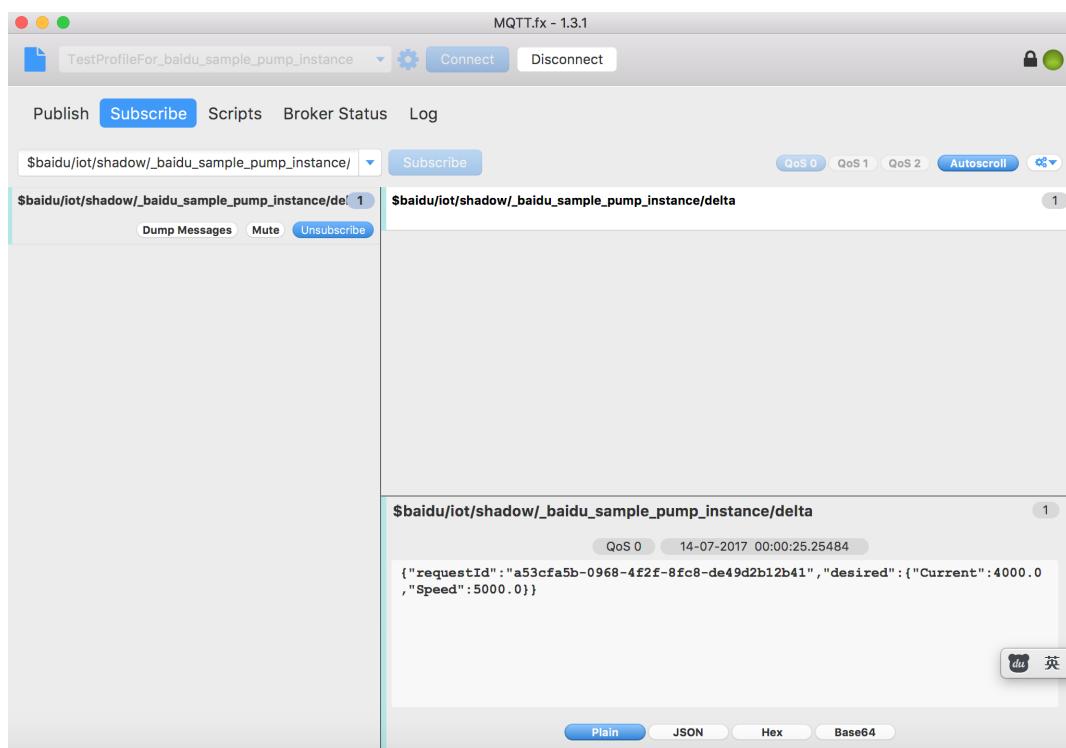
属性名称	显示名称	类型	默认值	单位	当前值	修改时间	期望值	发送时间
FrequencyIn	输入频率	number	0	Hz	300	2017-07-13 22:15:13	输入	2017-07-13 22:59:38
FrequencyOut	输出频率	number	0	Hz	400	2017-07-13 22:15:13	输入	2017-07-13 22:59:38
Current	电流	number	0	A	111	2017-07-06 14:14:54	100	2017-07-13 22:15:13
Speed	速度	number	0	rpm	1033	2017-07-06 14:14:54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

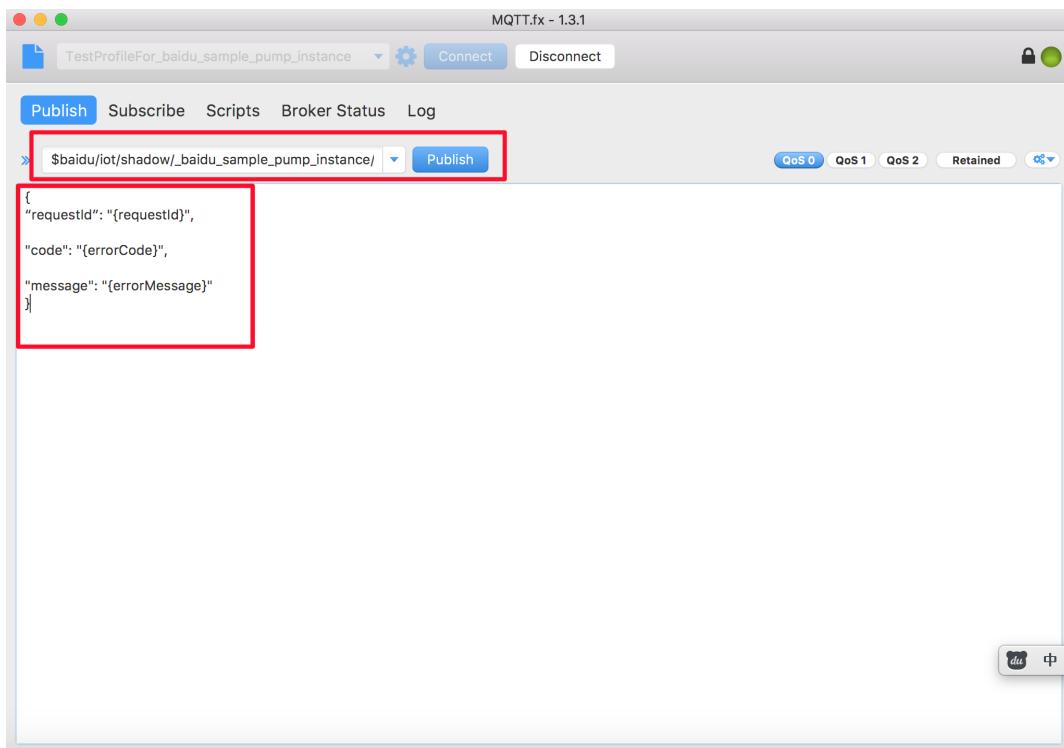
例如，当前‘reported’中的‘Current’字段为111，控制端将‘desired’中的‘Current’字段更新为4000，此时物管理会通过delta主题反控设备

The screenshot shows the Baidu IoT Cloud Device 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' and 'Speed' properties have their current values 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 client interface. A publish message is being sent to the topic '\$baidu/iot/shadow/_baidu_sample_pump_instance/delta'. The message payload is: {"requestId": "a53cfa5b-0968-4f2f-8fc8-de49d2b12b41", "desired": {"Current": 4000.0, "Speed": 5000.0}}. This message is highlighted with a red box.

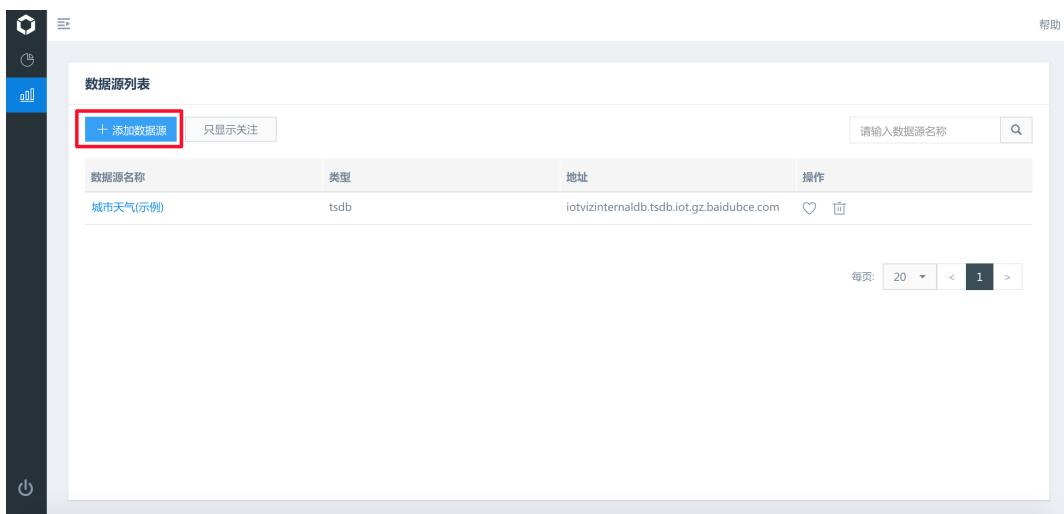
- 若设备更新状态失败，可在设备端Publish主题\$baidu/iot/shadow/myDeviceName/delta/rejected，将相关错误信息发送到物管理：



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

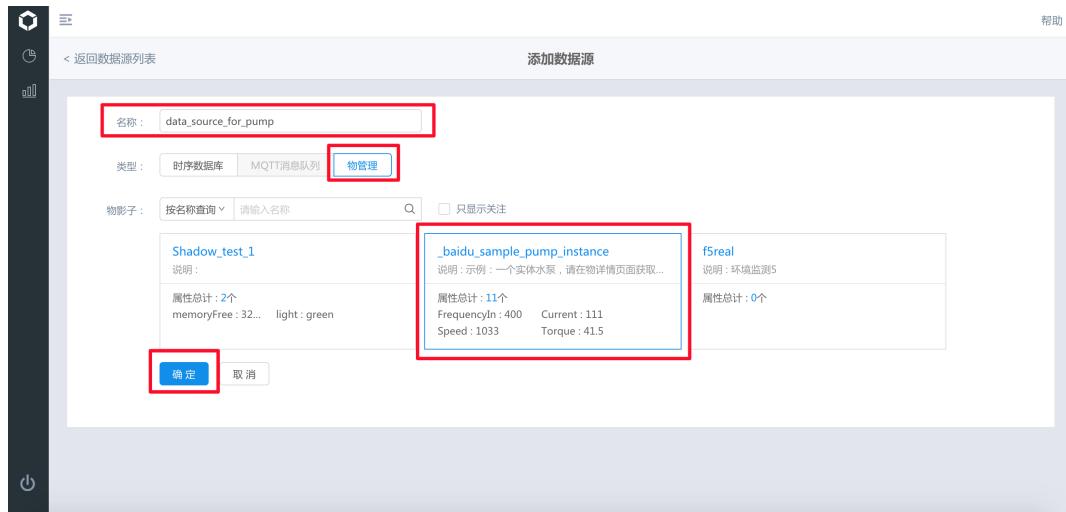
- 在控制台中，选择“产品服务>物可视 IoT Visualization>数据源”，进入物可视服务的数据源列表。点击“添加数据源”。



- 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。

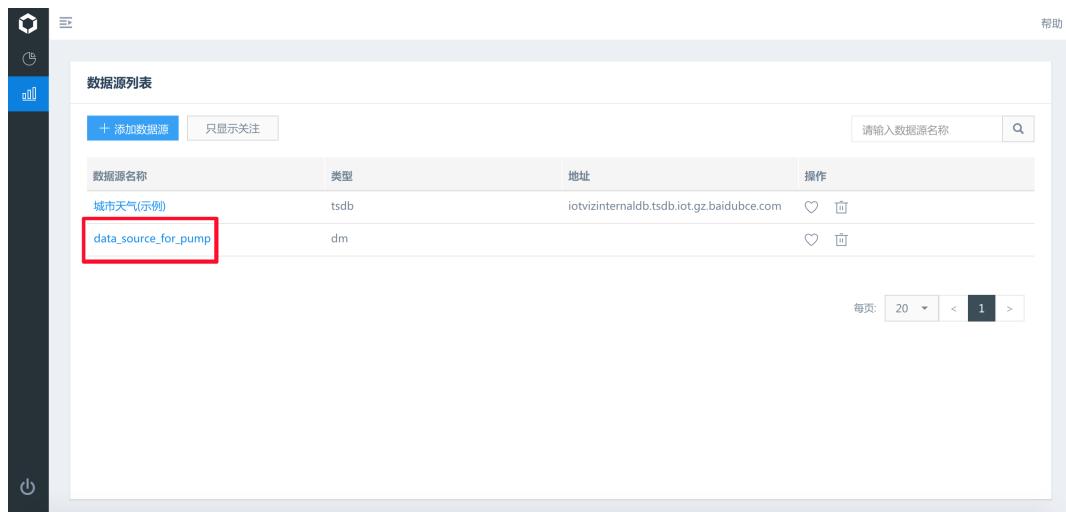
- **类型：**物可视支持读取时序数据库、MQTT消息队列和物管理中的数据。



创建数据流

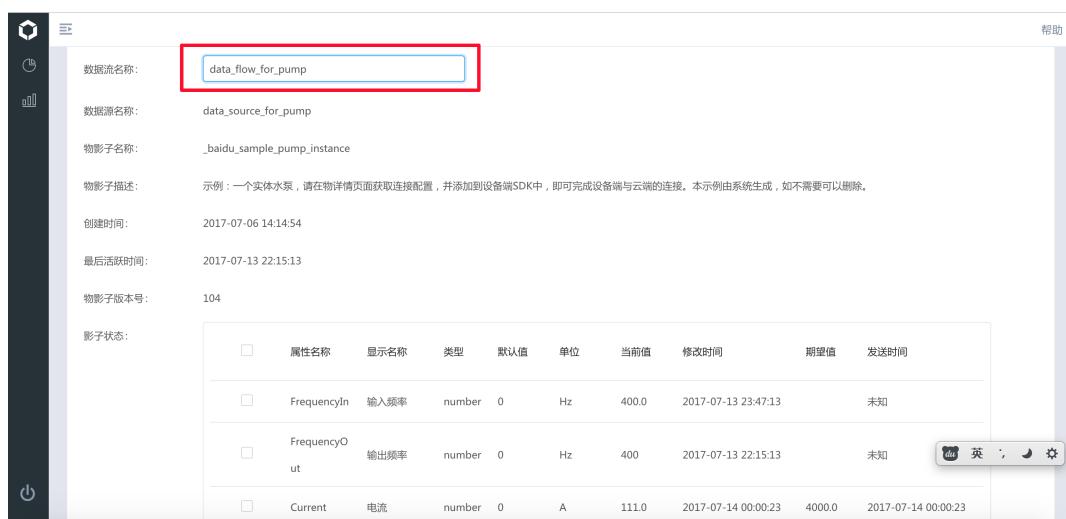
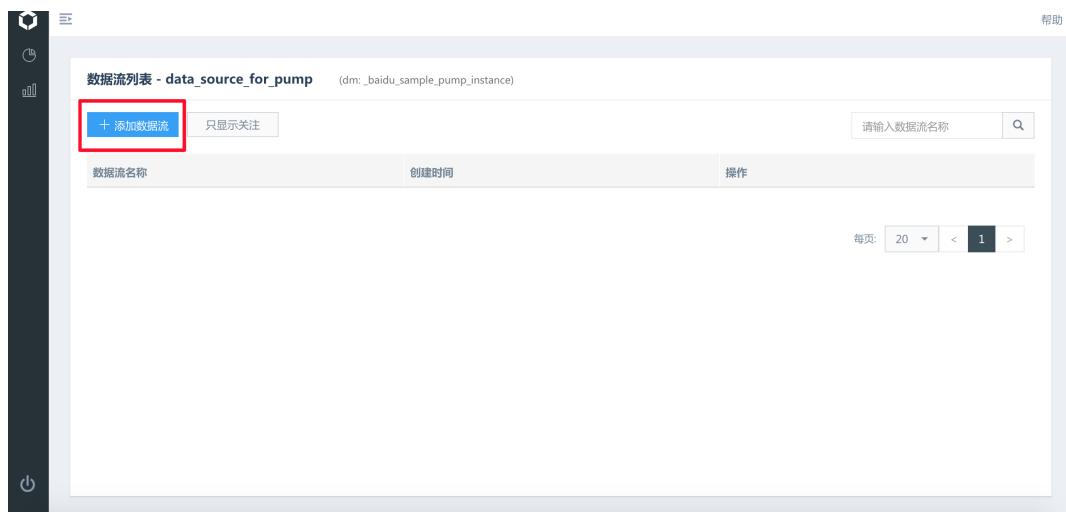
在创建数据流之前，应先完成创建数据源。

- 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



- 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：

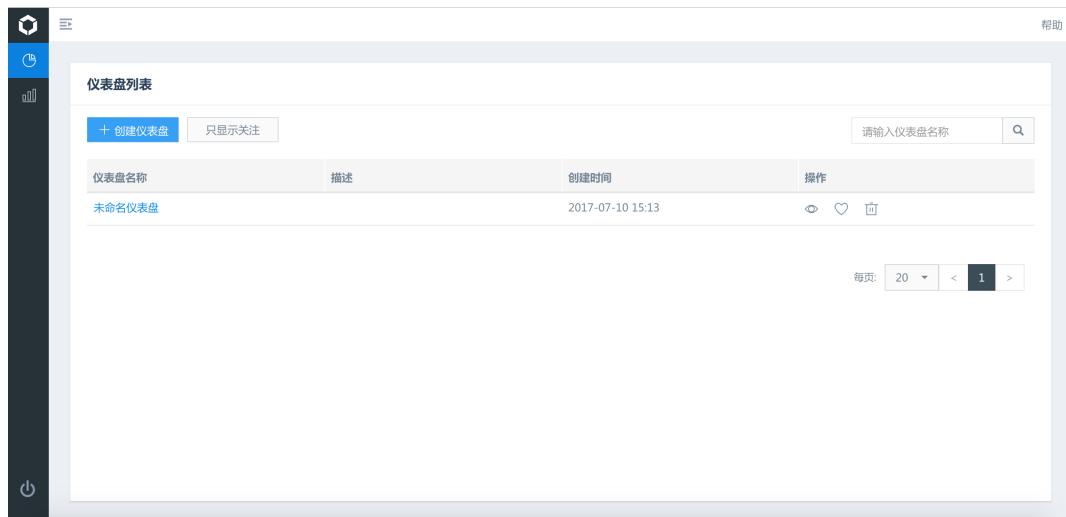
- **名称：**指定数据流名称，支持中英文字符、数字和下划线，支持2-32个字符。
- **数据源：**无需配置。
- **度量：**选择指定度量值的数据，度量值即数据的类别，如发动机的温度、转速等。
- **标签：**通过标签对数据机型过滤。
- **数据处理：**可以对一段时间的数据点做聚合，如每10分钟的和值、平均值、最大值、最小值等。



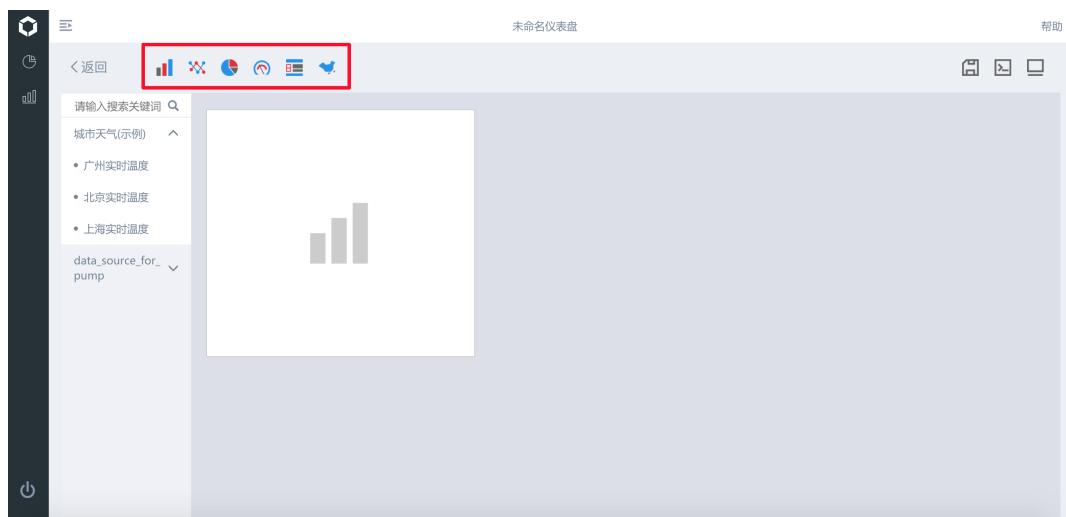
创建仪表盘

在创建仪表盘之前，应先完成创建数据源操作。

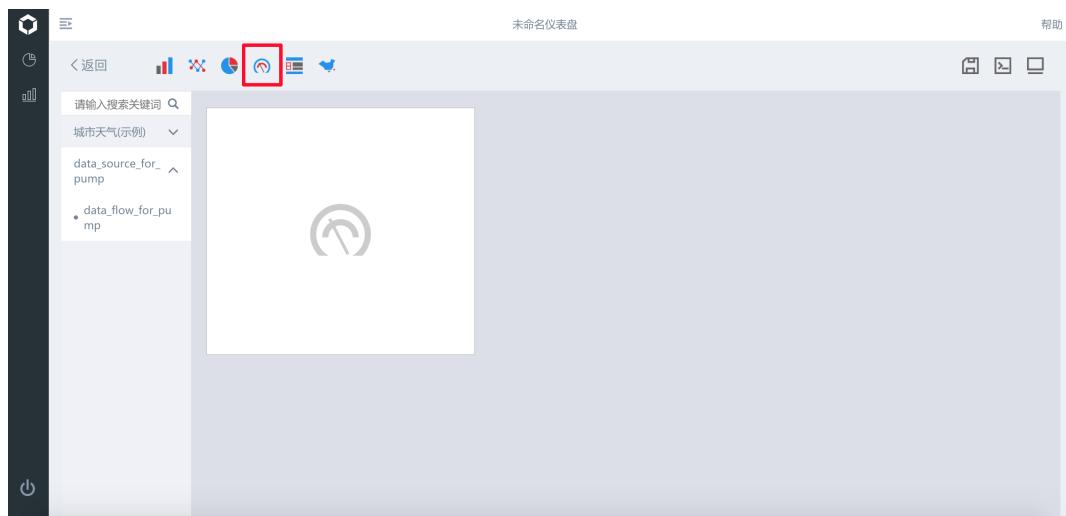
1. 在控制台中，选择“产品服务>物可视 IoT Visualization>可视化设计器>仪表盘”，进入仪表盘列表，点击“创建仪表盘”。



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。

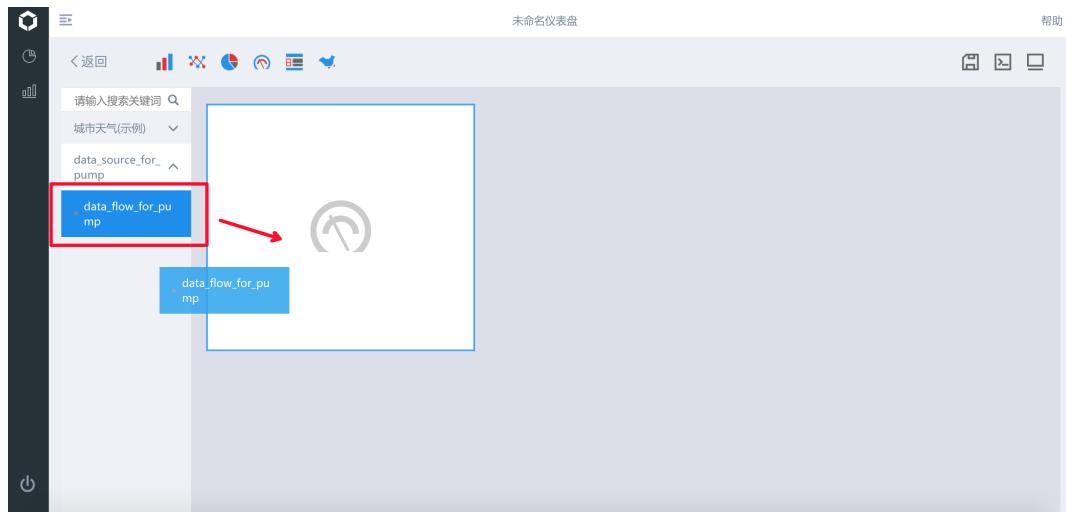


3. 点击“仪表盘”图标，生成一个空仪表盘。

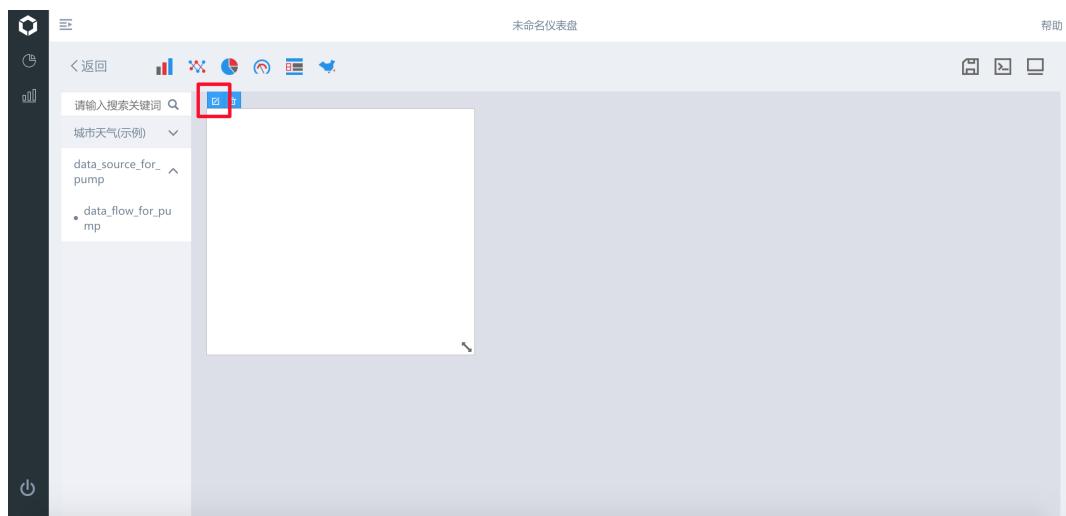


4. 从左侧列表中找到已经添加的数据源，并在该数据源中找到对应的数据流。以上一步中为示例创建的data\flow\for_pump为例，用鼠标将该数据流拖拽到空图表上。此时该空图

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。



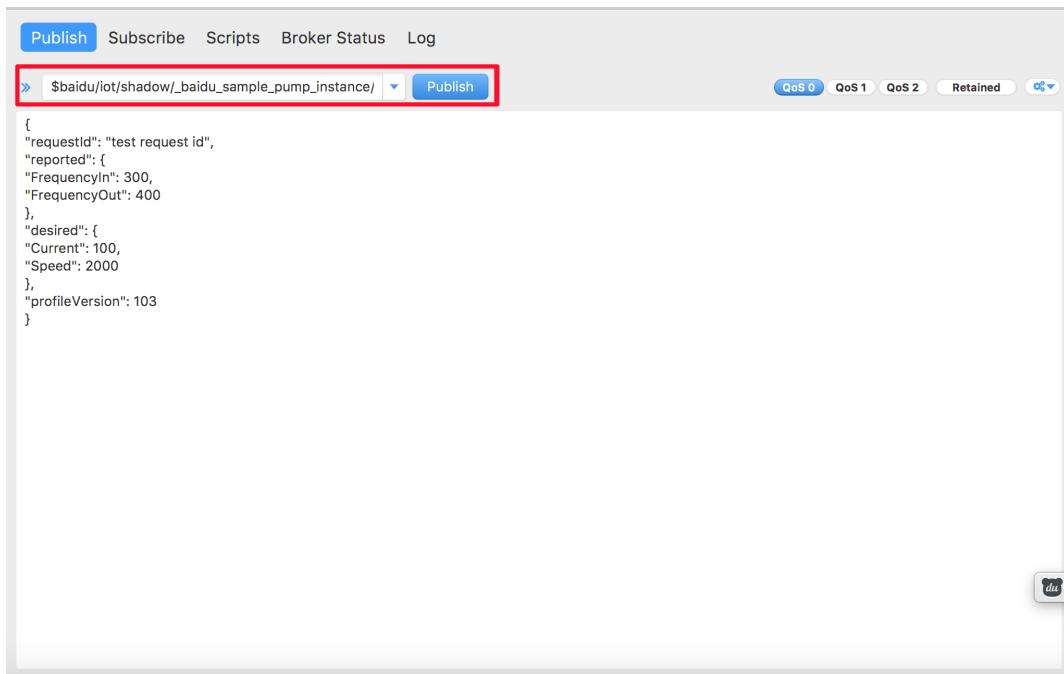
设备在线状态的说明

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 'Device Shadow' card list in the Baidu Cloud IoT Management console. The left sidebar has a 'Device Shadow' section highlighted. The main area displays two cards: 'Shadow_test_1' (offline) and 'baidu_sample_pump_instance' (offline). The 'baidu_sample_pump_instance' card is highlighted with a red box.

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\$baidu/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\baidu\sample\pump\instance，且成功向\$baidu/iot/shadow\baidu\sample\pump\instance/update发布第一条消息。

The screenshot shows the 'Edit Connection Profiles' dialog in MQTT.fx. 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' (highlighted in blue), 'czwSandbox', and 'ziwei_test'. The main panel shows the configuration for 'TestProfileFor_baidu_sample_pump_instance'. The 'Client ID' field is set to '_baidu_sample_pump_instance' and is highlighted with a red box. Other settings include 'Broker Address' (4bf84731a8254b508cc8751660235f2c.mqtts.net), 'Broker Port' (1884), 'Connection Timeout' (30), 'Keep Alive Interval' (60), 'Clean Session' (checked), 'MQTT Version' (3.1.1), and 'User Credentials' tab selected. Buttons at the bottom include 'Revert', 'Cancel', 'OK', and 'Apply'.



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		2017-07-14 14:58:06
FrequencyOut	输出频率	number	0	Hz	400.0	2017-07-13 23:47:13		2017-07-14 14:58:06

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。

5.2 基于物管理快速搭建物联网应用

5.2.1 简介

本文档介绍了基于物管理快速搭建一个物联网应用的方式，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](#)。

应用场景

一个智能设备（如智能家居设备）生产厂商，智能设备部署到各地，需要按照一定的频率回传状态数据。

应用端需要收到这些数据进行下一步消费。同时这些数据也需要用一个手机APP端实时查看。

5.2.2 1. 云端创建设备影子

通过控制台或API创建物影子，每一个物影子就是一个连接到云端的实体设备。创建物影子时，每一个物影子都会生成一个连接用户名和密码，将该用户名密码的配置到设备端的连接程序中。

- 如果使用MQTT开源SDK，则每个物影子有独立的MQTT连接，该用户名密码即作为MQTT连接的用户名密码。
- 如果使用天工Edge SDK（下载地址：<https://github.com/baidu/iot-edge-c-sdk>），可以在如下路径中找到物管理的sample

[iot-edge-c-sdk / iothub_client / samples / iotdm_client_sample / iotdm_client_sample.c](#)

并将如下位置，替换成要每个物影子生成的相对应的用户名、密码。

```
// 2. "ssl://xxxxxx.mqtt.iot.xx.baidubce.com:1884"  
#define ADDRESS "tcp://xxxxxx.mqtt.iot.xx.baidubce.com:1883"  
  
// The device name you created in device management service.  
#define DEVICE "xxxxxx"  
  
// The username you can find on the device connection configuration web,  
// and the format is like "xxxxxx/xxxxx"  
#define USERNAME "xxxxxx/xxxxxx"  
  
// The key (password) you can find on the device connection configuration web.  
#define PASSWORD "xxxxxx"
```

5.2.3 2. 设备端发消息更新云端的物影子

如果使用物管理SDK，物管理SDK已经封装好了更新物影子的函数 [iotdm_client_update_shadow_with_bin](#)，更新成功后，云端的物影子即更新了相应的值。对于物影子的操作，物管理不仅提供更新状态到物影子，还可以

- 从物影子获取最新状态。
- 通过物影子反控设备。
- 获取本次物影子与上一次的变化。
- 获取物影子的实时快照。

物管理SDK都提供相应操作的封装函数，可以直接调用。物影子操作的相关介绍请参考：[物影子操作](#)

如果使用MQTT开源SDK，设备端按照生成的用户名密码连接云端后，可以向主题 `$baidu/iot/shadow/{thingName}/update` 发布消息。需要符合物影子的schema，具体详情请见[操作指南](#)

其他功能，如从物影子获取最新状态、获取本次物影子与上一次的变化、获取物影子实时快照等操作也有相对应的mqtt主题，详情请参见[操作指南](#)

特别注意：物影子名称是主题名称的一部分，如`$baidu/iot/shadow/{thingName}/update`，需要更新到设备端的代码中。物影子生成的用户名密码，只能对该物影子进行操作，即只对该物影子的一系列主题有发布或订阅的权限。

5.2.4 3. 设备端与云端通信（非更新物影子）

如果设备端除了更新物影子以外，还有其他的数据信息需要上传到云端。物管理提供自定义的主题，供用户使用。

- define类主题，形如 `$baidu/iot/define/{thingName}/#`，用户可以在『#』处自定义字段，如`$baidu/iot/define/{thingName}/abc/sys`等。该主题的特点是，主题名中含有thingName（物影子名称）。物影子生成的用户名密码只能向该物影子发布和订阅消息，不能对其他物影子的define类主题发布和订阅消息。
- general类主题，形如 `$baidu/iot/general/#`，用户可以在『#』处自定义字段，如`$baidu/iot/general/alarm`等。该主题的特点是，任意一个物影子的连接或者权限组的连接，都对该主题拥有订阅或发布权限，可以用于不同设备之间的通信。

5.2.5 4. 应用服务和手机APP端消费设备的数据

用户的应用服务程序可以通过物管理的API接口，通过HTTP来获取设备的数据，如获取物影子实时数据等。

如果用户的应用服务程序需要实时订阅到物影子的变化，则可以通过MQTT的连接获取。

- 物影子生成的用户名密码，同样可以在应用程序或手机APP中连接到物管理，订阅与该物相关的主题，如`$baidu/iot/shadow/{thingName}/update/accepted`或`$baidu/iot/define/{thingName}/abc`等，就可以实时订阅到设备端发到云端的消息。
- 应用服务程序或手机APP使用MQTT连接时需要注意：clientID不能是物影子名称。因为设备端的MQTT连接中的clientID是物影子名称，并以此来作为设备是否在线的判断标准，详情请参见：[链接](#)
- 对于一个APP需要同时订阅多个设备的数据的情况，可以利用物管理的权限管理来完成。物管理会为每一个权限组分配一个用户名密码，权限组可以添加多个设备，则这

个权限组即拥有这多个设备的权限，包括物影子操作（shadow类主题）和物的自定义主题（define类主题）。一个手机APP用此权限组的用户名密码建立MQTT连接，即可以实时订阅该权限组里的设备的信息。

第6章 API参考 - V3

6.1 目录

- 介绍
 - 多区域选择
- 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
- 物管理数据写入TSDB
 - 创建规则
 - 获取规则详情
 - 修改规则
 - 删除规则
 - 禁用一条规则
 - 启用一条规则
 - 参数定义
- 权限管理

- [设备权限组管理](#)
- [参数定义](#)
- [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](#)

6.2 介绍

6.2.1 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。

6.2.2 多区域选择

“华南-广州”区域

- 区域的API地址为：iotdm.gz.baidubce.com

“华北-北京”区域

- 区域的API地址为：iotdm.bj.baidubce.com

6.3 设备列表管理

6.3.1 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	描述
schemaId	string	必选	物模型

[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+XVaHpxfKdMPvmdJzN0bvbY="
}

```

6.3.2 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"
  ]
}

```

6.3.3 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
    }
}

```

6.3.4 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
}
]
}

```

6.3.5 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

参数名称	参数类型	是否必须	说明
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&amount=20&pageNo=1&pageSize=10&favourite=true

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
        }
    }
}
}

...
]
```

}

6.3.6 获取设备接入详情

方法	API	说明
GET	/v3/iot/management/device/{deviceName}/access-Detail	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"
}
```

6.3.7 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+XVaHpxfKdMPvmdJzNObvbY="
}
```

6.3.8 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": {
      "time": 1494904250
    }
  }
}
```

```

    "reported": {
        "firewareVersion": 1494904250,
        "light": 1494904250
    },
    "desired": {
        "light": 1494904250
    }
},
"profileVersion": 10
}
}
}

```

6.3.9 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-5ddac970054
{
  "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
    }
  ]
}

```

6.3.10 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
}
}

```

6.3.11 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"
  ]
}

```

6.4 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。

6.4.1 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-5ddac970054
{
  "schemaid": "ab13ef935b84173a0f41f90c33daa87"
}
```

6.4.2 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-5ddac970054
```

注：删除模板，要求没有设备引用该模板。否则，会抛出异常。

6.4.3 获取模板列表

方法	**API**	**说明**
GET	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10
order	String	可选	查询结果升序或降序排列，asc desc，默认asc

参数名称	**参数类型**	**是否必须**	**说明**
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,
            "version": 1
        }
    ]
}
```

```

"properties": [
    {
        "name": "temperature",
        "type": "number",
        "displayName": "温度",
        "defaultValue": "38.2",
        "unit": "deg"
    },
    {
        "name": "pressure",
        "type": "number",
        "displayName": "压力",
        "defaultValue": "9",
        "unit": "MPa"
    }
]
}

```

6.4.4 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"
        }
    ]
}

```

6.4.5 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
```

6.5 物管理数据写入TSDB

6.5.1 创建规则

方法	API	说明
POST	/v3/iot/rules/device/{deviceName}	创建规则引擎的规则

[请求参数](#)

DeviceRuleRequest

[返回参数](#)

DeviceRuleResponse

[请求示例](#)

```
POST /v3/iot/rules/device/myDeviceName HTTP/1.1
```

```
Host: iotdm.gz.baidubce.com
```

```
Authorization:{authorization}
```

```
Content-Type: application/json; charset=utf-8
```

```
{
    "name": "device xxxx to TSDB yyyy",
    "sources": [
        {
            "description": "This is condition 1",
            "name": "name",
            "type": "string",
            "condition": "<>",
            "value": "aaa"
        }, {
            "description": "", // 可以为空
            "name": "temperature",
            "type": "number",
            "condition": ">=",
            "value": "20"
        }, {
            "description": "This is condition 3",
            "name": "speed",
            "type": "number",
            "condition": ">",
            "value": "0"
        }
    ],
    "destinations": [
        {
            "value": "test.tsdb.iot.gz.baidubce.com",
            "kind": "TSDB"
        }
    ]
}
```

```
]  
}
```

[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": "63d92c1de2bd46e0b257c6df67b4a7e9",  
    "deviceName": "myDeviceName",  
    "name": "device xxxx to TSDB yyyy",  
    "sources": [  
        {"description": "This is condition 1",  
         "name": "name",  
         "type": "string",  
         "displayName": "名字",  
         "unit": "count",  
         "defaultValue": "0",  
         "condition": "<>",  
         "value": "aaa",  
         "lastSaveTime": 0  
        }, {  
            "description": "",  
            "name": "temperature",  
            "type": "number",  
            "displayName": "数量",  
            "unit": "piece",  
            "defaultValue": "1",  
            "condition": ">=",  
            "value": "20",  
            "lastSaveTime": 1494904250  
        }, {  
            "description": "This is condition 3",  
            "name": "speed",  
            "type": "number",  
            "displayName": "速度",  
            "unit": "km/s",  
            "defaultValue": "60",  
            "condition": ">",  
            "value": "0",  
            "lastSaveTime": 1494904250  
        }  
    ]  
}
```

}, { // 这个规则是来自Schema里的新的属性，用户在之前的规则建立的时候并没有创建对应的规则。如果创建的规则在Schema里没有对应的属性，则那条规则不会显示

```

    "description": "",
    "name": "temp",
    "type": "number",
    "displayName": "温度",
    "unit": "c",
    "defaultValue": "36",
    "condition": "",
    "value": "",
    "lastSaveTime": 0
},
"destinations": [
{
    "uuid": "6653da99bf9a4e35ba4f997e000a699f",
    "value": "test.tsdb.iot.gz.baidubce.com",
    "kind": "TSDB"
},
],
"enable": true,
"createTime": 1494904250,
"updateTime": 1494904250
}
}
```

6.5.2 获取规则详情

请求参数

方法	API	说明
GET	/v3/iot/rules/device/{deviceName}	获取单个规则的规则详情

返回参数

DeviceRuleResponse

请求示例

```

GET /v3/iot/rules/device/myDeviceName 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": "63d92c1de2bd46e0b257c6df67b4a7e9",
    "deviceName": "myDeviceName",
    "name": "device xxxx to TSDB yyyy",
    "sources": [
        {
            "description": "This is condition 1",
            "name": "name",
            "type": "string",
            "displayName": "名字",
            "unit": "count",
            "defaultValue": "0",
            "condition": "<>",
            "value": "aaa",
            "lastSaveTime": 0
        },
        {
            "description": "This is condition 2",
            "name": "temperature",
            "type": "number",
            "displayName": "数量",
            "unit": "piece",
            "defaultValue": "1",
            "condition": ">=",
            "value": "20",
            "lastSaveTime": 1494904250
        },
        {
            "description": "This is condition 3",
            "name": "speed",
            "type": "number",
            "displayName": "速度",
            "unit": "km/s",
            "defaultValue": "60",
            "condition": ">",
            "value": "0",
            "lastSaveTime": 1494904250
        }
    ],
    // 这个规则是来自Schema里的新的属性，用户在之前的规则建立的时候并没有创建对应的规则。如果创建的规则在Schema里没有对应的属性，则那条规则不会显示
}
```

```

    "description": "",
    "name": "temp",
    "type": "number",
    "displayName": "温度",
    "unit": "c",
    "defaultValue": "36",
    "condition": "",
    "value": "",
    "lastSaveTime": 0

}],
"destinations": [
{
    "uuid": "6653da99bf9a4e35ba4f997e000a699f",
    "value": "test.tsdb.iot.gz.baidubce.com",
    "kind": "TSDB"
}
],
"enable": true,
"createTime": 1494904250,
"updateTime": 1494904250
}

```

6.5.3 修改规则

方法	API	说明
PUT	/v3/iot/rules/device/{deviceName}	修改单个规则

请求参数

参数名	参数类型	是否必须	说明	示例
name	String	可选	规则名称	过滤传感器温度过高消息的规则
sources	List<DeviceRuleSource>	可选	需要存储的数据的属性和条件	

参数名	参数类型	是否必须	说明	示例
destinations	List<DeviceRuleDestination>	可选	处理后的消息写往的目的地数组 (TSDB, KAFKA, 另一个 MQTT主题) 目前只支持TSDB	

[返回参数](#)

DeviceRuleResponse

[请求示例](#)

```

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

// 全修改
{
    "name": "device xxxx to TSDB yyyy",
    "sources": [
        {
            "description": "",
            "name": "name",
            "type": "string",
            "condition": "<>",
            "value": "bbbbbb"
        },
        {
            "description": "This is condition 2",
            "name": "temperature",
            "type": "number",
            "condition": "<=",
            "value": "100000"
        }
    ],
    "destinations": [
        {
            "value": "TSDB_NAME", // e.g. test.tsdb.iot.gz.baidubce.com
            "kind": "TSDB"
        }
    ]
}

// 只修改名字

```

```
{  
    "name": "cccc" // 亦可为空  
}  
  
{  
    "name": "cccc",  
    "sources": [] // 亦可为空  
}  
  
// 只修改 sources  
{  
    "sources": [  
        {"  
            "description": "",  
            "name": "name",  
            "type": "string",  
            "condition": "<>",  
            "value": "bbbbbb"  
        }, {  
            "description": "This is condition 2",  
            "name": "temperature",  
            "type": "number",  
            "condition": "<=",  
            "value": "100000"  
        }]  
}  
  
// 只修改 destinations  
{  
  
    "destinations": [  
        {  
            "value": "TSDB_NAME", // e.g. test.tsdb.iot.gz.baidubce.com  
            "kind": "TSDB"  
        }  
    ]  
}
```

[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": "63d92c1de2bd46e0b257c6df67b4a7e9",  
    "deviceName": "myDeviceName",  
    "name": "device xxxx to TSDB yyyy",  
    "sources": [{  
        "description": "This is condition 1"  
        "name": "name",  
        "type": "string",  
        "displayName": "名字",  
        "unit": "count",  
        "defaultValue": "0",  
        "condition": "<>",  
        "value": "aaa",  
        "lastSaveTime": 0  
    }, {  
        "description": "This is condition 2",  
        "name": "temperature",  
        "type": "number",  
        "displayName": "数量",  
        "unit": "piece",  
        "defaultValue": "1",  
        "condition": ">=",  
        "value": "20",  
        "lastSaveTime": 1494904250  
    }, {  
        "description": "This is condition 3",  
        "name": "speed",  
        "type": "number",  
        "displayName": "速度",  
        "unit": "km/s",  
        "defaultValue": "60",  
        "condition": ">",  
        "value": "0",  
        "lastSaveTime": 1494904250  
    }, { // 这个规则是来自Schema里的新的属性，用户在之前的规则建立的时候并没有创建对应的规则。 如果创建的规则在Schema里没有对应的属性，则那条规则不会显示  
        "description": "",  
        "name": "temp",  
        "type": "number",  
        "displayName": "温度",  
        "unit": "c",  
        "defaultValue": "36",  
        "condition": "",  
        "value": "",  
        "lastSaveTime": 0  
    }]  
}
```

```

    },
    "destinations": [
        {
            "uuid": "6653da99bf9a4e35ba4f997e000a699f",
            "value": "test.tsdb.iot.gz.baidubce.com",
            "kind": "TSDB"
        }
    ],
    "enable": true,
    "createTime": 1494904250,
    "updateTime": 1494904250
}

```

6.5.4 删除规则

方法	API	说明
DELETE	/v3/iot/rules/device/{deviceName}	删除规则

请求示例

```

DELETE /v3/iot/rules/device/myDeviceName 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

```

6.5.5 禁用一条规则

方法	API	说明
PUT	/v3/iot/rules/device/{deviceName}?disable	禁用一条规则

[请求示例](#)

```
PUT /v3/iot/rules/device/myDeviceName?disable 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
```

6.5.6 启用一条规则

方法	API	说明
PUT	/v3/iot/rules/device/{deviceName}?enable	启用一条规则

[请求示例](#)

```
PUT /v3/iot/rules/device/myDeviceName?enable 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
```

6.5.7 参数定义

参数名	参数类型	说明	示例
name	String	规则名称， 默认为空	规则名称1
sources	List<DeviceRuleSource>	必须， 需要存储的数据的属性和其条件	
destinations	List<DeviceRuleDestination>	必须， 处理后的消息写往的目的地数组 (TSDB, KAFKA, 另一个MQTT主题) 目前只支持TSDB	[{"kind": "TSDB", "value": "test.tsdb.iot.gz.baidubce.com"}]

DeviceRuleRequest

参数名	参数类型	说明	示例
description	String	非必须， 默认为空， 条件的描述信息， 最长 255 个字符， 默认为空	规则名称1
name	String	必须， 对应的模版的属性名称	temperature
type	String	必须， 属性数据类型， 对应于Schema 的数据类型， string, number, bool, object	
(与 TSDB 的数据类型的对应类型分别为： String, Double, Long, String)	string		
value	String	非必需， 默认为空， 条件对应的阈值	37
condition	String	非必需， 默认为空 (即表示没有约束条件)， 约束条件运算符	>=

DeviceRuleSource 约束条件运算符包括：

运算符 | 描述

----- | -----> | 大于 >= | 大于等于

< | 小于

<= | 小于等于

= | 等于

<> | 不等于 * | 表示有数据即存储，此属性数据都会存到TSDB里，此时条件对应的阀值也无效

| 空，即表示没有约束条件，此项数据不作转发处理，此时条件对应的阀值也无效

参数名	类型	说明	实例
kind	String	必须， 目的 地 类 型， 可 能 取 值:MQTT, KAFKA, TSDB, BOS, 目前 仅限TSDB	TSDB
value	String	必 须, 对 于 TSDB: value 是 目 的 地 TSDB 数 据 库 的 访 问 域 名 (也 即 end- point,e.g. test.tsdb.iot.g z.baidubce.com) 对 于 MQTT: value 是 目 的 地 MQTT 主 题 对 于 KAFKA: value 是 目 的 地 KAFKA 主 题 对 于 BOS: value 是 目 的 地 BOS 的 bucket, 如 bos:// mybucket 对 于 MQTT_DYNAMIC: value 是 一 个 SQL SELECT 子 句, 用 于 从 原 始 消 息 里 面 选 择 字 段 作 为 目 的 地 主 题, 如: dest.topic	

DeviceRuleDestination

参数名	类型	说明	示例
id	String	规则对应的id	63d92c1de2bd46e0b257c6df67b4a7e9
deviceName	String	对应的设备名称	myDevicename
name	String	规则名称	规则名称1
sources	List<DeviceRuleSourceDetail>	规则的具体约束条件	
destinations	List<DeviceRuleDestinationDetail>	处理后的消息写往的目的地数组(TSDB, KAFKA, 另一个MQTT主题)目前只支持TSDB	[{ "kind": "TSDB", "value": "test.tsdb.iot.gz.baidubce.com" }]
enable	Boolean	条件默认开启	true
createTime	Long	创建时间	1494904250
updateTime	Long	更新时间	1494904250

DeviceRuleResponse

参数名	类型	说明	示例
description	String	条件的描述信息，最长255个字符	规则名称1
name	String	对应的模版的属性名称	temperature
type	String	属性数据类型，对应于Schema的数据类型，string, number, bool, object	string
displayName	String	对应的Schema的属性显示名称	温度
unit	String	对应的Schema的数据单位	Mpa
defaultValue	String	对应的Schema的数据默认值	200
value	String	条件对应的阈值	37

参数名	类型	说明	示例
lastSaveTime	Long	最后一次存储时间，没有则为0	1494904250
condition	String	约束条件运算符	>=

DeviceRuleSourceDetail 约束条件运算符包括：

运算符 | 描述

----- | -----> | 大于

>= | 大于等于 < | 小于 <= | 小于等于 = | 等于 <> | 不等于

| 表示有数据即存储，此属性数据都会存到TSDB里，此时条件对应的阀值也无效

| 空，即表示没有约束条件，此项数据不作转发处理，此时条件对应的阀值也无效

参数名	类型	说明	示例
id	String	规则目的地的uuid	6653da99bf9a4e3 5ba4f997e000a 699f
kind	String	目的地类型，可能取值:MQTT, KAFKA, TSDB, BOS, 目前仅限TSDB	TSDB
value	String	对于 TSDB: value 是目的地 TSDB 数据库的访问域名（也即 endpoint, e.g. test.tsdb.iot.gz.baidubce.com 对于 MQTT: value 是目的地 MQTT 主题对于 KAFKA: value 是目的地 KAFKA 主题对于 BOS: value 是目的地 BOS 的 bucket, 如 bos:// mybucket 对于 MQTT_DYNAMIC: value 是一个 SQL SELECT 子句, 用于从原始消息里面选择字段作为目的地主题, 如: dest.topic	

DeviceRuleDestinationDetail

6.6 权限管理

6.6.1 简介

在新版物管理中，我们引入了权限组的概念，权限组拥有对其中包含的所有设备的topic的访问权限。

我们定义了两种权限组的概念，普通权限组和超级权限组。其中，普通权限组即为正常的权限组，可以向权限组内增添删减设备。

而超级权限组则包含对该用户下所有设备的topic访问权限。因而超级权限组不支持组内设备的查看及更新操作。

6.6.2 设备权限组管理

方法	API	说明
POST	/v3/iot/management/domain	创建权限组

创建权限组 请求参数

参数名	参数类型	是否必须	说明
name	String	必须	权限组名称
description	String	必须	权限组的相关说明
type	String	必须	权限组的类型，支持ROOT, NORMAL

返回参数

一个AccessDetail实例

请求示例

```
POST /v3/iot/management/domain HTTP/1.1
```

```
Host:iotdm.gz.baidubce.com
```

```
Authorization:{authorization}
```

Content-Type: application/json; charset=utf-8

```
{  
  "name": "myNormalDomain",  
  "description": "description",  
  "type": "NORMAL"  
}
```

[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",  
  "wssEndpoint": "wss://test.baidu.iot.com",  
  "username": "endpointName/mySuperDomain",  
  "key": "bWwCxGwaw3boV48NqsuG+XVaHpxfKdMPvmdJzNObvbY="  
}
```

方法	API	说明
DELETE	/v3/iot/management/domain/{domainName}	根据domainName删除权限组

[删除权限组 请求示例](#)

```
DELETE /v3/iot/management/domain/{domainName} HTTP/1.1
```

Host:iotdm.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

方法	API	说明
GET	GET / v3/ iot/ management/ domain? pageNo=xx&pageSize=xx&order=xx&orderBy=xx&key=xx&type=xx&deviceName=xx	根据条件查询权限组列表

获取权限组列表 请求参数

参数名	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	一页所包含的最大数量， 默认为10
order	String	可选	查询结果升序或降序排列， asc, desc， 默认desc
orderBy	String	可选	排序的索引列， name, createTime, lastUpdatedTime， 默认 createTime
key	String	可选	查询关键字， 权限组名称

参数名	参数类型	是否必须	说明
type	String	可选	权限组类型, ROOT, NORMAL, ALL, 默认是ALL
deviceName	String	可选	设备名称, 表示查询包含该设备的权限组

[返回参数](#)

参数名	参数类型	说明
amount	int	权限组总数
pageNo	int	当前页码
pageSize	int	一页所包含的最大数量
domains	List<Domain>	权限组列表

[请求示例](#)

GET /v3/iot/management/domain?pageNo=1&pageSize=10&order=desc&orderBy=createTime&key=myDomain&type=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

```
{
    "amount": 1,
    "pageNo": 1,
    "pageSize": 10,
    "domains": [
        {
            "name": "myDomain"
        }
    ]
}
```

```

        "name": "myDomain",
        "description": " description",
        "type": "NORMAL",
        "createTime": 1494904250000,
        "lastUpdatedTime": 1494904250000,
        "deviceNum": 0
    }
]
}

```

方法	API	说明
GET	/v3/iot/management/domain/{domainName}	根据名称获取权限组详情

获取权限组详情 [返回参数](#)

一个DomainDetail实例

请求示例

```
GET /v3/iot/management/domain/{domainName} 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

```
{
    "name": "myDomain",
    "description": " description",
    "type": "NORMAL",
    "createTime": 1494904250000,
    "lastUpdatedTime": 1494904250000,
    "deviceNum": 2,
    "devices": [
        "device-1",

```

```

        "device-2"
    ]
}

```

方法	API	说明
PUT	/v3/iot/management/domain/{domainName}?modify	权限组中更改设备

权限组中更改设备 请求参数

参数名	参数类型	是否必须	说明
addedDevices	List<String>	可选	需要添加的设备名称列表
removedDevices	List<String>	可选	需要移除的设备名称列表

返回参数

参数名	参数类型	是否必须	说明
addedDevices	List<String>	可选	成功添加的设备名称列表
removedDevices	List<String>	可选	成功移除的设备名称列表

请求示例

```
PUT /v3/iot/management/domain/1234939554?modify HTTP/1.1
Host:iothdm.gz.baidubce.com
```

```
Authorization:{authorization}
```

```
{
    "addedDevices": [
        "device-1",
        "device-2"
    ],
    "removedDevices": [
        "device-3",
        "device-4"
    ]
}
```

```
]
}
```

[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

```
{
  "addedDevices": [
    "device-1",
    "device-2"
  ],
  "removedDevices": [
    "device-3",
    "device-4"
  ]
}
```

方法	API	说明
PUT	/v3/iot/management/domain/{domainName}	更新注册信息

[更新权限组注册信息 请求参数](#)

参数名	参数类型	是否必须	说明
description	String	必选	需要更新的设备描述信息

[请求示例](#)

```
PUT /v3/iot/management/domain/1234939554 HTTP/1.1
Host:iothdm.gz.baidubce.com
```

Authorization:{authorization}

```
{
```

```
"description": "new description"  
}
```

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

方法	API	说明
GET	/v3/iot/management/domain/{domainName}/accessDetail	根据名称获取连接详情

获取权限组接入信息 返回参数

一个AccessDetail实例。

请求示例

```
GET /v3/iot/management/domain/{domainName}/accessDetail 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

```
{  
  "tcpEndpoint": "tcp://test.baidu.iot.com",
```

```
"sslEndpoint": "ssl://test.baidu.iot.com",  
"wssEndpoint": "wss://test.baidu.iot.com",  
"username": "endpointName/device_1",  
"key": "xxxxxxxxxx"  
}
```

方法	API	说明
PUT	/v3/iot/management/domain/{domainName}?updateSecretKey	更改密钥

[更新权限组密钥](#) [返回参数](#)

一个AccessDetail实例。

[请求示例](#)

```
PUT /v3/iot/management/domain/{domainName}?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",
"wssEndpoint": "wss://test.baidu.iot.com",
"username": "endpointName/domainName",
"key": "bWwCxGwaw3boV48NqsuG+XVaHpxfKdMPvmdJzNObvY="

}

```

方法	API	说明
GET	GET / v3/ iot/ management/ domain/ {domainName}/ devices? pageNo=xx&pageSize=xx&order=xx&orderBy=xx&name=xx&value=xx&favourite=xx	根据条件查询设备列表，返回设备名字、是否存在与权限组内、设备所在普通权限组数及信息

获取及查询设备列表 请求参数

参数名	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	一页所包含的最大数量，默认为10
order	String	可选	查询结果升序或降序排列，asc,desc，默认desc
orderBy	String	可选	排序的索引列，name, createTime, lastUpdatedTime， 默认createTime
name	String	可选	查询属性名
value	String	可选	查询属性值
favourite	String	可选	收藏，true, false, all，默认all

返回参数

参数名	参数类型	说明
pageNo	int	当前页码

参数名	参数类型	说明
pageSize	int	一页所包含的最大数量
amount	int	设备总数
devices	List<DeviceInDomain>	设备列表

请求示例

```
GET /v3/iot/management/domain/{domainName}/devices?pageNo=1&pageSize=10&orderBy=createTime&order=d
```

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

```
{
    "amount": 2,
    "pageNo": 1,
    "pageSize": 10,
    "devices": [
        {
            "deviceName": "device_1",
            "existed": true,
            "domainNum": 1
        },
        {
            "deviceName": "device_2",
            "existed": false,
            "domainNum": 2
        }
    ]
}
```

6.6.3 参数定义

Domain参数列表

参数名	参数类型	说明
name	String	权限组名称
description	String	权限组说明
type	String	权限组的类型，支持ROOT, NORMAL，二者选其一
createTime	long	权限组创建时间
lastUpdatedTime	long	权限组更新时间
deviceNum	int	权限组包含设备数目

DomainDetail参数列表

参数名	参数类型	说明
Domain对应项	Domain	权限组基本信息
devices	List<String>	设备列表

AccessDetail参数列表

参数名	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
wssEndpoint	String	支持wss的物接入endpoint
username	String	endpointName/ thingName
key	String	物接入Thing密钥

DeviceInDomain参数列表

参数名	参数类型	说明
deviceName	String	设备名称
existed	boolean	是否存在该权限组内
domainNum	int	设备被包含在普通权限组内的数目

6.7 参数定义

6.7.1 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

6.7.2 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

6.7.3 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	long	设备创建时间(以毫秒为单位)
state	String	设备状态
lastActiveTime	long	设备最后一次的活跃时间(以毫秒为单位)
schemaId	String	设备采用模型Id
schemaName	String	设备采用模型名称
favourite	boolean	是否收藏， 默认false

6.7.4 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，以long值表示，单位为毫秒

6.7.5 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

6.7.6 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

6.7.7 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/ thingName
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
unit	String	数据单位
defaultValue	String	数据默认值
type	String	属性数据类型, string number bool object object类型说明: object类型的数据格式与Json object相同, 其定义为: 一个无序的“‘名称/值’对”集合。一个对象以“{”(左括号)开始, “}”(右括号)结束。每个“名称”后跟一个“:”(冒号); “‘名称/值’对”之间使用“,”(逗号)分隔。格式举例如下:

SchemaProperty参数列表 正确的格式

```
{
    "key1": "value1",
    "key2": 123
}

{
    "key1": "value1",
    "key2": [1, 3, 4],      # value 可以是Array
    "key3" : {
        "key4": "value4"    # 支持嵌套
    }
}
```

错误的格式

```
[1, 3, 5]          # 直接是一个 Array, Object 类型必须在 {} 里面
"hello world"      # 直接是一个 字符串
12.34              # 直接是一个 数字
```

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 (以毫秒为单位)
lastUpdatedTime	long	模板更新时间 (以毫秒为单位)
properties	List<SchemaProperty>	模板属性列表

Schema参数列表

6.8 更新历史

[2017-12-01](#)

在原有的3种数据类型string, number, bool的基础上，增加object类型。

[2017-11-2](#)

增加物管理接入规则引擎。

6.9 API参考-V1

适用于开发人员，介绍如何通过API实现创建设备、获取设备Profile、查询设备等操作。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。

[API参考-V1](#)

6.10 API参考-V2

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）

[API参考-V2](#)

第7章 Java SDK文档

7.1 V3

7.1.1 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](#)。

7.1.2 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。

方式一：使用Maven安装

在Maven的pom.xml文件中添加bce-java-sdk的依赖：

```
<dependency>
    <groupId>com.baidubce</groupId>
    <artifactId>bce-java-sdk</artifactId>
    <version>{version}</version>
</dependency>
```

其中，`{version}`为版本号，可以在[SDK下载页面](#)找到。

方式二：直接使用JAR包安装

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

```

7.1.3 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
connectionTimeoutInMillis	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
proxyPreemptiveAuthenticationEnabled	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

7.1.4 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<DeviceViewAttribute> devices = 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, value);

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);
```

7.1.5 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
```

7.1.6 版本说明

v0.10.21 首次发布

7.2 旧版

7.2.1 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](#)。

7.2.2 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
```

7.2.3 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
connectionTimeoutInMillis	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
proxyPreemptiveAuthenticationEnabled	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

7.2.4 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);
```

7.2.5 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);
```

7.2.6 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();
```

7.2.7 版本说明

v0.10.13 首次发布。

第8章 IoT Edge SDK

为了帮助开发者能更高效地将各类设备与云端互联，并利用天工物联网平台完善的接入、存储、计算和分析能力打造物联网应用，天工物联网推出了完全开源开放的IoT Edge SDK。

IoT Edge SDK包含了物接入(IoT Hub)的C语言客户端、序列化和反序列化、设备管理、协议解析等功能组件，涵盖了实现设备上云时在断线缓存、在线检测、设备管理、数据安全传输等场景，同时更多的服务也将在未来开放。

如须下载SDK，或者了解更详细的IoT Edge SDK使用说明，请访问Github：<https://github.com/baidu/iot-edge-c-sdk>

天工物接入服务完全兼容标准MQTT协议。对于仅需将设备数据与云端进行双向通讯的用户，也可以使用众多的第三方MQTT SDK来进行开发。比如Paho MQTT SDK。

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。

9.1.3 新版物管理提供了哪些topic？

\$baidu/iot/shadow/{deviceName}/update、\$baidu/iot/shadow/{deviceName}/get、\$baidu/iot/shadow/{deviceName}/delta、\$baidu/iot/shadow/{deviceName}/delete、\$baidu/iot/shadow/{deviceName}/update/documents、\$baidu/iot/shadow/{deviceName}/update/snapshot 用户在物管理中创建设备后，系统默认提供这些topic，每个topic的具体用法请参见[文档](#)。

9.1.4 可以使用MQTT的SDK连接物管理吗？

可以，按照mqtt的连接方式，对应物管理系统提供的相应topic中发送数据就可以了。

也可以使用天工官方提供的SDK，SDK中提供了示例demo.

9.1.5 物管理如何获得设备在线状态？

如果使用MQTT开源SDK，用物影子的名称作为clientID连接，物管理则可通过该连接来判断是否在线，如果clientID不是物影子的名称，则无法判断是否在线。天工官方提供的SDK，已将该功能集成好。

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\$baidu/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_baidu_sample_pump_instance，且成功向\$baidu/iot/shadow/baidusample_pump_instancee/update发布第一条消息。

9.1.6 物接入和物管理如何选择？

物接入提供原生MQTT支持，开发者可基于MQTT数据流做物联网应用开发，数据存储TSDB需要通过规则引擎。更灵活。

物管理提供基于物提供了更好的支持，比如物影子、物模型、云端的状态暂存、直接获取在线状态、数据可以直接存储TSDB、物影子数据可以直接在物可视中展示。更便捷。

9.1.7 物影子有什么作用？

物影子是设备在云端的状态暂存。通过reported字段和desired字段来反应设备的上报值和期望值。

9.1.8 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。

9.1.9 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。

9.1.10 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。

9.1.11 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，并具有相同主题收发权限。

9.1.12 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）。