
IOT 文档

2017-07-31



百度云
cloud.baidu.com

目录

1	产品描述	1
1.1	目录	1
1.2	介绍	1
1.2.1	概述	1
1.2.2	优势	2
1.2.3	架构	2
1.2.4	核心概念	2
1.2.5	应用场景	3
1.3	MQTT协议	4
1.3.1	概述	4
1.3.2	如何使用MQTT?	4
2	产品定价	7
2.1	目录	7
2.2	按需计费	7
2.3	到期提醒和处理	8
3	入门指南	9
3.1	目录	9
3.2	概述	9
3.3	试用物接入服务	10
3.3.1	Python	10

3.3.2	NodeJS	10
3.3.3	Java	11
3.3.4	Arduino D1 & NodeMCU	13
3.3.5	Arduino Wido	13
3.3.6	Arduino Yun	14
3.3.7	MQTT.fx	15
3.4	控制台快速操作指导	21
3.4.1	注册并登录IoT Hub	21
3.4.2	创建计费套餐	21
3.4.3	创建实例	22
3.4.4	配置实例	23
3.4.5	配置MQTT客户端	25
4	操作指南	29
4.1	目录	29
4.2	概述	30
4.2.1	视频介绍	30
4.2.2	核心概念	30
4.2.3	操作流程	31
4.3	注册并登录物接入	31
4.4	创建计费套餐	32
4.4.1	升级和续费	32
	续费	33
4.5	创建物接入服务	34
4.5.1	创建物接入实例	34
4.5.2	创建物接入设备	35
4.5.3	创建物接入身份	37
4.5.4	重新生成密钥	38

4.5.5	创建物接入策略	39
	关于通配符的使用方法	40
	避免用广播方式向特定设备发送消息	41
4.6	连接测试	43
4.7	配置数据存储	45
4.8	连接实体设备	45
4.8.1	使用MQTT.fx客户端模拟实体设备	45
4.8.2	使用MQTT Client SDK模拟实体设备	47
4.8.3	持久化会话和消息的缓存	49
4.8.4	保留消息	51
4.9	多用户协作	52
5	IoT Hub CLI	55
5.1	目录	55
5.2	概述	55
5.3	系统限制	55
5.4	安装IoT Hub CLI	56
5.5	使用IoT Hub CLI	57
5.5.1	配置IoT Hub CLI	57
5.5.2	获得CLI帮助信息	57
5.6	通过CLI使用IoT Hub服务	57
5.6.1	创建IoT Hub服务	57
5.6.2	使用MQTT客户端验证IoT Hub服务	61
5.6.3	应用IoT Hub服务	63
5.7	常用操作	64
5.8	版本更新记录	67
6	API参考	69
6.0.1	目录	69

6.1	介绍	70
6.1.1	简介	70
6.1.2	调用方式	71
	概述	71
	通用约定	71
	公共请求头	71
	公共响应头	71
	响应状态码	71
	通用错误返回格式	72
	公共错误码	73
	签名认证	73
	签名生成算法	73
6.1.3	多区域选择	73
6.2	认证	73
6.2.1	认证	73
6.2.2	鉴权	74
6.3	动作	75
6.3.1	给一个Thing添加一个Principal	75
6.3.2	从一个Thing移除一个Principal	76
6.3.3	给一个Principal添加一个Policy	77
6.3.4	从一个Principal移除一个Policy	78
6.4	Endpoint	79
6.4.1	获取endpoint列表	79
6.4.2	获取指定的endpoint信息	81
6.4.3	创建endpoint	82
6.4.4	删除endpoint	83
6.5	Thing	84
6.5.1	获取thing列表	84

6.5.2	获取指定的thing信息	86
6.5.3	创建thing	87
6.5.4	删除thing	88
6.6	Principal	89
6.6.1	获取principal列表	89
6.6.2	获取指定的principal信息	90
6.6.3	创建principal	91
6.6.4	重新生成密钥	92
6.6.5	删除principal	94
6.7	Policy	95
6.7.1	获取policy列表	95
6.7.2	获取指定的policy信息	96
6.7.3	创建policy	97
6.7.4	删除policy	98
6.8	Permission	99
6.8.1	获取policy下所有topic信息	99
6.8.2	获取指定topic的信息	101
6.8.3	在policy下设置topic	102
6.8.4	更新已有的topic设置	103
6.8.5	删除已有的topic	104
6.9	Client	105
6.9.1	获取指定MQTT客户端在线状态	105
6.9.2	获取所有MQTT客户端在线状态	106
6.9.3	Publish Message	107
7	Java SDK文档	109
7.1	目录	109
7.2	概述	110

7.3	安装SDK工具包	110
7.4	快速入门	111
7.5	创建IotHubClient	111
7.6	endpoint操作	112
7.6.1	创建endpoint	112
7.6.2	获取endpoint列表	113
7.6.3	查看指定的endpoint信息	113
7.6.4	删除endpoint	113
7.7	thing操作	113
7.7.1	创建thing	113
7.7.2	查看thing	114
7.7.3	删除thing	114
7.8	principal操作	114
7.8.1	创建principal	114
7.8.2	查看principal	114
7.8.3	绑定指定的thing和principal	115
7.8.4	重新获得principal的证书和密码	115
7.8.5	删除principal	115
7.9	policy操作	115
7.9.1	创建policy	115
7.9.2	查看policy	115
7.9.3	绑定指定的principal和policy	116
7.9.4	删除policy	116
7.10	permission操作	116
7.10.1	创建permission	116
7.10.2	更新permission	117
7.10.3	查看permission	117
7.11	版本说明	117

7.11.1 v0.10.13	117
8 常见问题	119
8.1 目录	119
8.2 产品配置操作问题	120
8.2.1 物接入中是否能批量创建设备？	120
8.2.2 物接入实例列表中为什么出现了不是我自己创建的实例？	120
8.2.3 物接入的数据能存储到哪里？	120
8.2.4 物接入中数据存储主题无法填写？	120
8.3 产品规格及使用限制	120
8.3.1 物接入上传的消息大小有限制吗？	120
8.3.2 每个百度云账号可以创建多少个实例？	120
8.3.3 每个实例下可以创建多少个策略、身份和设备？	121
8.3.4 每个实例下允许的最大连接数？	121
8.3.5 每个实例下的物接入设备与实际连接数是什么关系，限额都是多少？	121
8.4 计费相关问题	121
8.4.1 物解析和物管理的实例中收发的消息数也算在物接入的计费套餐里吗？	121
8.5 客户端及MQTT SDK相关问题	121
8.5.1 请问是否有基于FreeRTOS或者RTX操作系统的MQTT SDK源码？	121
8.5.2 为什么publish的QOS等级为2后，则会断开服务？	121
8.5.3 物接入是否支持消息缓存？	122
8.5.4 与物接入服务连接成功，往一个主题发送消息，就直接断开。	122
8.5.5 连接物接入服务时，出现连接协议错误。	122
8.5.6 遗嘱消息的触发条件有哪些？	122
8.5.7 MQTT客户端网络连接异常，但在keepalive时间内恢复，客户端是否需要重新建立MQTT连接？	122
8.6 API使用问题	123
8.6.1 使用API连接物接入时，返回connection timeout错误。	123
8.6.2 使用API连接物接入时，返回invalid ClientID	123

9 MQTT协议介绍	125
9.1 目录	125
9.2 关于本章	125
9.3 MQTT协议是什么？	125
9.4 MQTT协议如何工作？	126
9.4.1 如何将消息正确送达？	126
9.4.2 如何使用通配符订阅多个主题？	127
9.4.3 如何确保消息已被送达？	128
9.4.4 什么是临终遗嘱？	128
10 MQTT客户端使用指南	129
10.1 目录	129
10.2 Websockets Client	129
10.3 MQTT.fx	129
10.3.1 连接IoT Hub服务	129
10.3.2 订阅消息	130
10.3.3 发布消息	131
11 MQTT Client SDK	133
12 MQTT客户端代码示例	135
12.1 目录	135
12.2 C代码示例	135
12.2.1 下载TLS认证文件	135
12.2.2 下载并执行示例代码	135
12.3 C#代码示例	139
12.4 python代码示例	140
12.4.1 下载TLS认证文件	140
12.4.2 下载并执行示例代码	141
12.5 Java代码示例	145

13 百度天工 - 更懂行业的物联网平台	147
13.1 目录	147
13.2 天工简介	147
13.3 天工架构	148
13.4 应用天工	149
13.4.1 风电机组运维	149
13.4.2 智慧路灯	150
13.4.3 工业协议解析	151

第1章

产品描述

1.1 目录

- [介绍](#)
 - [概述](#)
 - [优势](#)
 - [架构](#)
 - [核心概念](#)
 - [应用场景](#)
- [MQTT协议](#)
 - [概述](#)
 - [如何使用MQTT？](#)

1.2 介绍

1.2.1 概述

物接入IoT Hub 是全托管的云服务，可以在智能设备与云端之间建立安全的双向连接，并通过主流的物联网协议（如MQTT）通讯，快速实现物联网项目。

IoT Hub 服务主要具备如下功能：

- 从设备到云端以及从云端到设备安全稳定的进行大规模消息传输。
- 对设备认证与权限管理，并保证数据安全传输。
- 支持多种语言开发，兼容主流硬件设备。
- 与大数据服务无缝对接，以数据分析驱动业务进步。

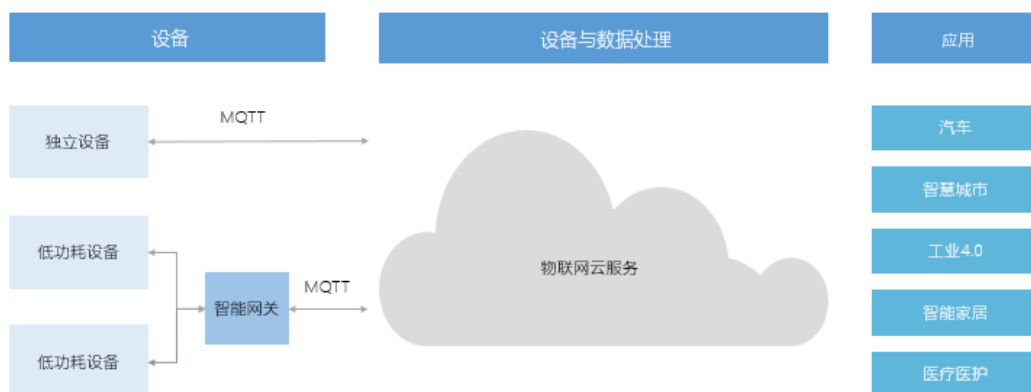
1.2.2 优势

- 独特的全托管服务支持
 - 从设备到云端以及从云端到设备安全稳定的进行大规模消息传输。
 - 与大数据服务无缝对接，以数据分析驱动业务进步。
- 多场景支持
 - 支持MQTT协议。
 - 兼容主流硬件设备。
- 多语言环境
 - 可以多种语言开发，兼容主流硬件设备。支持C、C#、Python、Java、PHP等。
 - 支持windows和linux开发环境。
- 稳定强大
 - 设备认证与权限管理，并保证数据安全传输。
 - 个别实例故障不影响整体服务。

1.2.3 架构

百度云物接入服务支持MQTT（Message Queuing Telemetry Transport）协议，使用的发布/订阅（Publish/Subscribe）模式更符合机器之间（Machine-to-Machine，M2M）的大规模沟通。

通过支持轻量级可扩展的MQTT，百度云物接入服务非常适合需要低功耗和网络带宽有限的IoT Hub 场景：



1.2.4 核心概念

概念	描述
MQTT	MQTT 是基于二进制消息的发布/订阅（Publish/Subscribe）模式的协议，最早由IBM提出的，如今已经成为OASIS规范，更符合M2M大规模沟通。
endpoint	IoT Hub 的服务实例，代表一个完整的IoT Hub 服务。
thing	表示IoT Hub 设备，用户可以在每个endpoint中创建一个或多个thing。
principal	principal是一个抽象概念，表示设备(thing)的身份。每个thing可以绑定一个principal，每个principal拥有一个policy权限。
policy	为身份principal设置对应的策略policy，一个principal对应一个policy。
permission	为每一个policy设置一组权限permission，其中包括主题topic，和对该主题的操作权限operation。
topic	每一个policy都需要指定一个主题项目（topic），在进行使用IoT Hub 服务之前，需要先为我们即将开展的订阅发布信息创建一个主题名称，该主题应用于MQTT客户端。topic规则允许字符串可以带一个通配符“#”，例如“temperature/#”就是匹配前缀是temperature的所有topic；单独的“#”表示匹配所有topic。
operation	对topic的操作权限。目前基于MQTT协议，IoT Hub 支持创建发布PUBLISH和订阅SUBSCRIBE两种权限。

1.2.5 应用场景

为了更好的推动物联网在中国的发展，百度云正式推出了物接入服务，全面助力构建物联网社会。IoT Hub 适用于五大业务场景：

- 智慧能源
- 智能硬件
- 工业4.0
- 智能家居
- 车联网

特别地，通过支持轻量级可扩展的MQTT，IoT Hub 非常适合需要低功耗和网络带宽有限的IoT Hub 场景。

1.3 MQTT协议

1.3.1 概述

MQTT (Message Queuing Telemetry Transport) 是一个客户端服务端架构的发布/订阅模式的消息传输协议。它的设计思想是轻巧、开放、简单、规范，易于实现。这些特点使得它对很多场景来说都是很好的选择，特别是对于受限的环境如机器与机器的通信（M2M）以及物联网环境（IoT）。

支持MQTT底层传输协议的相关设备有：

- 客户端-使用它连接服务端。
- 服务端-全托管的云服务，帮助建立设备与云端之间安全可靠的双向连接，以支撑海量设备的数据收集、监控、故障预测等各种物联网场景。

客户端

使用MQTT的程序或设备，推荐您使用MQTT.fx。客户端总是通过网络连接到服务端。它可以

- 发布应用消息给其它相关的客户端。
- 订阅以请求接受相关的应用消息
- 取消订阅以移除接受应用消息的请求。
- 从服务端断开连接。

服务端

全托管的云服务，帮助建立设备与云端之间安全可靠的双向连接，以支撑海量设备的数据收集、监控、故障预测等各种物联网场景。

- 接受来自客户端的网络连接
- 接受客户端发布的应用消息
- 处理客户端的订阅和取消订阅请求。
- 转发应用消息给符合条件的客户端订阅。

更详细的MQTT协议介绍请参考[MQTT官网](#)。

1.3.2 如何使用MQTT？

要使用MQTT连接，需要使用到下面三部分服务：

1.MQTT托管服务

MQTT托管服务搭建在百度云 IoT Hub 上，提供可弹性扩展，安全稳定的消息服务，轻松支持百万级消息并发。

2.MQTT客户端

MQTT客户端目前可使用社区工具或者程序库来进行连接，随后百度也会提供各类官方SDK支持。

3.IoT Hub CLI（命令行工具）

IoT Hub CLI命令行工具可进行消息队列的管理，比如创建设备、授权、消息等。

详细的操作方法请参考[入门指南](#)。

第2章

产品定价

2.1 目录

- [按需计费](#)
- [到期提醒和处理](#)

2.2 按需计费

您可以通过[物接入价格计算器](#)快速获取价格。

IoT Hub为预付费服务，实行阶梯定价，根据每月可收发的消息额度及购买时长计费。用户在购买服务前对每月发布/订阅的消息条数进行预估，选择购买相应的消息额度。

对于产品公测期间已开通免费物接入服务的用户，系统会自收费开始日起（具体日期以公告为准）创建一个3个月的免费配置（可发布、订阅100万条消息/月）订单，请在3个月内登录控制台升级或续费套餐。

购买前需保证账户无欠款。

费率表

注意

- 以下计价单位中的条数代表发布消息（PUB）和订阅消息（SUB）之和。例如：有5台设备订阅了同一个主题（topic），第6台设备向该主题发布1条消息，则总条数的计算方法为：1（PUB）+5（SUB）=6。
- 在计费上，每条消息的最大长度为512Bytes，超出部分将被算作是一条或多条新的消息，也就是“实际消息长度/512Bytes”的计算结果向上取整。在实际使用中，用户上传的单条消息大小限制是32KB，超过32KB的消息会被丢弃。
- 当月消息用量超过所购买的消息额度后，新发送的消息将不会被接收处理。

- 当月剩余消息额度不累计至次月。

规格(百万条/月)	目录价-月单价(元/百万条/月)	目录价-年单价(元/百万条/年)
0-1	免费	免费
2-5	¥1.00	¥10.00
6-10	¥0.90	¥9.00
11-50	¥0.80	¥8.00
51-100	¥0.70	¥7.00
101-500	¥0.60	¥6.00
501-1000	¥0.50	¥5.00
1000+	请提交工单申请	-

计算公式

以下通过一个实例介绍IoT Hub的费用计算方法。

以用户选择购买每月收发1亿（即100百万）条消息额度为例，用户需要支付的月费用计算公式为：

$$0\backslash 1+1\backslash 4+0.9\backslash 5+0.8\backslash 40+0.7\backslash *50=75.5\text{元/月}$$

2.3 到期提醒和处理

用量提醒：当发布和订阅的消息总数达到已购额度的75%和90%时，系统将分别发送用量提醒消息，当月消息用量超过所购买的消息额度后，新发送的消息将不会被接收处理。为避免因达到额度上限导致业务中断，请及时进行升级续费操作。

到期提醒：IoT Hub服务到期前7天，系统会通过邮件及短信给您发送即将到期提醒通知。

到期后处理：服务到期后立即停止，系统会通过邮件及短信发送欠费停机通知。数据为您保留30天，期间不收取费用，30天内未充值则释放，释放前1天和释放时系统都会发送释放通知。

第3章

入门指南

3.1 目录

- [概述](#)
- [试用物接入服务](#)
 - [Python](#)
 - [NodeJS](#)
 - [Java](#)
 - [MQTT.fx](#)
 - [Arduino D1 & NodeMCU](#)
 - [Arduino Wido](#)
 - [Arduino Yun](#)
- [控制台快速操作指导](#)
 - [注册并登录IoT Hub](#)
 - [创建计费套餐](#)
 - [创建实例](#)
 - [配置实例](#)
 - [配置MQTT客户端](#)

3.2 概述

说明

如果您还不了解MQTT协议，推荐您首先查看[MQTT协议介绍](#)，了解MQTT的工作原理。

本文档用于帮助用户试用物接入服务或快速完成物接入服务的部署，如果您想要了解更多功能，请参看[配置指南](#)。

3.3 试用物接入服务

3.3.1 Python

试用物接入服务时，用户无需登录控制台，物接入服务的云端配置已经由百度云工程师预置好，可直接使用以下代码测试消息收发。有关控制台的配置方法，请参看[控制台快速操作指导](#)。

注意事项

- 工程中所使用的物接入实例隶属于百度天工研发团队。请勿在真实产品中使用。
- MQTT协议规定两个连接的client id不能相同。否则后连的会踢走先连的。如果两个设备都带重连的话，相同的client id会导致互踢死循环。因此工程中的client id做了随机化，避免连接间互踢的情况发生。
- 本工程的用户名和密码只能访问“demoTopic”这个主题。访问其它主题会导致连接断开。

百度云工程师已在云端配置了物接入实例“iotfreetest”，并为其创建了设备“thing01”，给主题demoTopic设置了发布和订阅权限。您可以使用[MQTT客户端](#)或调用[MQTT SDK](#)进行连接测试。连接的用户名为“iotfreetest/thing01”，密码为“YU7Tov8zFW+WuaLx9s9I3MKyclie9SGDuuNkl6o9LXo”也可以参考以下步骤，运行工程代码测试消息收发。

1. 预安装环境

本工程需要使用Python 3以上版本。本工程在Windows、Linux和Mac上均可运行。

2. 下载项目文件

从<http://iot-demo.cdn.bcebos.com/SampleCode/TestMQTTPython.zip>处下载工程文件，并解压至磁盘。

3. 编译并运行

打开命令行工具，进入目录TestMQTTPython（该目录里有一个requirements.txt文件和一个server.py文件）。然后运行“pip install -r requirements.txt && python server.py”。这时在命令行就能看到工程向百度天工发送了消息，并且自己接收了该消息。（按“Ctrl + C”可以退出运行。）

4. 测试工程

打开源代码文件，我们能看到连接IoT Hub所使用的用户名和密码。打开MQTT.fx（[MQTT调试工具](#)），使用相同的用户名和密码连接百度天工。连接完毕之后，向topic “demoTopic”发送字符串。发送的字符串会显示在“步骤3”运行测试项目的命令行中。最后，发送“exit”字符串，工程会结束运行。

3.3.2 NodeJS

试用物接入服务时，用户无需登录控制台，物接入服务的云端配置已经由百度云工程师预置好，可直接使用以下代码测试消息收发。有关控制台的配置方法，请参看[控制台快速操作指](#)

导。

注意事项

- 工程中所使用的实例隶属于百度天工研发团队。请勿在真实产品中使用。
- 工程使用SSL/TLS来连接百度天工。如果不想使用SSL/TLS，把endpoint URL换成“tcp://iotfreetest.mqtt.iot.gz.baidubce.com:1883”
- MQTT协议规定两个连接的client id不能相同。否则后连的会踢走先连的。如果两个设备都带重连的话，相同的client id会导致互踢死循环。因此工程中的client id做了随机化，避免连接间互踢的情况发生。
- 本工程的用户名和密码只能访问“demoTopic”这个主题。访问其它主题会导致连接断开。

百度云工程师已在云端配置了物接入实例“iotfreetest”，并为其创建了设备“thing01”，给主题demoTopic设置了发布和订阅权限。您可以使用[MQTT客户端](#)或调用[MQTT SDK](#)进行连接测试。连接的用户名为“iotfreetest/thing01”，密码为“YU7Tov8zFW+WuaLx9s9I3MKyclie9SGDuuNkl6o9LXo”也可以参考以下步骤，运行工程代码测试消息收发。

1. 预安装环境

本工程需要使用Node.js 4以上版本。本工程在Windows、Linux和Mac上均可运行。

2. 下载项目文件

从<http://iot-demo.cdn.bcebos.com/SampleCode/TestMQTTNode.zip>处下载工程文件，并解压至磁盘。

3. 编译并运行

打开命令行工具，进入目录TestMQTTNode（该目录里有一个package.json文件和一个server.js文件）。然后运行“npm install && npm start”。这时在命令行就能看到工程向百度天工发送了消息，并且自己接收了该消息。（按“Ctrl + C”可以退出运行。）

4. 测试工程

打开源代码文件，我们能看到连接IoT Hub所使用的用户名和密码。打开MQTT.fx（[MQTT调试工具](#)），使用相同的用户名和密码连接百度天工。连接完毕之后，向topic“demoTopic”发送字符串。发送的字符串会显示在“步骤3”运行测试项目的命令行中。最后，发送“exit”字符串，工程会结束运行。

3.3.3 Java

试用物接入服务时，用户无需登录控制台，物接入服务的云端配置已经由百度云工程师预置好，可直接使用以下代码测试消息收发。有关控制台的配置方法，请参看[控制台快速操作指导](#)。

注意事项

- 工程中所使用的实例隶属于百度天工研发团队。请勿在真实产品中使用。
- 工程使用SSL/TLS来连接百度天工。如果不想使用SSL/TLS，把endpoint URL换成“tcp://iotfreetest.mqtt.iot.gz.baidubce.com:1883”
- MQTT协议规定两个连接的client id不能相同。否则后连的会踢走先连的。如果两个设备都带重连的话，相同的client id会导致互踢死循环。因此工程中的client id做了随机化，避免连接间互踢的情况发生。
- 本工程的用户名和密码只能访问“demoTopic”这个主题。访问其它主题会导致连接断开。

百度云工程师已在云端配置了物接入实例“iotfreetest”，并为其创建了设备“thing01”，给主题demoTopic设置了发布和订阅权限。您可以使用[MQTT客户端](#)或调用[MQTT SDK](#)进行连接测试。连接的用户名为“iotfreetest/thing01”，密码为“YU7Tov8zFW+WuaLx9s9I3MKyclie9SGDuuNkl6o9LXo”，也可以参考以下步骤，运行工程代码测试消息收发。

1. 预安装环境

本工程需要使用JDK 1.8和Gradle 3.3。本工程在Windows、Linux和Mac上均可运行。

2. 下载项目文件

从<http://iot-demo.cdn.bcebos.com/SampleCode/TestMQTTJava.zip>处下载工程文件，并解压至磁盘。

您也可以通过导入Maven工程的方式测试消息收发，下载地址是<http://iot-demo.cdn.bcebos.com/SampleCode/TestMQTTJavaMaven.zip>。

3. 编译并运行

1. 命令行运行

打开命令行工具，进入目录TestMQTTJava（该目录里有一个gradle.build文件和一个src文件夹）。然后运行“gradle build && gradle run”。这时在命令行就能看到工程向百度天工发送了消息，并且自己接收了该消息。（按“Ctrl + C”可以退出运行。）

2. 使用Eclipse运行

打开Eclipse。单击菜单“File”->“Import...”。然后在Import对话框中选择“Gradle”->“Gradle Project”，然后单击两次“下一步”。再选择“TestMQTTJava”目录为工程的根目录，单击“Finish”即可。最后，单击“Run”->“Run”运行工程。如果在Run的时候弹出对话框选择运行类型，请选择“Application”。

4. 测试工程

打开源代码文件，我们可以看到连接IoT Hub所使用的用户名和密码。打开MQTT.fx（[MQTT 调试工具](#)），使用相同的用户名和密码连接百度天工。连接完毕之后，向topic“demoTopic”发送字符串。发送的字符串会显示在“步骤3”运行测试项目的命令行中。最后，发送“exit”字符串，工程会结束运行。

3.3.4 Arduino D1 & NodeMCU

Arduino平台是非常方便的开源硬件平台。本文介绍如何使用Arduino D1和NodeMCU开发板来连接百度天工物接入服务。Arduino D1和NodeMCU都使用了一块价廉物美的Wifi芯片，ESP8266。使用ESP8266，可以让开发板非常方便的通过Wifi来连接百度天工物接入服务。

注意事项

- 工程中所使用的实例隶属于百度天工研发团队。请勿在真实产品中使用。
- 工程使用SSL/TLS来连接百度天工。如果不想使用SSL/TLS，可以把WiFiClientSecure换成WiFiClient，且把端口1884换成1883。
- MQTT协议规定两个连接的client id不能相同。否则后连的会踢走先连的。如果两个设备都带重连的话，相同的client id会导致互踢死循环。因此工程中的client id做了随机化，避免连接间互踢的情况发生。
- 本工程的用户名和密码只能访问“demoTopic”这个主题。访问其它主题会导致连接断开。

1. 准备环境

1. 下载Arduino IDE并且安装对Arduino D1和NodeMCU的支持。
2. 使用USB线把开发板接入PC，并且在IDE中选择开发板的型号。
3. 在Arduino IDE中，点击“Sketch” -> “Include Library” -> “Manage Libraries...”，安装“ESP8266Wifi”，“PubSubClient”和“TaskScheduler”三个库。

2. 下载代码至开发板

1. 从<http://iot-demo.cdn.bcebos.com/SampleCode/TestESP8266.zip>处下载代码，并且解压至本地磁盘。
2. 使用Arduino IDE打开“TestESP8266.ino”文件，修改“WIFI_SSID”和“WIFI_PASSWORD”两个变量的值，把这两个变量设置成开发板能访问的Wifi的名称和密码。
3. 点击“Upload”按钮，编译并上传代码至开发板。

3. 运行并查看结果

打开Arduino IDE，单击“Tools” -> “Serial Monitor”。观察输出的字符，可以看到开发板连接上了Wifi，然后连上了天工物接入服务，然后开始打印收到的字符串。

3.3.5 Arduino Wido

Arduino平台是非常方便的开源硬件平台。本文介绍如何使用Arduino Wido开发板来连接百度天工物接入服务。Arduino Wido使用了CC3000这款Wifi芯片。使用CC3000，可以让开发板非常方便的通过Wifi来连接百度天工物接入服务。

注意事项

- 工程中所使用的实例隶属于百度天工研发团队。请勿在真实产品中使用。
- 由于CC3300不支持SSL/TLS，因此只能使用TCP来连接天工物接入服务。
- MQTT协议规定两个连接的client id不能相同。否则后连的会踢走先连的。如果两个设备都带重连的话，相同的client id会导致互踢死循环。因此工程中的client id做了随机化，避免连接间互踢的情况发生。
- 本工程的用户名和密码只能访问“demoTopic”这个主题。访问其它主题会导致连接断开。

1. 准备环境

1. 下载Arduino IDE。
2. 使用USB线把开发板接入PC，并且在IDE中选择开发板的型号“Arduino Leonardo”。
3. 在Arduino IDE中，点击“Sketch” -> “Include Library” -> “Manage Libraries...”，安装“Adafruit_CC3000”，“PubSubClient”和“TaskScheduler”三个库。

2. 下载代码至开发板

1. 从<http://iot-demo.cdn.bcebos.com/SampleCode/TestWido.zip>处下载代码，并且解压至本地磁盘。
2. 使用Arduino IDE打开“TestWido.ino”文件，修改“WIFI_SSID”和“WIFI_PASSWORD”两个变量的值，把这两个变量设置成开发板能访问的Wifi的名称和密码。
3. 点击“Upload”按钮，编译并上传代码至开发板。

3. 运行并查看结果

打开Arduino IDE，单击“Tools” -> “Serial Monitor”。观察输出的字符，可以看到开发板连接上了Wifi，然后连上了天工物接入服务，然后开始打印收到的字符串。

3.3.6 Arduino Yun

Arduino平台是非常方便的开源硬件平台。本文介绍如何使用Arduino Yun开发板来连接百度天工物接入服务。Arduino Yun自带Wifi和Ethernet模块。本文介绍在Arduino Yun上通过Wifi来连接百度天工物接入服务。

注意事项

- 工程中所使用的实例隶属于百度天工研发团队。请勿在真实产品中使用。
- 由于Arduino Yun不支持SSL/TLS，因此只能使用TCP来连接天工物接入服务。
- MQTT协议规定两个连接的client id不能相同。否则后连的会踢走先连的。如果两个设备都带重连的话，相同的client id会导致互踢死循环。因此工程中的client id做了随机化，避免连接间互踢的情况发生。

- 本工程的用户名和密码只能访问“demoTopic”这个主题。访问其它主题会导致连接断开。

1. 准备环境

1. 下载Arduino IDE。
2. 使用USB线把开发板接入PC，并且在IDE中选择开发板的型号“Arduino Yun”。
3. 在Arduino IDE中，点击“Sketch”->“Include Library”->“Manage Libraries...”，安装“PubSubClient”和“TaskScheduler”两个库。
4. 复位Arduino Yun的Wifi，并设置其连接上本地的Wifi连接。

2. 下载代码至开发板

1. 从<http://iot-demo.cdn.bcebos.com/SampleCode/TestYun.zip>处下载代码，并且解压至本地磁盘。
2. 使用Arduino IDE打开“TestYun.ino”文件，点击“Upload”按钮，编译并上传代码至开发板。

3. 运行并查看结果

打开Arduino IDE，单击“Tools”->“Serial Monitor”。观察输出的字符，可以看到开发板连接上了Wifi，然后连上了天工物接入服务，然后开始打印收到的字符串。

3.3.7 MQTT.fx

MQTT.fx简介

MQTT.fx是一款免费的MQTT测试工具，我们可以使用它来连接百度天工的物接入服务并测试消息收发。

下载MQTT.fx

- 源站：“<http://www.jensd.de/apps/mqttfx/1.3.1/>”
- 国内：“<http://mqttfx.bceapp.com/>”

连接百度天工平台的物接入服务

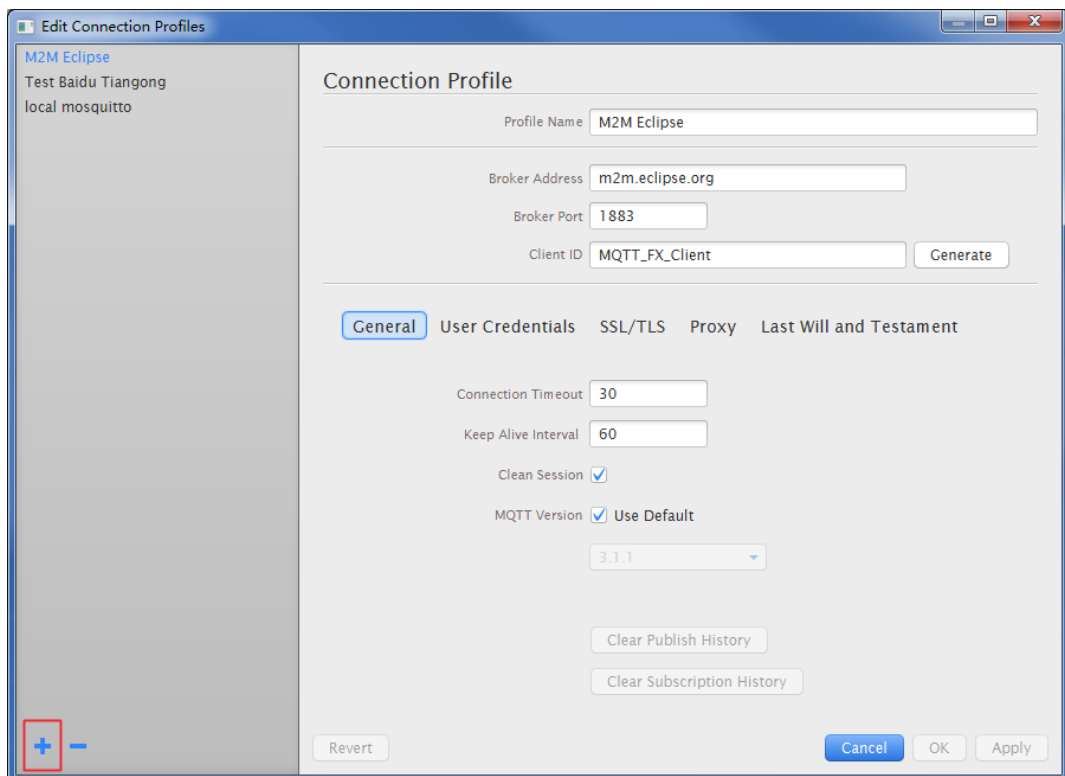
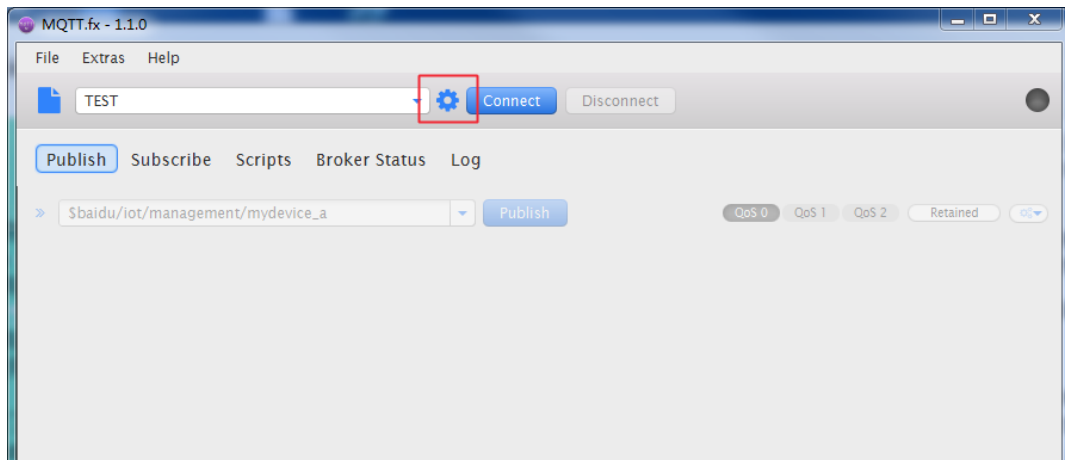
注意事项

- Client Id不可以重复，因此请保证足够随机。
- 本文使用SSL/TLS连接物接入。如果不想使用SSL/TLS，请把端口改成1883并且取消对“Enable SSL/TLS”的勾选。

- 本文的用户名和密码只有访问“demoTopic”这一个主题的权限。访问其它主题会导致连接断开。
- 本文的用户名和密码所有权属于百度天工研发团队，仅供冒烟测试使用。请不要在真实产品中使用。

百度云工程师已在云端配置了物接入实例“iotfreetest”，并为其创建了设备“thing01”，给主题demoTopic设置了发布和订阅权限。连接的用户名为“iotfreetest/thing01”，密码为“YU7Tov8zFW+WuaLx9s9I3MKyclie9SGDuuNkl6o9LXo=”。

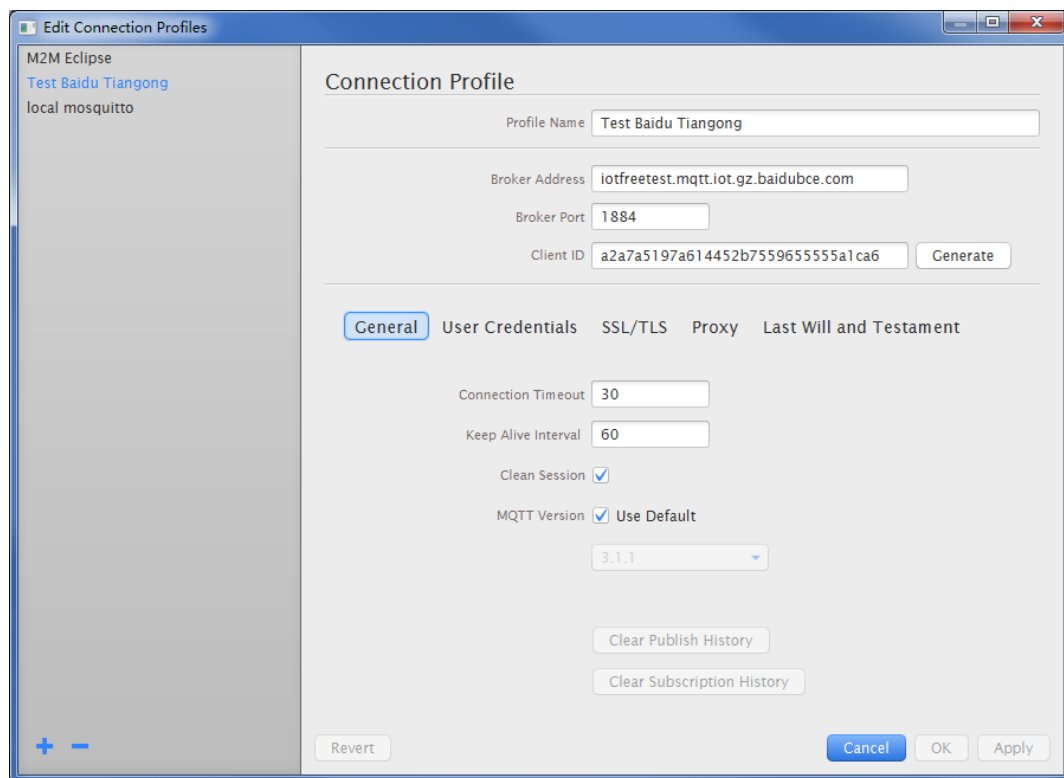
1. 打开MQTT.fx，单击“设置”图标。然后单击弹出的对话框的左下角的“添加”图标。



2. 按照下面的设置填写相应字段：

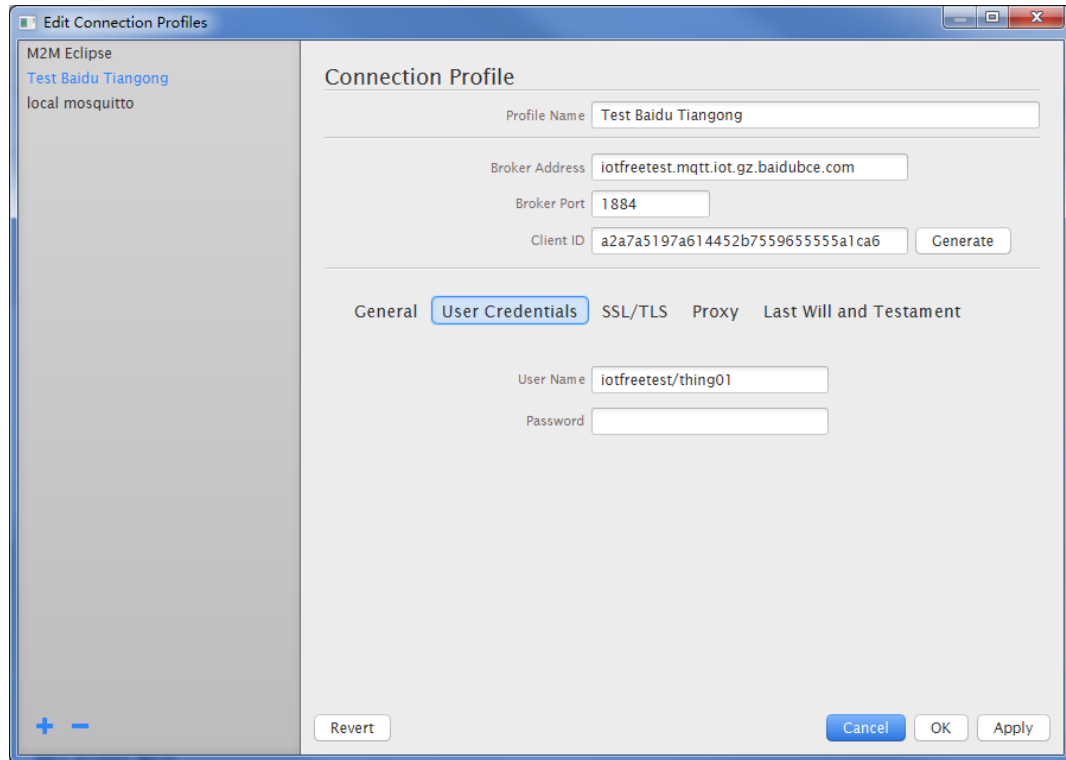
- Profile Name: Test Baidu Tiangong
- Broker Address: iotfreetest.mqtt.iot.gz.baidubce.com
- Broker Port: 1884
- Client ID: MQTT\FX\Client_81923749

注意，由于同一个Endpoint的Client Id不允许重复，因此上面的Client Id请确保足够随机。

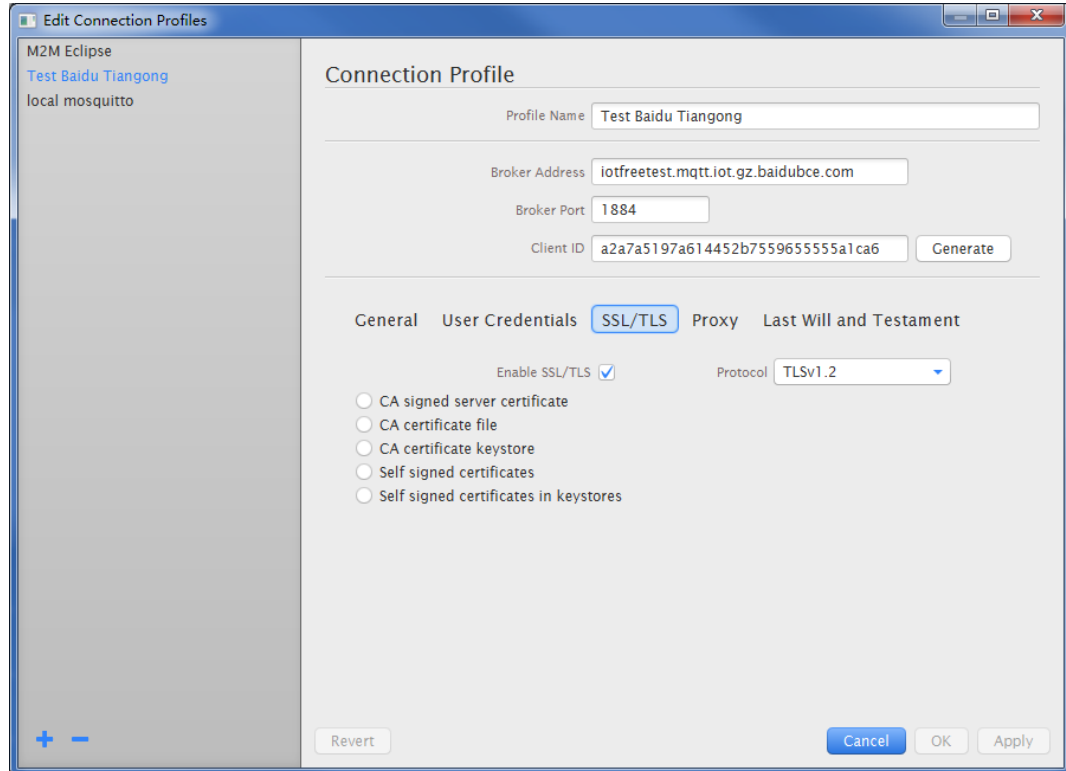


3. 选择“User Credentials”选项卡，并且按以下字段填写：

- User Name: iotfreetest/thing01
- Password: YU7Tov8zFW+WuaLx9s9I3MKyclie9SGDuuNkl6o9LXo=

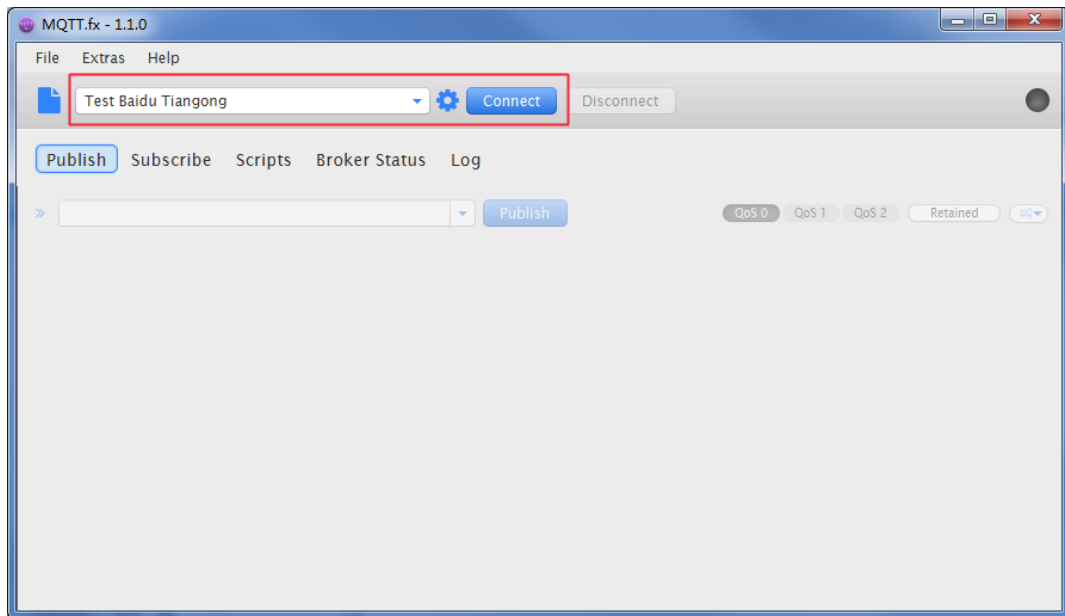


4. 选择“SSL/TLS”选项卡，勾选“Enable SSL/TLS”，并选择“CA signed server certificate”。

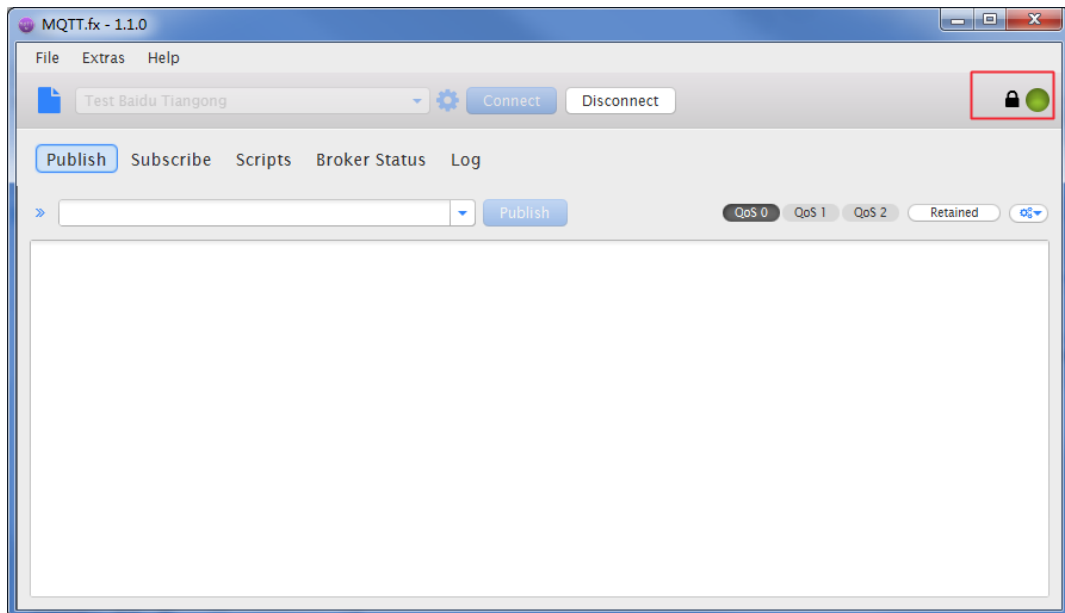


单击“OK”按钮保存设置。

5. 回到主对话框，确保选择了“Test Baidu Tiangong”，然后单击“Connect”按钮。

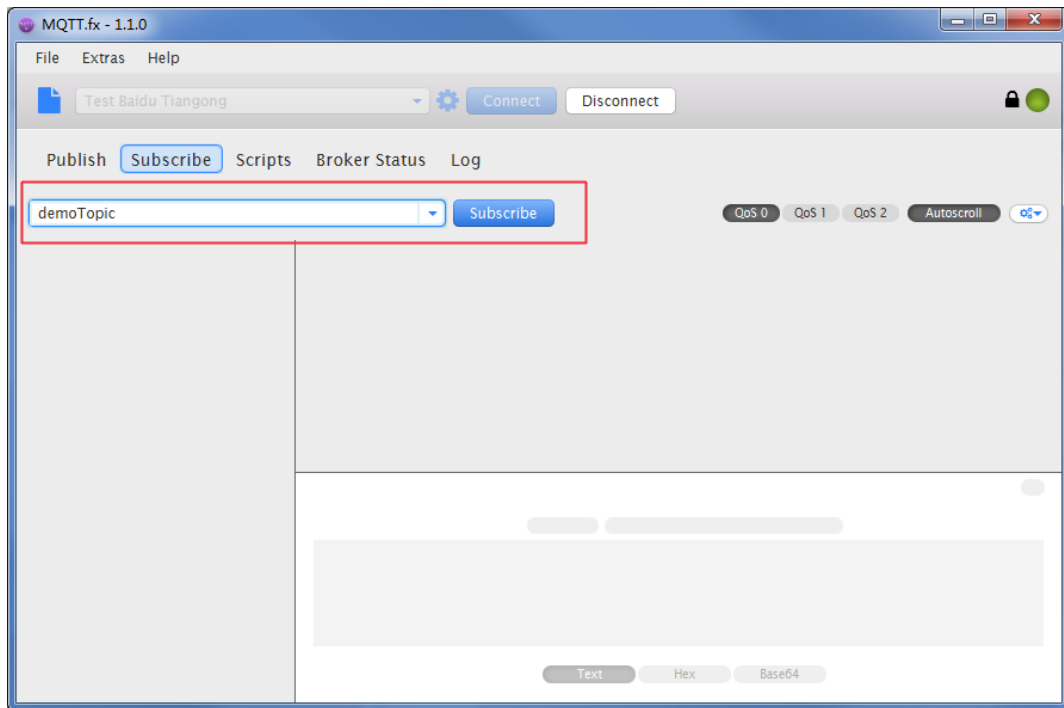


连接状态变绿，说明连接上了百度天工物联网平台的物接入服务。

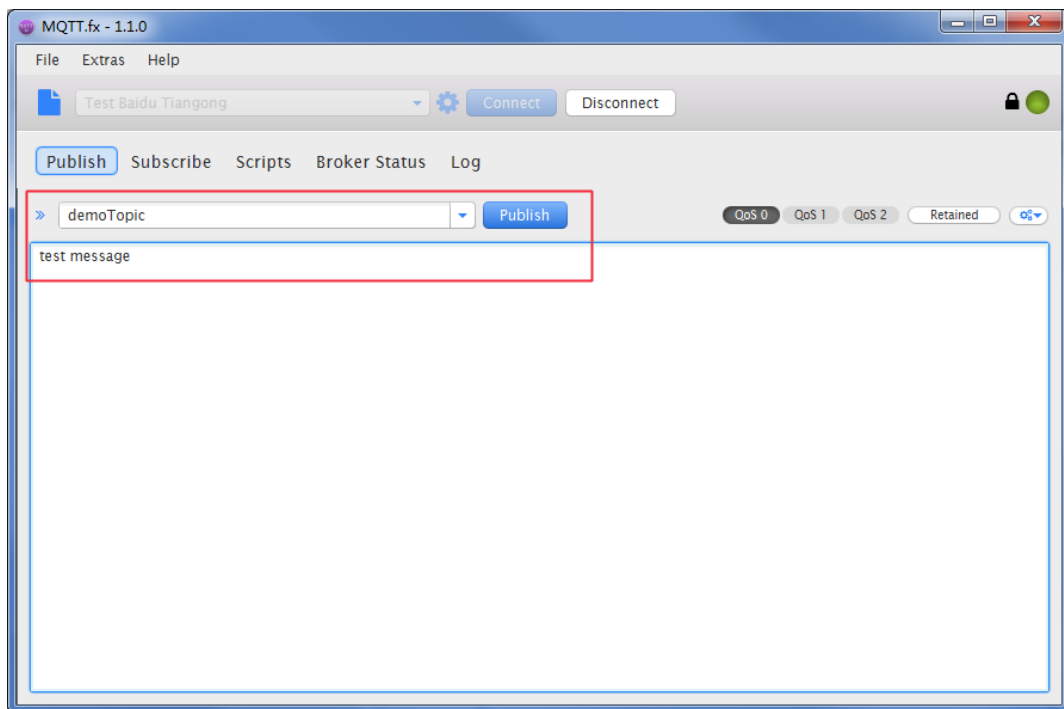


4. 测试消息收发

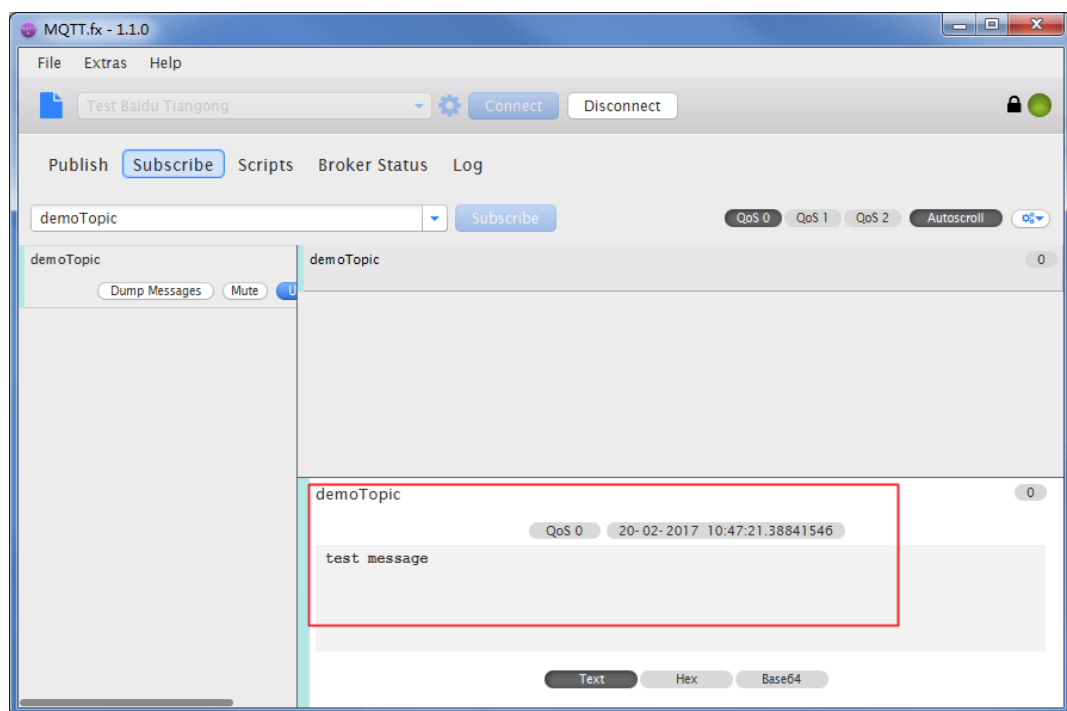
单击“Subscribe”选项卡，然后在编辑框中输入demoTopic，最后单击“Subscribe”按钮。“Subscribe”按钮变灰表示监听“demoTopic”这个主题成功。



单击“Publish”选项卡，向demoTopic发送消息。



单击“Subscribe”选项卡，查看是否收到了消息。



右下方的列表里出现了内容，则表示收到了消息。

3.4 控制台快速操作指导

3.4.1 注册并登录IoT Hub

在使用IoT Hub服务前，您需要创建一个百度云账号，请按照下述步骤进行注册和登录。

1. 注册并登录百度云平台，请参考[注册](#)和[登录](#)。
2. 如果未进行实名认证，请参考[实名认证](#)操作方法完成认证。
3. 登录成功后，导航栏选择“产品服务>物接入IOT Hub”，即可开始使用物接入服务。

3.4.2 创建计费套餐

注意

对于产品公测期间已开通免费物接入服务的用户，系统会自收费开始日起（具体日期以公告为准）创建一个3个月的免费配置（可发布、订阅100万条消息/月）订单，请在3个月内登录控制台升级或续费套餐。

在[创建实例](#)之前应先创建计费套餐并设定每个月收发消息的额度，系统将根据额度自动计算每个月的服务费用。每个用户只能创建一个计费套餐，所有实例将共享该套餐的额度。

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>物接入IoT Hub”，进入服务页面。
3. 点击“创建计费套餐”，在配置页面中设定“购买规格”和“购买时长”。物接入采用阶梯计价方式，系统将根据“购买规格”和“购买时长”自动计算服务费用。关于产品的定价和费用计算方法，请参看[产品计价](#)。

注意

- “购买规格”中的条数代表发布消息（PUB）和订阅消息（SUB）之和。例如：有5台设备订阅了同一个主题（topic），第6台设备向该主题发布1条消息，则总条数的计算方法为：1（PUB）+5（SUB）=6。
- 每条消息的最大长度为512Bytes，超出部分将被算作是一条或多条新的消息，也就是“实际消息长度/512Bytes”的计算结果向上取整。
- 当月消息用量超过所购买的消息额度后，新发送的消息将不会被接收处理。
- 当月剩余消息额度不累计至次月。

完成配置后，点击“下一步”进入在线支付页面进行支付。支付成功后，用户可进入“实例列表”，创建物接入实例。

3.4.3 创建实例

连接IoT Hub服务需要创建一个实例(endpoint)，一个endpoint表示一个完整的IoT Hub服务。登录IoT Hub控制台页面，点击“创建实例”，填写需要创建IoT Hub服务的实例名称。

说明：

- 目前每个账户能创建100个endpoint，且每个实例的名称是全局唯一的，不能重名。每个endpoint下可创建10000个thing、10000个policy和10000个principal。如果需要更多配额，请提交工单申请。
- 您也可以使用IoT Hub CLI命令[create-endpoint](#)创建实例，参考[IoT CLI](#)。

创建实例时，IoT Hub默认提供三种地址，选择不同的地址，意味着您可以通过不同的方式连接到百度云IoT Hub。

- tcp://yourendpoint.mqtt.iot.gz.baiduce.com:1883，端口1883，不支持传输数据加密，可以通过MQTT.fx客户端连接。
- ssl://yourendpoint.mqtt.iot.gz.baiduce.com:1884，端口1884，支持SSL/TLS加密传输，MQTT.fx客户端连接，参考[配置MQTT客户端](#)。
- wss://yourendpoint.mqtt.iot.gz.baidubce.com:8884，端口8884，支持[Websockets](#)浏览器方式连接，同样包含ssl加密，参考[Websockets Client](#)。

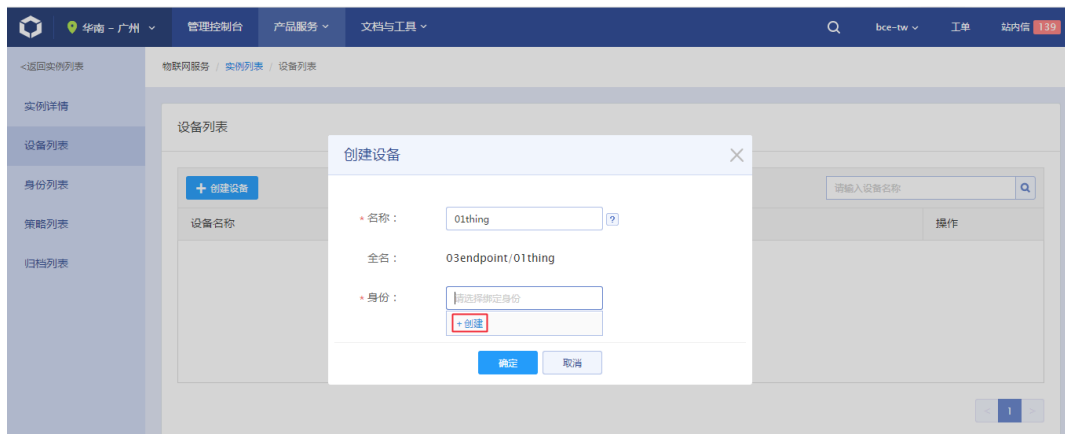
3.4.4 配置实例

成功创建IoT Hub实例后，点击实例名称，进入配置IoT Hub实例页面，创建设备、身份和策略。具体操作步骤如下：

注意

每个实例下可创建10000个设备、10000个策略和10000个身份。如果需要更多配额，请提交工单申请。

1. 创建设备：选择设备列表，点击“创建设备”，输入需要连接IoT Hub服务的设备名称。为了便于灵活连接IoT Hub服务，每一台设备thing都需要绑定相应的身份principal。



2. 创建身份：在身份输入框内点击“创建”，快速创建身份principal。输入principal名称，以及为每个身份授权的策略policy。



3. 创建策略：在策略输入框内点击“创建”，快速创建策略policy。输入policy名称，主题topic，选择该身份拥有的权限：发布消息(publish)、订阅消息(subscribe)。

创建策略

* 名称 :

01policy

?

* 主题 :

test-iot-service

?

* 权限 :

☒ 发布(PUB)

☒ 订阅(SUB)

注 : 点击右侧加 (+) 号为本策略绑定更多的主题。

确定

取消

说明:

每个policy可以创建多个主题topic，在创建策略弹框右侧，点击“+”可以绑定更多的主题。

创建策略

* 名称 :

01policy

?

* 主题 :

test-iot-service

?

* 权限 :

☒ 发布(PUB)

☒ 订阅(SUB)

* 主题 :

add-new-topic

?

* 权限 :

☐ 发布(PUB)

☒ 订阅(SUB)

注 : 点击右侧加 (+) 号为本策略绑定更多的主题。

确定

取消

- 成功创建身份principal后，返回密码，在配置客户端时会用到，需要您复制保存，如下图所示：



5. 至此，成功配置了 IoT Hub 实例中的相关参数。

同时，也可以使用IoT Hub CLI命令创建各项参数，详细过程请参考[IoT Hub CLI](#)。

3.4.5 配置MQTT客户端

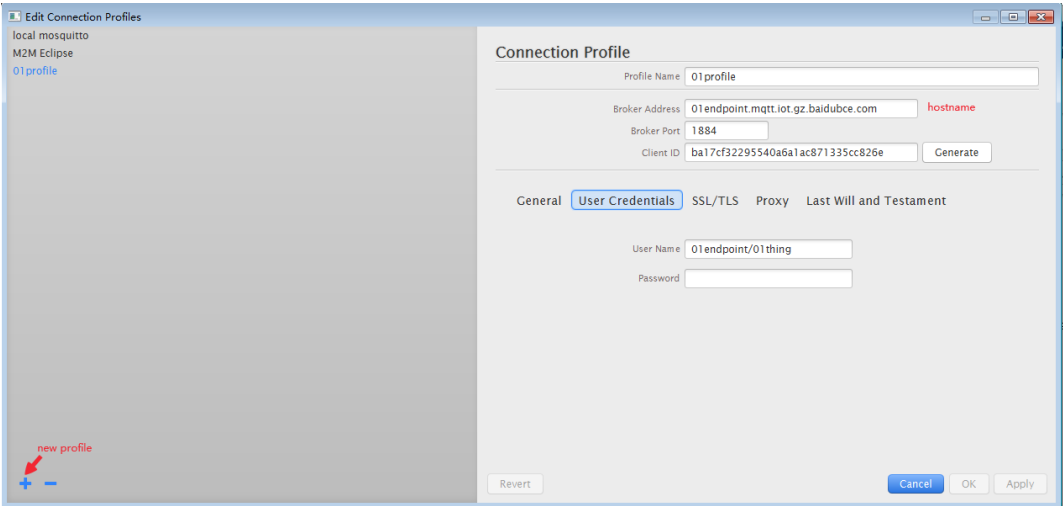
配置MQTT的应用客户端，可以快速验证是否可以实现与物接入服务交流发送或者接收消息。

前提条件

登录[MQTT.fx官网](#)，找到适合的版本下载并安装MQTT.fx客户端。

操作步骤

1. 打开MQTT客户端的设置页面，点击“+”按键，创建一个新的配置文件。



* 填写Connection profile相关信息：

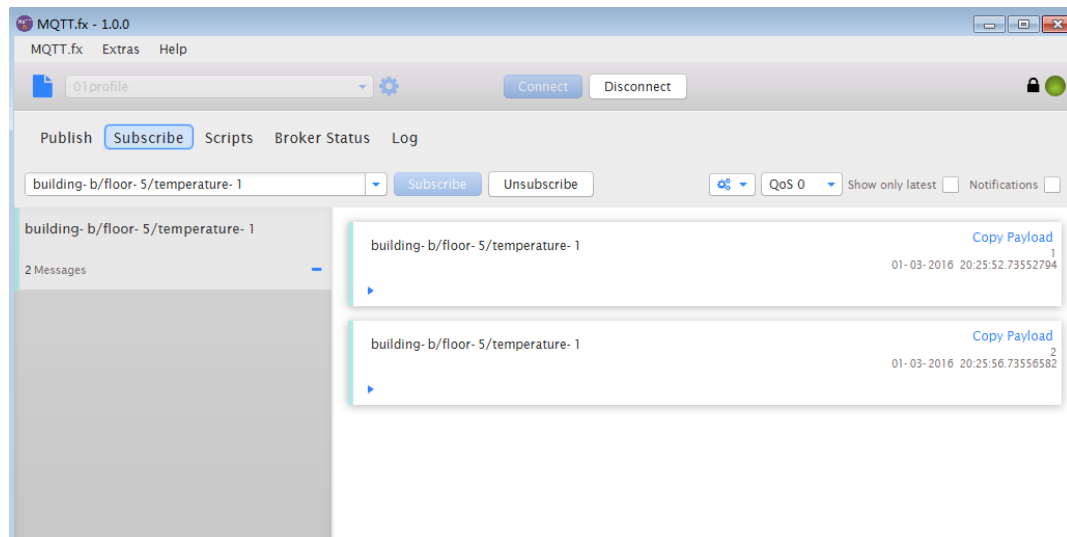
参数名称	说明
profile name	配置文件名称
Broker Address	创建endpoint后返回的hostname
Broker Port	1884
Client ID	客户端ID，支持“a-z”，“0-9”，“_”，“-”字符，且不能大于128bytes，UTF8编码

* 选择User Credential，输入创建 IoT Hub 服务返回的username/password，参考[配置实例] (<https://cloud.baidu.com/doc/IOT/Quickstart-new.html#.E9.85.8D.E7.BD.AE.E5.AE.9E.E4.BE.8B>)。

* 配置SSL/TLS安全认证，勾选 `Enable SSL/TLS`，选择`CA signed server certificate`认证。

点击“Apply”按键，完成客户端配置。

2. 返回MQTT客户端界面，选择新创建的配置文件，点击“connect”按键连接服务。
3. 成功连接后，即可开始订阅消息。
打开Subscribe标签，填写主题topic，例如building-b/floor-5/temperature-1，选择默认的QoS 0，点击“Subscribe”进行订阅操作。
4. 发布消息。
打开Publish标签，填写主题topic，例如building-b/floor-5/temperature-1，选择默认的QoS 0，点击“Publish”进行发布操作。
5. 返回Subscribe界面，即可看到已接收的订阅消息，参见下图。



第4章

操作指南

4.1 目录

- 概述
 - 视频介绍
 - 核心概念
 - 操作流程
- 注册并登录物接入
- 创建计费套餐
 - 升级和续费
 - * 升级
 - * 续费
- 创建物接入服务
 - 创建物接入实例
 - 创建物接入设备
 - 创建物接入身份
 - 重新生成密钥
 - 创建物接入策略
 - * 关于通配符的使用方法
 - * 避免用广播方式向特定设备发送消息
- 连接测试
- 配置数据存储
- 连接实体设备
 - 使用MQTT.fx客户端模拟实体设备
 - 使用MQTT Client SDK模拟实体设备
 - 持久化会话和消息的缓存
 - 保留消息
- 多用户协作

4.2 概述

4.2.1 视频介绍

4.2.2 核心概念

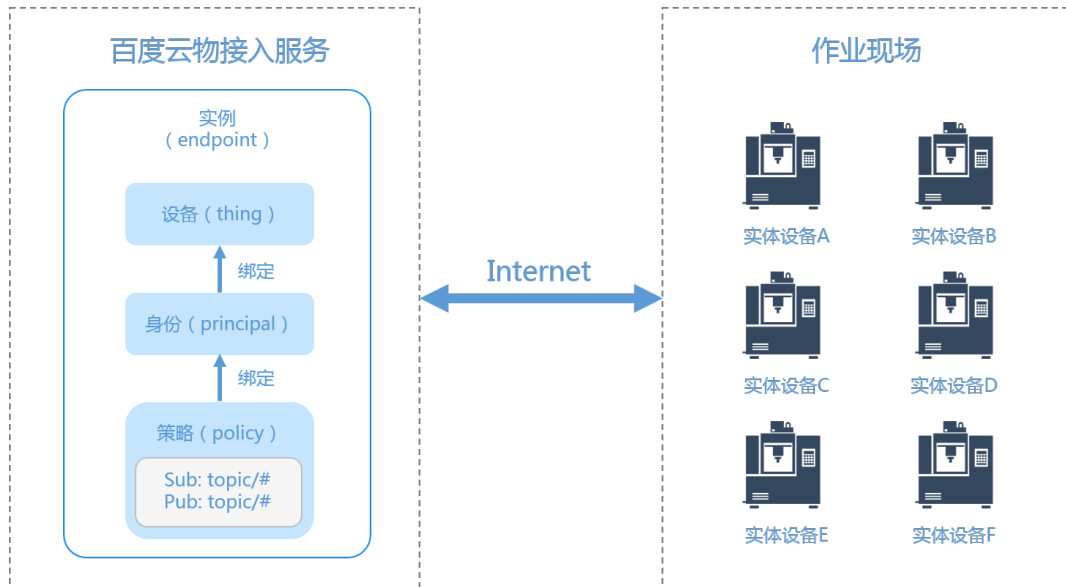
百度云物接入服务（IoT Hub）是全托管的云服务，可以在智能设备与百度云端之间建立安全的双向连接，并通过主流的物联网协议（如MQTT）通讯，快速实现物联网项目。

说明

如果您还不了解MQTT协议，推荐您首先查看[MQTT协议介绍](#)，了解MQTT的工作原理。

- 物接入实例（endpoint）：简称“实例”，代表一个完整的物接入服务。一个百度云账号可以创建100个实例。
- 物接入设备（thing）：简称“设备”，在实例中创建的虚拟设备，每个实例下最多可以创建10000个设备。
- 物接入身份（principal）：简称“身份”，在实例中创建的虚拟设备的身份，每个设备可以绑定一个身份。每个实例下最多可以创建10000个身份。
- 物接入策略（policy）：简称“策略”，策略中定义了关于特定主题的收发权限，每个身份可以绑定一个策略。每个实例下最多可以创建10000个策略。
- MQTT主题（topic）：简称“主题”，每个策略都需要指定主题及主题对应的权限。该主题应用于MQTT客户端。物接入允许主题中带一个通配符“#”，例如“temperature/#”就是匹配前缀是temperature/的所有topic；单独的“#”表示匹配所有topic。
- 实体设备：通过物接入连接的实体设备。

有关实例、设备、身份、策略和实体设备的关系，请参看下图：



4.2.3 操作流程

物接入服务配置流程如下图所示：



1. **注册并登录物接入**：在使用物接入服务前，您需要创建一个百度云账号。
2. **创建计费套餐**：首次使用物接入应先创建计费套餐并设定每个月收发消息的额度，系统将根据额度自动计算每个月的服务费用。
3. **创建物接入服务**：在云端对物接入服务进行配置，每个物接入服务应包含以下配置项：
 - **创建物接入实例**
 - **创建物接入设备**
 - **创建物接入身份**
 - **创建物接入策略**
4. **连接实体设备**：通过MQTT客户端模拟实体设备，快速验证是否可以实现消息发送和接收。

4.3 注册并登录物接入

在使用物接入服务前，您需要创建一个百度云账号，请按照下述步骤进行注册和登录。

1. 注册并登录百度云平台，请参考[注册](#)和[登录](#)。
2. 如果未进行实名认证，请参考[实名认证](#)操作方法完成认证。
3. 登录成功后，导航栏选择“产品服务>物接入IOT Hub”，即可开始创建实例。

4.4 创建计费套餐

注意

对于产品公测期间已开通免费物接入服务的用户，系统会自收费开始日起（具体日期以公告为准）创建一个3个月的免费配置（可发布、订阅100万条消息/月）订单，请在3个月内登录控制台升级或续费套餐。

在[创建物接入实例](#)之前应先创建计费套餐并设定每个月收发消息的额度，系统将根据额度自动计算每个月的服务费用。每个用户只能创建一个计费套餐，所有实例将共享该套餐的额度。

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>物接入IoT Hub”，进入服务页面。
3. 点击“创建计费套餐”，在配置页面中设定“购买规格”和“购买时长”。物接入采用阶梯计价方式，系统将根据“购买规格”和“购买时长”自动计算服务费用。关于产品的定价和费用计算方法，请参看[产品计价](#)。

注意

- “购买规格”中的条数代表发布消息（PUB）和订阅消息（SUB）之和。例如：有5台设备订阅了同一个主题（topic），第6台设备向该主题发布1条消息，则总条数的计算方法为： $1（PUB）+5（SUB）=6$ 。
- 每条消息的最大长度为512Bytes，超出部分将被算作是一条或多条新的消息，也就是“实际消息长度/512Bytes”的计算结果向上取整。
- 当月消息用量超过所购买的消息额度后，新发送的消息将不会被接收处理。
- 当月剩余消息额度不累计至次月。

完成配置后，点击“下一步”进入在线支付页面进行支付。支付成功后，用户可进入“实例列表”，创建物接入实例。

4.4.1 升级和续费

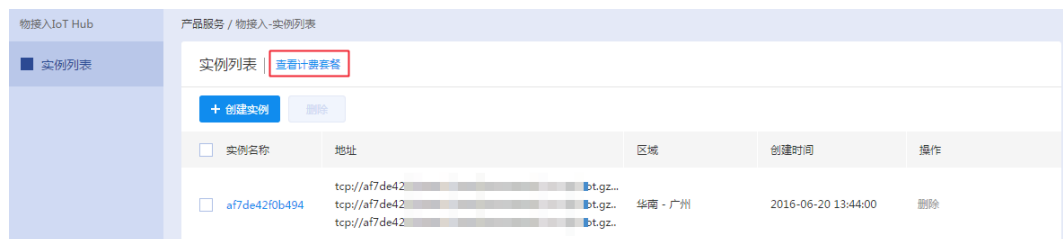
升级和续费操作仅针对计费套餐，每个用户只能创建一个计费套餐，所有实例将共享该套餐的额度。

注意：

不支持降级操作，如果用户需要降低套餐额度，可在当前套餐过期后重新创建套餐。

如果套餐额度不足，可选择对套餐配置进行升级。用户可在计费套餐详情中获取本月账期内剩余额度。当发布和订阅的消息总数达到已购额度的75%和90%时，系统将分别发送用量提醒消息，当月消息用量超过所购买的消息额度后，新发送的消息将不会被接收处理。为避免因达到额度上限导致业务中断，请及时进行升级配置操作。

1. 选择“产品服务>物接入IoT Hub”，进入实例列表。
2. 点击“查看计费套餐”，进入计费套餐详情页面。



3. 点击“配置升级”，在配置升级页面中重新提交购买规格，点击“下一步”进入在线支付页面进行支付。

升级时仅需支付差价，即：当月剩余天数 / 30 * 阶梯价格。支付成功后，用户当前剩余额度为：新套餐额度 - 当前已用额度。



续费 IoT Hub服务到期前7天，系统会通过邮件及短信给您发送即将到期提醒通知。为了避免服务中断，请在服务到期前进行续费。服务到期后立即停止，系统会通过邮件及短信发送欠费停服通知。数据为您保留30天，期间不收取费用，30天内未充值则释放，释放前1天和释放时系统都会发送释放通知。

1. 选择“产品服务>物接入IoT Hub”，进入实例列表。
2. 点击“查看计费套餐”，进入计费套餐详情页面。



3. 点击“续费”，在续费页面中输入续费时长，点击“下一步”进入在线支付页面进行支付。



4.5 创建物接入服务

4.5.1 创建物接入实例

说明：

- 目前每个账户能创建100个endpoint，且每个实例的名称是全局唯一的，不能重名。每个endpoint下可创建10000个thing、10000个policy和10000个principal。如果需要更多配额，请提交工单申请。
- 您也可以使用IoT Hub CLI命令`create-endpoint`创建实例，参考[IoT CLI](#)。

连接物接入服务时需要首先创建一个实例，具体操作方法如下：

1. 选择“产品服务>物接入IoT Hub”，进入实例列表。
2. 点击“创建实例”，填写实例名称，点击“确定”完成实例创建。



3. 返回实例列表，查看已经创建的物接入实例：



物接入默认提供三种地址，选择不同的地址，意味着您可以通过不同的方式连接到百度云物接入服务。每类地址的具体用法如下：

- * tcp://youreendpoint.mqtt.iot.gz.baiduce.com:1883，端口1883，不支持传输数据加密。
- * ssl://youreendpoint.mqtt.iot.gz.baiduce.com:1884，端口1884，支持SSL/TLS加密传输。可使用MQTT.fx客户端连接，参考[配置MQTT客户端](<https://cloud.baidu.com/doc/IOT/GettingStarted.html#.E9.85.8D.E7.BD.AEMQTT.E5.AE.A2.E6.88.B7.E7.AB.AF>)。
- * wss://youreendpoint.mqtt.iot.gz.baidubce.com:8884，端口8884，支持[Websockets](<http://websocketshow.fc.bdybsite.com/>)浏览器方式连接，同样包含ssl加密，参考[Websockets Client](<https://cloud.baidu.com/doc/IOT/Mqttclient.html#Websockets.20Client>)。

4.5.2 创建物接入设备

创建物接入设备前，应先完成[创建物接入实例](#)操作。

1. 选择“产品服务>物接入IoT Hub”，进入实例列表。
2. 点击实例名称，进入物接入实例页面。
3. 点击“创建设备”，在弹出窗口中输入设备名称，并点击“下一步”。
“实例名称/设备名称”将作为实体设备连接云端的用户名。

4. 为设备绑定身份。此处可以直接从下拉菜单中选择身份名称，如果没有可用身份，也可以选择“+创建”，创建新身份。



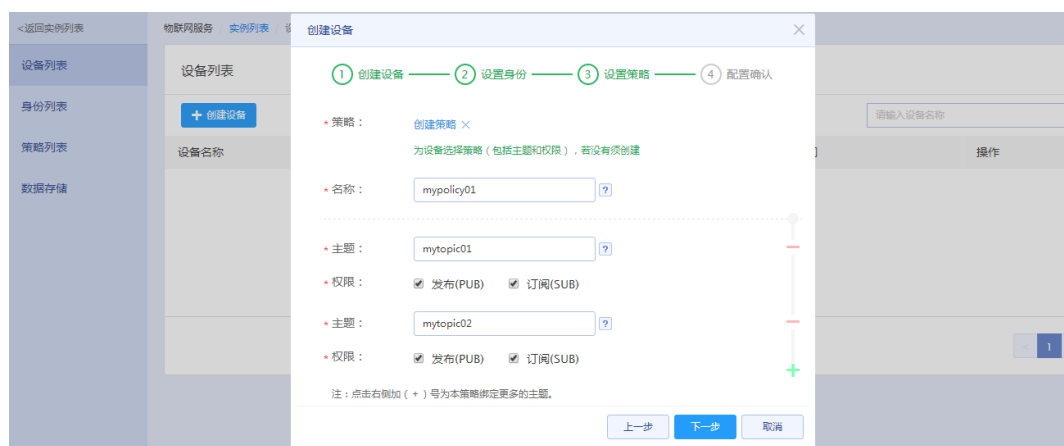
5. 输入身份名称，并点击“下一步”。

6. 为身份绑定策略。此处可以直接从下拉菜单中选择策略名称，如果没有可用策略，也可以选择“+创建”，创建新策略。

7. 输入策略名称、主题及主题的权限，点击“下一步”完成设备创建。

说明：

每个策略下只能配置100个主题。



每个策略可以创建多个主题，在创建策略弹框右侧，点击“+”可以绑定更多的主题。添加主题时，可以使用“#”或“+”作为通配符，关于通配符的介绍，请参看[关于通配符的使用方法](<https://cloud.baidu.com/doc/IOT/GettingStarted.html#.E5.85.B3.E4.BA.8E.E9.80.9A.E9.85.8D.E7.AC.A6.E7.9A.84.E4.BD.BF.E7.94.A8.E6.96.B9.E6.B3.95>)。

创建新设备后，系统将自动初始化改身份对应的密钥。该密钥将用于后续实体设备登录，请妥善保管。



您也可以使用IoT Hub CLI命令创建各项参数，详细过程请参考[IoT Hub CLI](#)。

[视频操作指导-创建物接入设备（无语音）](#)

4.5.3 创建物接入身份

身份需要与设备绑定后才有意义。用户可以在创建设备的过程中创建身份（具体操作请参考[创建物接入设备](#)），也可以单独创建。

创建物接入设备前，应先完成[创建物接入实例](#)操作。

1. 选择“产品服务>物接入IoT Hub”，进入实例列表。
2. 点击实例名称，进入物接入实例页面。
3. 点击左侧导航中的“身份列表”，进入身份列表。



4. 点击“创建身份”，在弹出窗口中输入身份名称，并点击“下一步”。
5. 为身份绑定策略。此处可以直接从下拉菜单中选择策略名称，如果没有可用策略，也可以选择“+创建”，创建新策略。



6. 输入策略名称、主题及主题的权限，点击“下一步”完成身份创建。

说明：

每个策略下只能配置100个主题。

每个策略可以创建多个主题，在创建策略弹框右侧，点击“+”可以绑定更多的主题。添加主题时，可以使用“#”或“+”作为通配符，关于通配符的介绍，请参看[关于通配符的使用方法](<https://cloud.baidu.com/doc/IOT/GettingStarted.html#.E5.85.B3.E4.BA.8E.E9.80.9A.E9.85.8D.E7.AC.A6.E7.9A.84.E4.BD.BF.E7.94.A8.E6.96.B9.E6.B3.95>)。

创建新身份后，系统将自动初始化改身份对应的密钥。该密钥将用于后续实体设备登录，请妥善保管。



4.5.4 重新生成密钥

说明

重新生成密钥后，已经连接的设备会在15分钟后断开连接，请谨慎使用。

每个身份对应一个密钥，该密钥用于实体设备连接物接入服务。创建新身份后，系统将自动初始化改身份对应的密钥。如果用户忘记改密钥，可重新生成密钥。具体操作如下：

1. 选择“产品服务>物接入IoT Hub”，进入实例列表。
2. 点击实例名称，进入物接入实例页面。
3. 点击左侧导航中的“身份列表”，进入身份列表。

- 找到指定的身份，点击“重新生成密钥”。此时系统将在弹出对话框中给出重新初始化后的密钥，请妥善保管。



4.5.5 创建物接入策略

策略需要与身份和设备绑定后才有意义。用户可以在创建设备的过程中创建策略（具体操作请参考[创建物接入设备](#)），也可以单独创建。

创建物接入策略前，应先完成[创建物接入实例](#)操作。

- 选择“产品服务>物接入IoT Hub”，进入实例列表。
- 点击实例名称，进入物接入实例页面。
- 点击左侧导航中的“策略列表”，进入策略列表。



- 点击“创建策略”，在弹出窗口中输入策略名称、主题及主题的权限，点击“确定”完成策略创建。

说明：

每个策略下只能配置100个主题。

每个策略可以创建多个主题，在创建策略弹框右侧，点击“+”可以绑定更多的主题。添加主题时，可以使用“#”或“+”作为通配符，关于通配符的介绍，请参看[\[关于通配符的使用方法\]](https://cloud.baidu.com/doc/IOT/GettingStarted.html#.E5.85.B3.E4.BA.8E.E9.80.9A.E9.85.8D.E7.AC.A6.E7.9A.84.E4.BD.BF.E7.94.A8.E6.96.B9.E6.B3.95) (<https://cloud.baidu.com/doc/IOT/GettingStarted.html#.E5.85.B3.E4.BA.8E.E9.80.9A.E9.85.8D.E7.AC.A6.E7.9A.84.E4.BD.BF.E7.94.A8.E6.96.B9.E6.B3.95>)。



关于通配符的使用方法 MQTT通过“主题”实现将消息从发布者客户端送达至接收者客户端。“主题”是附加在应用消息上的一个标签，发布者客户端将“主题”和“消息”发送至代理服务器，代理服务器将该消息转发至每一个订阅了该“主题”的订阅者客户端。

一个主题名可以由多个主题层级组成，每一层通过“/”斜杠分隔开，例如：“baidu/F1”，“baidu/F2”。如果用户需要一次订阅多个具有类似结构的主题，可以在主题过滤器中包含通配符。通配符只可用在主题过滤器中，在发布应用消息时的主题名不允许包含通配符，主题通配符有两种：

- \#：表示匹配 ≥ 0 个层次，比如a/#就匹配a，a/，a/b，a/b/c。单独的一个#表示匹配所有，不允许a#或a/#/c等形式。
- \+：表示匹配一个层次，例如a/+匹配a/b，a/c，不匹配a/b/c。单独的一个+是允许的，但a+为非法形式。

通配符可以应用在物接入策略中和实体设备的订阅主题中，通过以下示例我们可以进一步了解通配符的作用。

我们使用下表的配置在云端创建三个物接入设备。

实例名称	设备名称	身份名称	策略名称	主题	权限
endpoint01	thing01	principal01	policy01	baidu/#	发布/订阅
endpoint01	thing02	principal02	policy02	baidu/+/area1	发布/订阅
endpoint01	thing03	principal03	policy03	baidu/floor1	发布/订阅
endpoint01	thing04	principal04	policy04	baidu/floor1/area1	发布/订阅

通过MQTT.fx客户端模拟4台实体设备（Device01 ~ Device04），分别使用thing01、thing02、thing03和thing04的用户名和密码连接物接入服务。

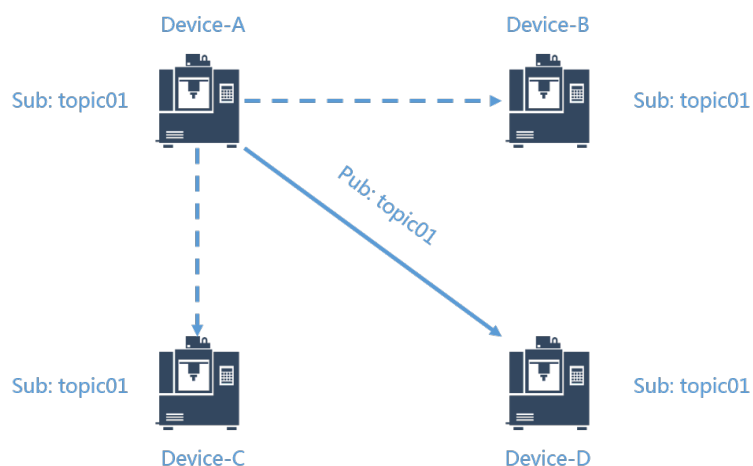
互通情况测试结果可参看下表：

发布设备	发布主题	订阅设备	订阅主题	是否可以互通
Device01	baidu/ de- vice02/area1	Device02	baidu/ de- vice02/area1	是
Device01	baidu/floor1	Device03	baidu/floor1	是
Device01	baidu/ floor1/ area1	Device04	baidu/ floor1/ area1	是
Device02	baidu/ de- vice01/area1	Device01	baidu/ de- vice01/area1	是
Device02	-	Device03	-	否
Device02	baidu/ floor1/ area1	Device04	baidu/ floor1/ area1	是
Device03	baidu/floor1	Device01	baidu/floor1	是
Device03	-	Device02	-	否
Device03	-	Device04	-	否
Device04	baidu/ floor1/ area1	Device01	baidu/ floor1/ area1	是
Device04	baidu/ floor1/ area1	Device02	baidu/ floor1/ area1	是
Device04	-	Device03	-	否

避免用广播方式向特定设备发送消息 在百度云创建一个物接入设备，具体配置如下：

实例名称	设备名称	身份名称	策略名称	主题	权限
endpoint01	thing01	principal01	policy01	topic01	发布/订阅

如下图所示，在该场景中所有实体设备都通过相同的物接入设备接入，拥有相同的用户名、密码和权限。由于所有设备都订阅了相同的主题，当Device-A向Device-D发布消息时Device-B和Device-C也会收到。这种部署方式会对实体设备造成额外的处理负担，同时也会对当月的物接入消息配额造成不必要的损失。



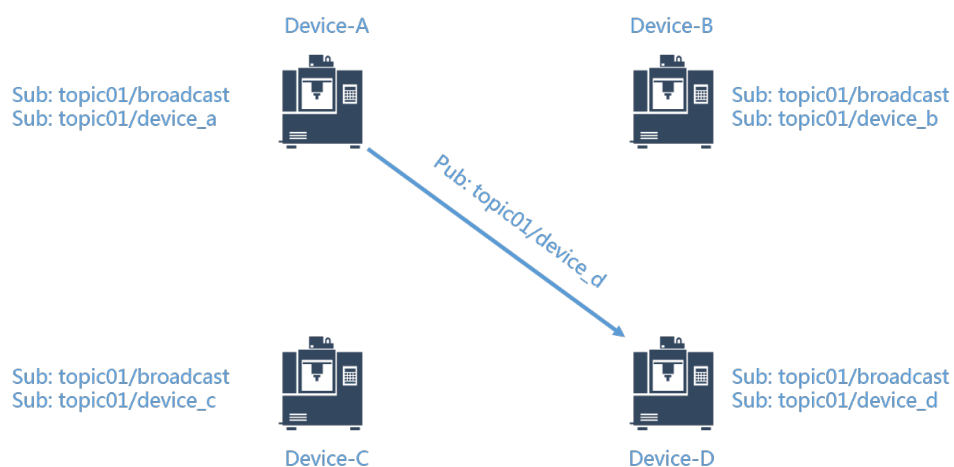
在不增加额外配置的情况下，可以采取以下部署方式解决上述问题。在百度云创建一个物接入设备，具体配置如下：

实例名称	设备名称	身份名称	策略名称	主题	权限
endpoint01	thing01	principal01	policy01	topic01/#	发布/订阅

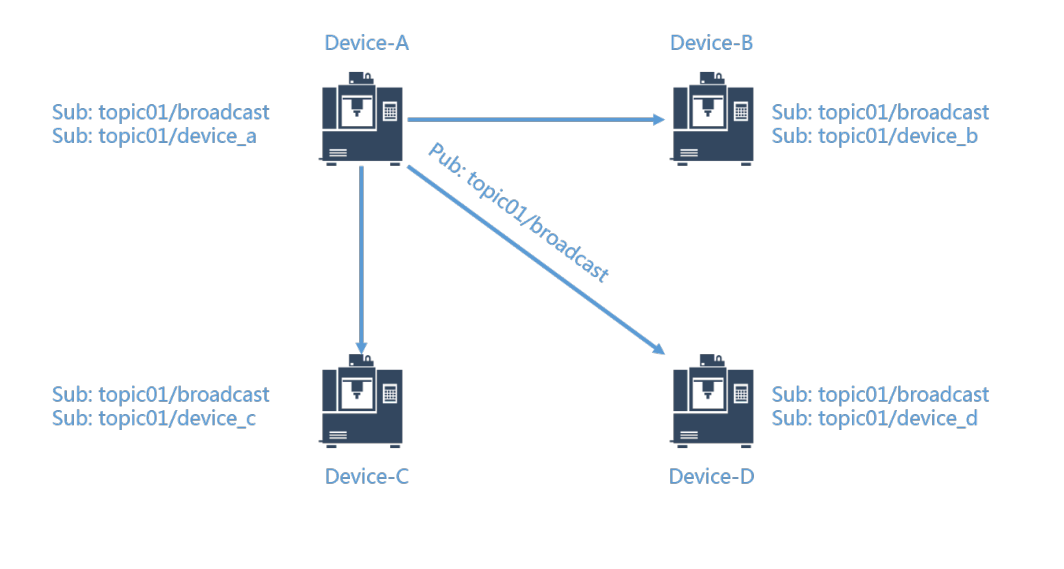
由于在主题中使用了“#”通配符，因此每个设备可以订阅两个主题：

- topic01/broadcast：用来订阅广播消息。
- topic01/DeviceName：DeviceName需要替换为设备的实际名称并全局唯一，用来订阅单播消息。

在单播场景下，Device-A向Device-D发送消息，Device-B和Device-C都不会接收到。



在广播场景下，Device-A发布的消息可以同时被Device-B、Device-C和Device-D接收到。



4.6 连接测试

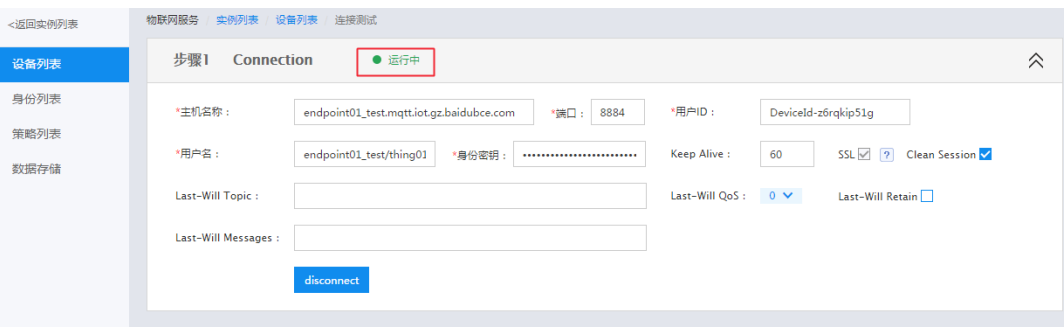
创建设备后，可以在控制台通过“连接测试”功能模拟MQTT客户端，验证物接入的连接情况。用户也可以通过MQTT.fx验证连接情况。

在执行连接测试前，必须先[创建物接入设备](#)。

- 1. 选择“产品服务>物接入IoT Hub”，进入实例列表。
- 2. 点击实例名称，进入物接入实例页面。



- 3. 点击“连接测试”，进入测试页面，填写“身份密钥”，点击“connect”，此时可看到设备状态由“未连接”变成“运行中”。



有关其它参数的介绍，如下表所示：

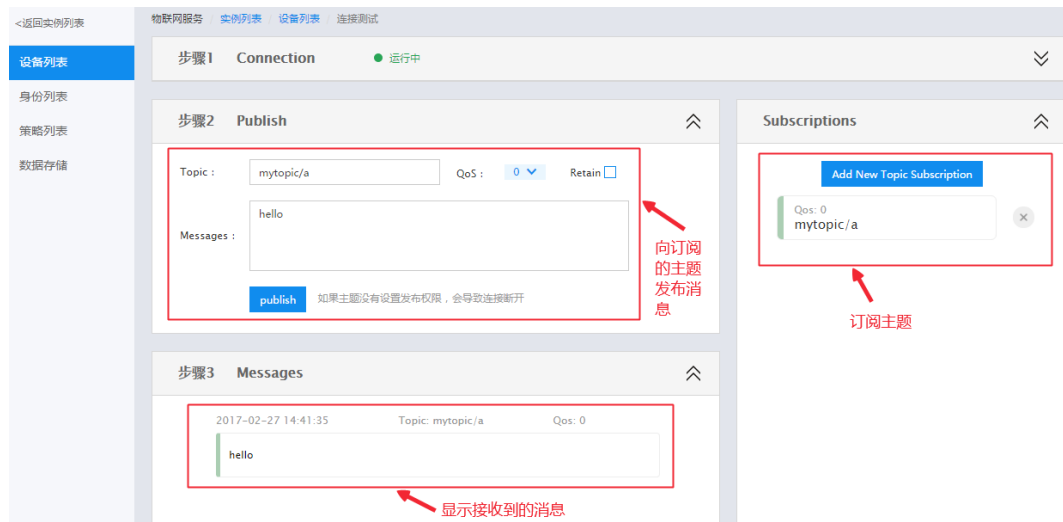
参数名称	描述	必要性
主机名称	实例地址, yourendpoint.mqtt.iot.gz.baidubce.com , 获取方法请参看 创建实例	必填
端口	wss端口8884	必填
用户ID	客户端ID, 用户自定义。在同一个实例下, 每个实体设备需要有一个唯一的ID, 不同实体设备使用同一个client id建立连接会导致其它连接下线。用户ID只支持英文大小写字母, 数字0-9, 中划线和下划线, 不支持其它字符。	必填
用户名	成功创建thing后生成的用户名称, yourendpoint/设备名称	必填
密码	与thing绑定的principal的密码。成功创建身份principal后系统会自动生成的密码	必填
Keep Alive 连接保持时长, 单位为秒 必填 SSL SSL安全验证 必填 Clean Session	清理会话	选填
Last-Will Topic	遗嘱消息主题	选填
Last-Will QoS	遗嘱QoS, 发布遗嘱消息时使用的服务等级	选填
Last-Will Retain	遗嘱保留, 如果勾选遗嘱保留, 遗嘱消息发布时将会保留且发送给新的订阅消息	选填
Last-Will Message	遗嘱消息, 在网络连接关闭后, IoT Hub将会自动发布本条遗嘱消息	选填

*****Keep Alive:** **MQTT协议是一个客户端和服务端长连接的过程。Keep Alive timer以秒为单位, 定义的是从客户端相邻两次接收消息的最大间隔时间, 也可以说是一次长连接的保持时间。因此, 客户端每隔一段时间就需要向服务器发送数据来保持连接(相当于心跳报文的功能), 服务器接收到连接信息后, 会反馈一个响应ACK。当服务器端在Keep Alive timer的1.5倍时间内都没有收到来自客户端的任何消息, 就会默认为客户端断开连接。

*****Clean Session:** **如果该位被设置为false，则该连接被认为是持久连接，其具体表现为：当该客户断开后，任何订阅的主题和QoS被设置为1或2的信息都会保存，直到该客户端再次连接上server端，物接入服务支持将该消息保留24小时。若“clean session”被设置为true，当该客户断开后，所有的订阅主题都会被移除。

Last-Will当一个客户端断开连接的时候，它希望客户端可以发送它指定的消息。该消息和普通消息的结构相同。通过设置Last-Will Topic和Last-Will Message实现。（[遗嘱消息的触发条件有哪些？](#)）

连接成功后，可以设置消息订阅和发布，通过自发自收的方式测试连接，如下图所示：



4.7 配置数据存储

用户可以通过[规则引擎](#)将物接入服务接收到的消息转存至BOS或者百度Kafka进行存储。

规则引擎为免费服务，有关规则引擎的详细操作，请参看[规则引擎-操作指南](#)。

4.8 连接实体设备

4.8.1 使用MQTT.fx客户端模拟实体设备

在执行以下操作前，应先完成[创建物接入服务](#)操作。

实体设备必须支持MQTT协议。用户可以通过调用Paho（即MQTT Client SDK），开发自己的MQTT客户端。

本节以MQTT.fx客户端模拟实体设备，通过一个示例介绍连接实体设备的操作方法。

场景介绍

实体设备（通过MQTT.fx客户端模拟）将当前所处的状态作为MQTT主题发送给物接入服

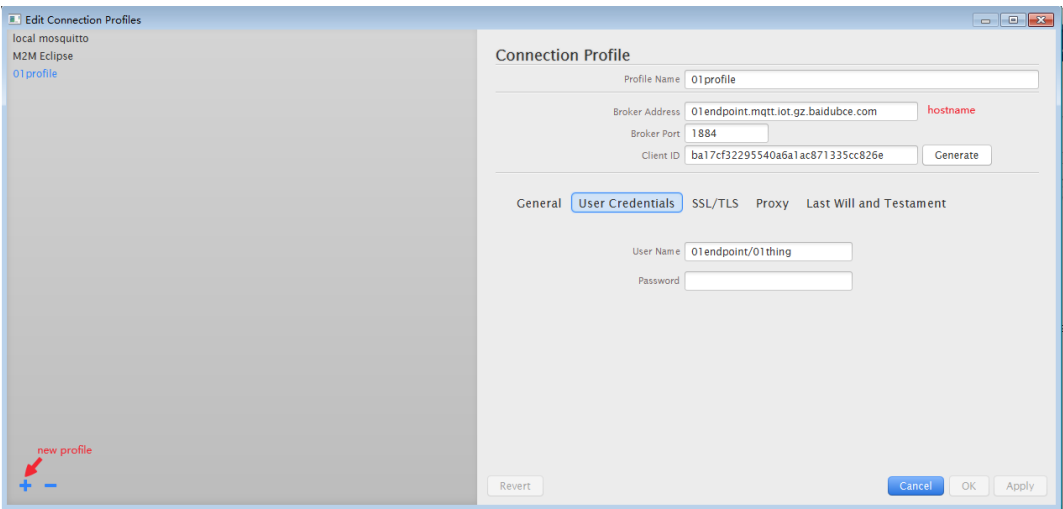
务，每个MQTT主题具有不同层级的名称，如“建筑/楼层/温度。” MQTT代理服务器将接收到的主题topic发送给给所有订阅的客户端。

前提条件

登录MQTT.fx官网，下载并安装MQTT.fx客户端。

操作步骤

- 1. 打开MQTT客户端的设置页面，点击“+” 按键，创建一个新的配置文件。



* 填写Connection profile相关信息：

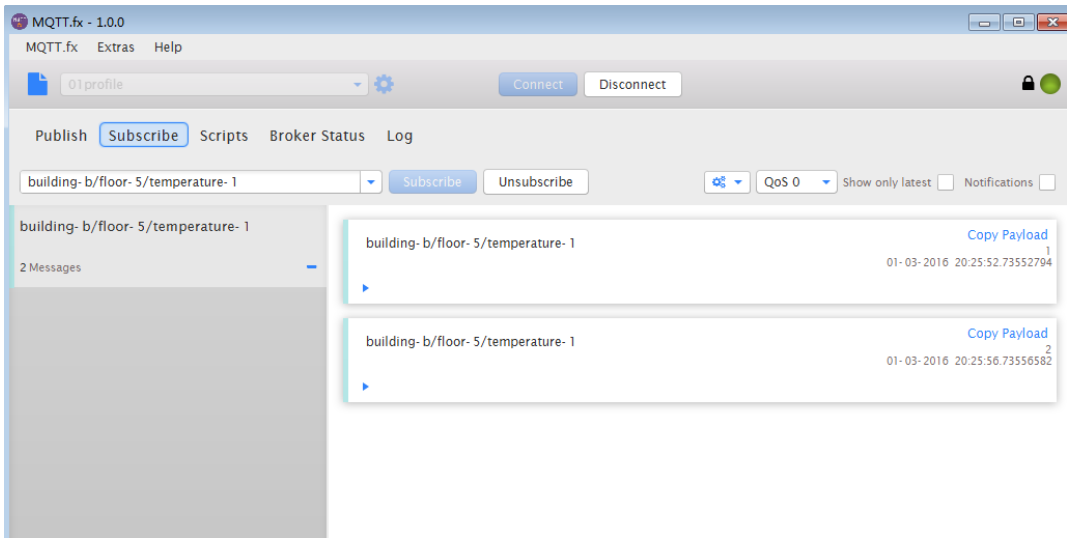
参数名称	说明
profile name	配置文件名称
Broker Address	创建endpoint后返回的hostname
Broker Port	1884
Client ID	客户端ID，支持“a-z”，“0-9”，“_”，“-”字符，且不能大于128bytes，UTF8编码

* 选择User Credential，输入username/password，请参考[创建物接入设备](<https://cloud.baidu.com/doc/IOT/GettingStarted.html#.E5.88.9B.E5.BB.BA.E7.89.A9.E6.8E.A5.E5.85.A5.E8.AE.BE.E5.A4.87>)中的步骤3和步骤7。

* 配置SSL/TLS安全认证，勾选 `Enable SSL/TLS`，选择`CA signed server certificate`认证。

点击“Apply”按键，完成客户端配置。

- 2. 返回MQTT客户端界面，选择新创建的配置文件，点击“connect”按钮连接服务。
- 3. 成功连接后，即可开始订阅消息。
打开Subscribe标签，填写主题topic，例如building-b/floor-5/temperature-1，选择默认的QoS 0，点击“Subscribe”进行订阅操作。
- 4. 发布消息。
打开Publish标签，填写主题topic，例如building-b/floor-5/temperature-1，选择默认的QoS 0，点击“Publish”进行发布操作。
- 5. 返回Subscribe界面，即可看到已接收的订阅消息，参见下图。



4.8.2 使用MQTT Client SDK模拟实体设备

实体设备必须支持MQTT协议。用户可以通过调用Paho（即MQTT Client SDK），开发自己的MQTT客户端。

Paho是Eclipse基金会提供的开源MQTT客户端实现，可以很好的支持百度云物接入服务以实现设备互联和物联网应用。

本示例介绍如何通过MQTT Client SDK订阅一栋办公楼内第五层的温度值。我们通过Net-Beans 来展示如何通过代码来与物联网服务发送或者接受消息。

在执行以下操作前，应先完成[创建物接入服务](#)操作。

新建一个Java应用程序类型的Maven项目，右击“依赖关系”选择添加依赖关系，查询org.eclipse.paho，并加入对org.eclipse.paho.client.mqttv3的依赖。

新建Thermometer类，并指定以下参数：

参数名称	解释
username	创建物接入设备后返回的用户名，参见 创建物接入设备 中的步骤3

参数名称	解释
password	创建身份后返回的密钥，参见 创建物接入身份
endpoint	实例地址，参见 创建物接入实例
port	实例的端口号。端口1883，不支持传输数据加密；端口1884，支持SSL/TLS加密传输。
topic	订阅的主题内容，参见 创建物接入策略
java-client	用来标识设备的ID，用户可自己定义，在同一个实例下，每个实体设备需要有一个唯一的ID

源代码如下：

```
package com.baidu.iot;

import java.io.InputStream;
import java.security.KeyStore;
import java.security.cert.Certificate;
import java.security.cert.CertificateFactory;
import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.TrustManagerFactory;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttMessage;

public class Thermometer {

    public static void main(String[] args) throws Exception {
        String endpoint = "yourhostname"; //输入创建endpoint返回的SSL地址
        String username = "yourendpoint/yourthing"; //输入创建thing返回的username
        String password = "Dm3yyv0Hb7zt/uRwsPgsgbnj7CxuVMc+uDbf4j960="; //输入创建principal返回的password
        String topic = "building-b/floor-5/temperature-1"; //订阅的消息主题，本例是指订阅b号楼第五层的温度

        TrustManagerFactory tmf = TrustManagerFactory.getInstance("X509");
        tmf.init((KeyStore)null);
        TrustManager[] trustManagers = tmf.getTrustManagers();

        SSLContext ctx = SSLContext.getInstance("TLS");
```

```

        ctx.init(null, trustManagers, null);

        MqttConnectOptions options = new MqttConnectOptions();
        options.setCleanSession(true);
        options.setUserName(username);
        options.setPassword(password.toCharArray());
        options.setSocketFactory(ctx.getSocketFactory());

        MqttClient client = new MqttClient(endpoint, "java-client"); //java-
        client为标识设备的ID, 用户可自己定义, 在同一个实例下, 每个实体设备需要有一个唯一的ID
        client.connect(options);
        client.subscribe(topic);

        MqttMessage message = new MqttMessage();
        message.setPayload("15".getBytes());
        client.publish(topic, message);

        client.disconnect();
    }
}

```

4.8.3 持久化会话和消息的缓存

如果客户端在连接的时候指定Clean Session=false, 那这个会话将成为一个持久化会话。如果客户端异常离线, 物接入将缓存“QoS=1”的消息, 待客户端重新连接后将消息发送至客户端。需要注意以下情况:

- 消息只能缓存24小时。
- 如果缓存消息的数量较大, 推荐使用百度Kafka服务。
- 客户端重新连接时Clean Session需置为False (flag=0)。

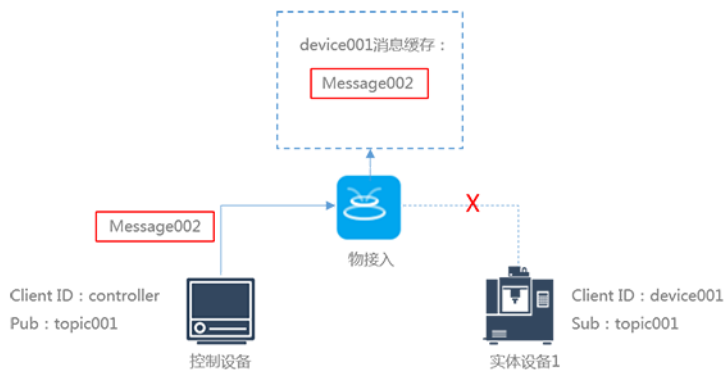
如果Clean Session为True(flag=1), 物接入会丢弃已经缓存任何会话状态信息, 创建一个新的会话连接。

如下例所示:

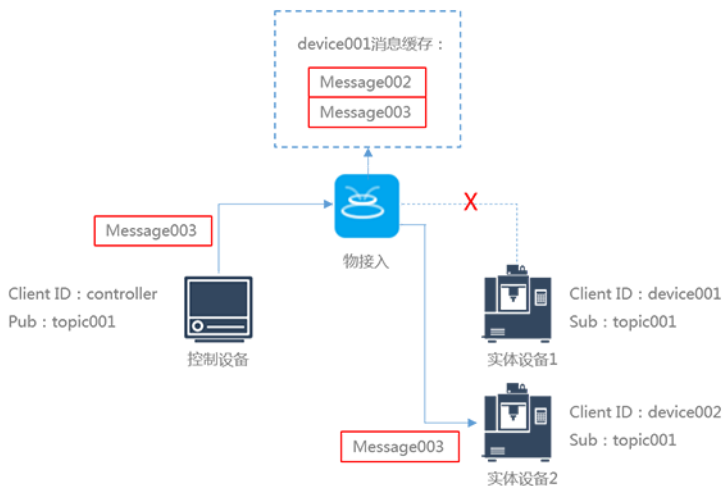
1. 控制设备和实体设备1正常接入物接入, 控制设备发布“QoS=1”的消息Message001, 实体设备1可以正常接收到。



2. 如果实体设备1异常离线，此时物接入将缓存所有发送至该设备的消息。

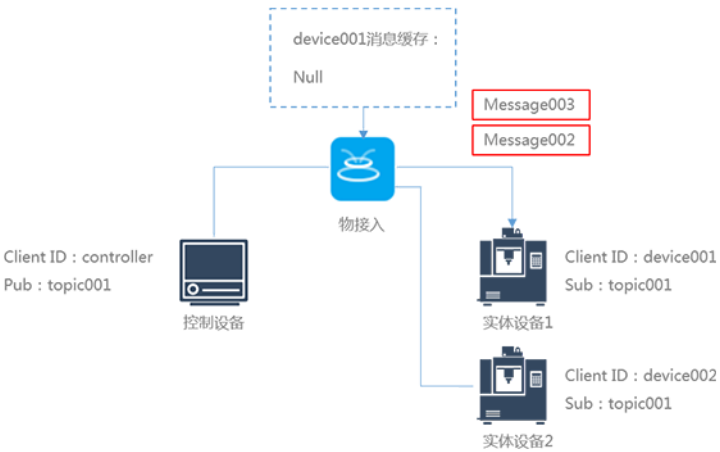


3. 此时如果有其它订阅了相同主题的设备连接至物接入，可以正常接收到控制设备新发送的消息，但无法接收到之前被缓存的消息。

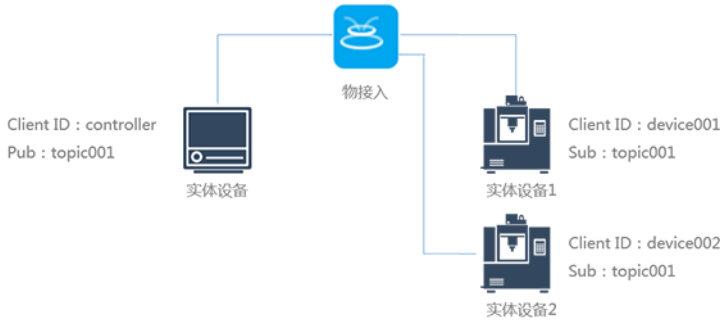


4. 实体设备1重新接入物接入，根据状态的不同，物接入有以下两种处理方式：

- 如果Clean Session置为False，此时物接入将所有缓存的消息转发至实体设备1。



* 如果Clean Session为True，物接入会丢弃已经缓存任何会话状态信息，为实体设备1创建一个新的会话连接。

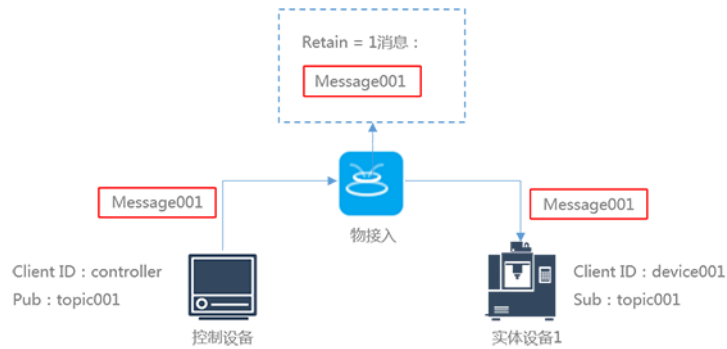


4.8.4 保留消息

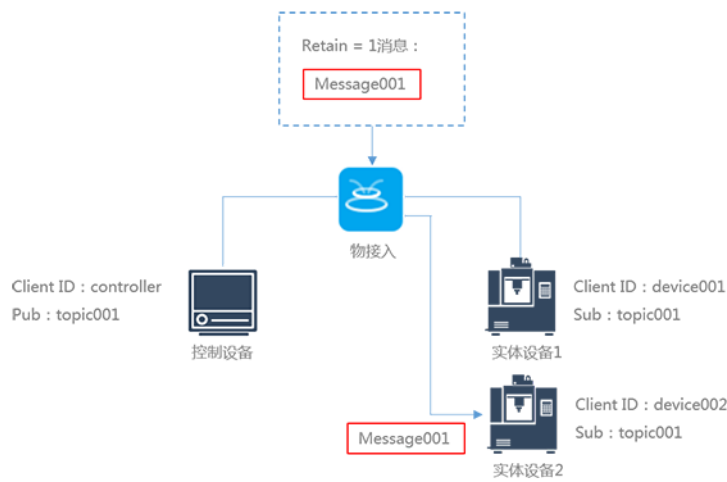
如果PUBLISH消息固定头部RETAIN标记为1，物接入会持久保存此消息，直到该消息被新的PUBLISH消息（RETAIN=1）覆盖或用户主动清除该消息。

对于“RETAIN=1”的消息，物接入不但会将该消息发送给所有当前的订阅者，同时新的接入设备也会收到该消息，如下图所示：

1. 控制设备和实体设备1正常接入物接入，控制设备发布“Retain=1”的消息Message001，实体设备1可以正常接收到并且物接入持久保存该消息。



2. 新实体设备2连接后，物接入持久保存的消息发布给新接入设备。



4.9 多用户协作

多用户访问控制功能实现了多用户协同开发，项目管理者可以基于实例为其他开发测试人员开放查看、配置等权限。项目管理者通过IAM子账号方式进行授权，即在主账号下添加子用户的账号并对子用户进行策略管理，子用户可以通过“IAM用户登录链接”访问主用户的资源。

物接入子账户权限范围说明如下：

权限资源粒度	权限	说明
实例(endpoint)	只读	被授权用户可以查看“实例列表”（只能看到有权限的实例）、“设备列表”、“身份列表”、“策略列表”、“用量统计”页面，只能查看，不能做任何修改操作；

权限资源粒度	权限	说明
实例(endpoint)	管理	被授权用户针对该实例享有与主账号权限一致的权限（不包括删除、创建endpoint，不包括物接入服务额度升级和续费）

配置方法

1. 主账号用户登录后在控制台左下角选择“多用户访问控制”进入用户中心/IAM用户页面。
2. 在“子用户管理列表”页面点击新建用户，填写“用户名”，用户名为子用户的昵称，说明主要用于标识用户类型或其它说明。
创建成功后，系统会自动为该用户生成AK/SK，请妥善保管。
3. 点击“去绑定”，将该用户与子用户的百度账号绑定。

子用户管理列表

IAM用户登录链接: http://bce.baidu.com					
+ 新建用户					
用户名	百度账号	说明	状态	创建时间	操作
zhw	未绑定 去绑定	-	● 活跃	2017-03-23 10:50:25	添加权限 禁用 删除 管理
		-	● 活跃	2016-05-31 17:49:13	添加权限 禁用 删除 管理
		-	● 活跃	2016-05-31 17:46:37	添加权限 禁用 删除 管理
		-	● 活跃	2016-04-05 14:52:54	添加权限 禁用 删除 管理

4. 子用户添加完成后需要对子用户进行策略管理，点击“策略管理” > “创建策略”，自定义物接入子账户权限。

权限配置

权限配置

* 服务类型: 物接入 IoT Hub IOT

策略生成方式: 策略生成器 编辑策略文件

* 操作权限: ☒ 只读权限 ☒ 管理权限 [权限说明](#)

* 资源选择: ● 华南 - 广州

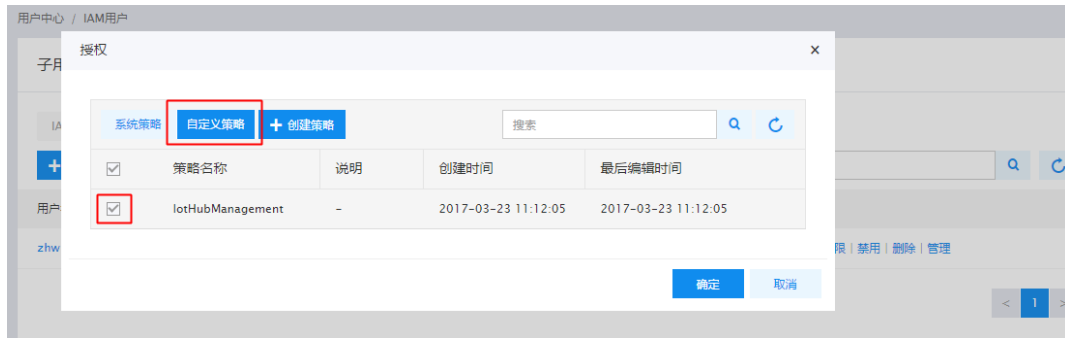
请输入实例名称进行搜索

已选择1个资源

实例名称	描述	地域
<input checked="" type="checkbox"/> test001		华南 - 广州

完成 取消

5. 返回子用户管理列表，点击“添加权限”，在弹出对话框中点击“自定义策略”，选择步骤4中创建的策略。



- 子用户的账户和权限添加完成后，用户可以直接通过子用户管理列表的“IAM用户登录链接”登录主账号的物接入资源，登录成功后用户根据设定的权限享有对主账户资源的操作和查看权限。

第5章

IoT Hub CLI

5.1 目录

- [概述](#)
- [系统限制](#)
- [安装IoT Hub CLI](#)
- [使用IoT Hub CLI](#)
 - [配置IoT Hub CLI](#)
 - [获得 CLI 帮助信息](#)
- [通过CLI使用IoT Hub服务](#)
 - [创建IoT Hub服务](#)
 - [使用MQTT客户端验证IoT Hub服务](#)
 - [应用IoT Hub服务](#)
- [常用操作](#)
- [版本更新记录](#)

5.2 概述

IoT Hub CLI (Command Line Interface) 工具用于在命令行环境下使用物接入IoT Hub，可以通过CLI完成 IoT Hub 服务实例endpoint，设备thing、身份principal的创建和删除，以及与身份对应的权限policy的绑定等功能。

5.3 系统限制

目前每个账户能创建100个endpoint，每个endpoint下可创建10000个thing、10000个policy和10000个principal。如果需要更多配额，请提交工单申请。

5.4 安装IoT Hub CLI

前提条件

IoT Hub CLI命令行工具基于Python 2.7开发，根据操作系统安装相应Python 2.7，目前支持Linux。>说明：>>暂时不支持Python 3.5版本，建议您采用其他版本使用。

下载链接：[Python](#)、[CLI](#) (MD5:90c647ea95d21e6b942fc45611f1961f)。

操作步骤

1.准备Python环境。

- Windows环境

在python官网下载python27.msi后，双击安装。

- Linux环境

以Ubuntu和Redhat为例：

```
# Ubuntu
\ $ sudo apt-get install python python-dev python-setuptools python-pip

# Redhat
\ $ sudo yum install python python-dev python-setuptools python-pip
```

2.安装IoT Hub CLI工具。

- Windows环境

- (1) 解压CLI工具包。
- (2) 在cmd中执行如下命令：

```
\ $ cd bcecli_path %CLI文件路径%
\ $ python_path setup.py install %python路径%
```

- Linux环境

- (1) 解压CLI工具包。

```
\ $ unzip bce-cli-0.3.1.zip
\ $ cd bcecli-0.3.1
```

- (2) 将bcecli的库安装到系统的python目录下。

```
\ $ python setup.py install
```

5.5 使用IoT Hub CLI

5.5.1 配置IoT Hub CLI

配置AK/SK、Region、Host信息

使用IoT Hub CLI工具之前，推荐先设置Access Key、Secure Key、Region。可以通过-c/-configure来设置Access Key ID/Secret Access Key、Region，CLI会在配置后自动写入当前用户主目录。

```
\$ bce.py -c
\$ Access Key [None]: Enter Your AK
\$ Secret Access Key [None]: Enter Your SK
\$ Default region name [gz]:
```

说明：

目前IoT Hub 只支持华南-广州(gz)区域，后续会扩展到更多的区域，敬请期待。

5.5.2 获得 CLI 帮助信息

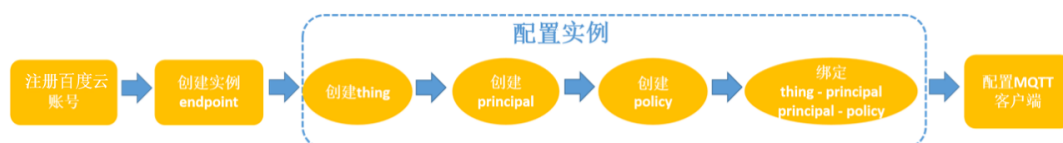
可以在任意命令后面添加-h或-help来查看该命令的帮助信息。

```
\$ bce.py iot -h
\$ bce.py iot list-endpoint --help
```

5.6 通过CLI使用IoT Hub服务

5.6.1 创建IoT Hub服务

使用本手册，使用IoT Hub CLI创建实例，并通过客户端将您的智能设备连接到 IoT Hub 云服务，基本流程如下图所示：



操作步骤

1. 创建IoT Hub 实例endpoint

bce.py iot create-endpoint --endpoint-name "endpointname"

说明：
IoT Hub 服务创建的服务名称（例如endpointname、thingname等）命名方式如下：

- 可以由’ 0-9 “，’ a ’ - ’ z ’ ， ’ - ’ 或 ’ _ ’ 等3-32位长度的字符组成。
- ’ - ’ 或 ’ _ ’ 不要放在开头或者结尾，且不要单独使用或连续使用，比如’ - ’ 。

创建成功后，系统返回如下信息：

```
root@tw-6xm:~/IoT/bcecli-0.3.1# bce.py iot create-endpoint --endpoint-name "iot-endpointname"
{
  "accountId": "d58120db764647d9a14dbb0f969f8c50",
  "createTime": "2016-05-23T06:39:29Z",
  "endpointName": "iot-endpointname",
  "hostname": "iot-endpointname.mqtt.iot.gz.baidubce.com:1884",
  "mqttHostname": "iot-endpointname.mqtt.iot.gz.baidubce.com:1883",
  "mqttTlsHostname": "iot-endpointname.mqtt.iot.gz.baidubce.com:1884",
  "websocketHostname": "iot-endpointname.mqtt.iot.gz.baidubce.com:8884"
}
```

参数	描述
accountId	用户ID
createTime	创建时间
endpointName	实例名称
mqttHostname	不加密传输方式的主机地址
mqttTlsHostname	SSL/TLS加密传输的主机地址
websocketHostna me	Websockets浏览器连接地址，同样支持ssl加密传输

也可以使用-e参数缩略语快速创建endpointname，例如：

bce.py iot create-thing -e "endpointname"

同理，可以通过下列缩写快速创建其他命令：

命令	缩写	描述
-endpointname	-e	创建实例
-thingname	-t	创建设备
-principalname	-p	创建身份
-policyname	-c	创建策略

命令	缩写	描述
-permissions	-p(仅在create policy指定权限列表时使用)	创建权限列表

2. 创建设备thing

在一个endpoint里可以创建一或多个设备，

```
bce.py iot create-thing -e "endpointname" -t "thingname"
```

成功创建设备后，系统返回createTime、endpointName、thingName、username，其中username需要在配置MQTT客户端时输入。

```
root@tw-cxm:~/IoT/bcecli-0.3.1# bce.py iot create-thing -e "iot-endpointname" -t "thing01"
{
  "createTime": "2016-05-23T07:42:58Z",
  "endpointName": "iot-endpointname",
  "thingName": "thing01",
  "username": "iot-endpointname/thing01"
}
root@tw-cxm:~/IoT/bcecli-0.3.1# bce.py iot create-thing -e "iot-endpointname" -t "thing02"
{
  "createTime": "2016-05-23T07:43:29Z",
  "endpointName": "iot-endpointname",
  "thingName": "thing02",
  "username": "iot-endpointname/thing02"
}
root@tw-cxm:~/IoT/bcecli-0.3.1# bce.py iot create-thing -e "iot-endpointname" -t "thing03"
{
  "createTime": "2016-05-23T07:43:38Z",
  "endpointName": "iot-endpointname",
  "thingName": "thing03",
  "username": "iot-endpointname/thing03"
}
```

3. 权限管理

(1) 创建身份principal

在同一个实例endpoint中可以创建一个或多个身份principal。

```
bce.py iot create-principal -e "endpointname" -p "principalname"
```

创建成功后，系统返回endpointName、password、principalName，同样password需要在配置MQTT客户端时输入。

```
root@tw-cxm:~/IoT/bcecli-0.3.1# bce.py iot create-principal -e "iot-endpointname" -p "principal01"
{
  "endpointName": "iot-endpointname",
  "password": "KahBxJlMh+nlg1vm31IO3+UAbmqCJlT9",
  "principalName": "principal01"
}
root@tw-cxm:~/IoT/bcecli-0.3.1# bce.py iot create-principal -e "iot-endpointname" -p "principal02"
{
  "endpointName": "iot-endpointname",
  "password": "ng51yTFJUU+6HkkDmkj8GcJxAiztnIjBK",
  "principalName": "principal02"
}
root@tw-cxm:~/IoT/bcecli-0.3.1# bce.py iot create-principal -e "iot-endpointname" -p "principal03"
{
  "endpointName": "iot-endpointname",
  "password": "GGKe8w1wOc5Hxi1KxzCA0/psQmRPRoyCV",
  "principalName": "principal03"
}
```

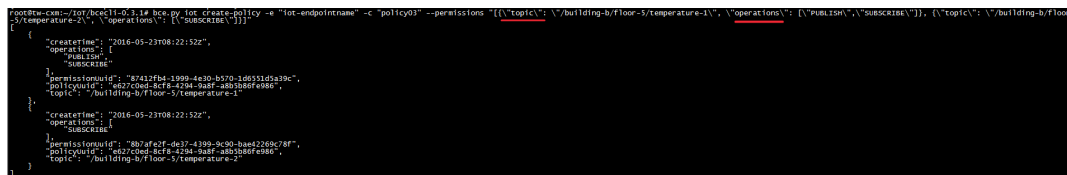
(2) 创建策略policy

每一个principal都可以创建其相应的策略policy，每个policy包含一组权限列表，其中包含消息的主题topic和订阅发布等操作权限。

举例，设定一个主题，用于订阅和发布b号楼第五层的温度，实现方式如下：

```
bce.py iot create-policy -e "endpointname" -c "policyname" --permissions "[{\
\"topic\": \"\"/building-b/floor-5/temperature-1\"\", \"operations\": [\"PUBLISH\
\", \"SUBSCRIBE\"]}, {\"topic\": \"\"/building-b/floor-5/temperature-2\"\", \"\
\"operations\": [\"SUBSCRIBE\"]}]\""
```

创建policy成功后，系统返回operations，permissionUuid等信息，



```
[{"createTime": "2016-05-23T08:22:52Z",
  "operations": [
    "PUBLISH",
    "SUBSCRIBE"
  ],
  "permissionUuid": "87412fb4-1999-4e30-b570-1d6551d5a39c",
  "policyUuid": "e627c0ed-8cf8-4294-9a8f-a8b5b86fe986",
  "topic": "/building-b/floor-5/temperature-1"},
{"createTime": "2016-05-23T08:22:52Z",
  "operations": [
    "SUBSCRIBE"
  ],
  "permissionUuid": "8b7afe2f-de37-4399-9c90-bae42269c78f",
  "policyUuid": "e627c0ed-8cf8-4294-9a8f-a8b5b86fe986",
  "topic": "/building-b/floor-5/temperature-2"}]
```

示例如下：

```
[
  {
    "createTime": "2016-05-23T08:22:52Z",
    "operations": [
      "PUBLISH",
      "SUBSCRIBE"
    ],
    "permissionUuid": "87412fb4-1999-4e30-b570-1d6551d5a39c",
    "policyUuid": "e627c0ed-8cf8-4294-9a8f-a8b5b86fe986",
    "topic": "/building-b/floor-5/temperature-1"
  },
  {
    "createTime": "2016-05-23T08:22:52Z",
    "operations": [
      "SUBSCRIBE"
    ],
    "permissionUuid": "8b7afe2f-de37-4399-9c90-bae42269c78f",
    "policyUuid": "e627c0ed-8cf8-4294-9a8f-a8b5b86fe986",
    "topic": "/building-b/floor-5/temperature-2"
  }
]
```


说明：

Topic不能以 '\$ ' 或 ' / ' 开头，长度不超过255bytes，UTF8编码，且 ' / ' 字符不能超过8个。

(3) 绑定principal和policy

IoT Hub 拥有设备认证与权限管理功能，为了保证数据安全传输，一个principal绑定一个policy。

```
bce.py iot attach-principal-policy -e "endpointname" -c "policyname" -p "principalname"
```

系统返回message:ok，绑定成功。

(4) 绑定thing和principal

每个设备独立认证，一个设备thing绑定一个身份principal。

```
bce.py iot attach-thing-principal -e "endpointname" -t "thingname" -p "principalname"
```

绑定成功后系统返回message:ok。至此，您已成功创建了IoT Hub 物接入应用的一系列服务。

```
root@tw-cxm:~/IoT/bcecli-0.3.1# bce.py iot attach-principal-policy -e "iot-endpointname" -c "policy03" -p "principal01"
{
  "message": "ok"
}
root@tw-cxm:~/IoT/bcecli-0.3.1# bce.py iot attach-thing-principal -e "iot-endpointname" -t "thing01" -p "principal01"
{
  "message": "ok"
}
```

说明：

目前每个账户能创建100个endpoint，每个endpoint下可创建10000个thing、10000个policy和10000个principal。如果需要更多配额，请提交工单申请。

5.6.2 使用MQTT客户端验证IoT Hub服务

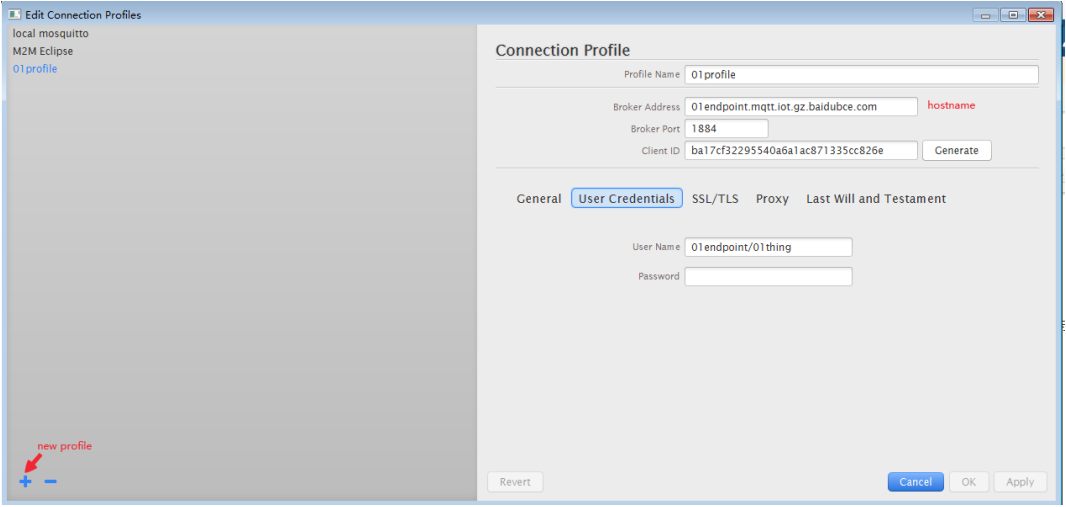
配置MQTT的应用客户端，可以快速验证是否可以实现与IoT Hub服务交流发送或者接收消息。IoT Hub客户端参考[MQTT.fx使用帮助](#)。

前提条件

- 下载并安装[MQTT 客户端](#)。

操作步骤

1.打开MQTT客户端的设置页面，点击“+”按钮，创建一个新的配置文件。



- 填写Connection profile相关信息：

参数名称	说明
profile name	配置文件名称
Broker Address	创建endpoint后返回的hostname
Broker Port	1884
Client ID	客户端ID，支持“a-z”，“0-9”，“_”，“-”字符，且不能大于128bytes，UTF8编码

- 选择User Credential，输入创建 IoT Hub 服务返回的username/password，参考[创建 IoT Hub服务](#)。
- 配置SSL/TLS安全认证，勾选 [Enable SSL/TLS](#)，选择[CA signed server certificate](#)认证。

点击“Apply”按钮，完成客户端配置。

2.返回MQTT客户端界面，选择新创建的配置文件，点击“connect”按钮连接服务。

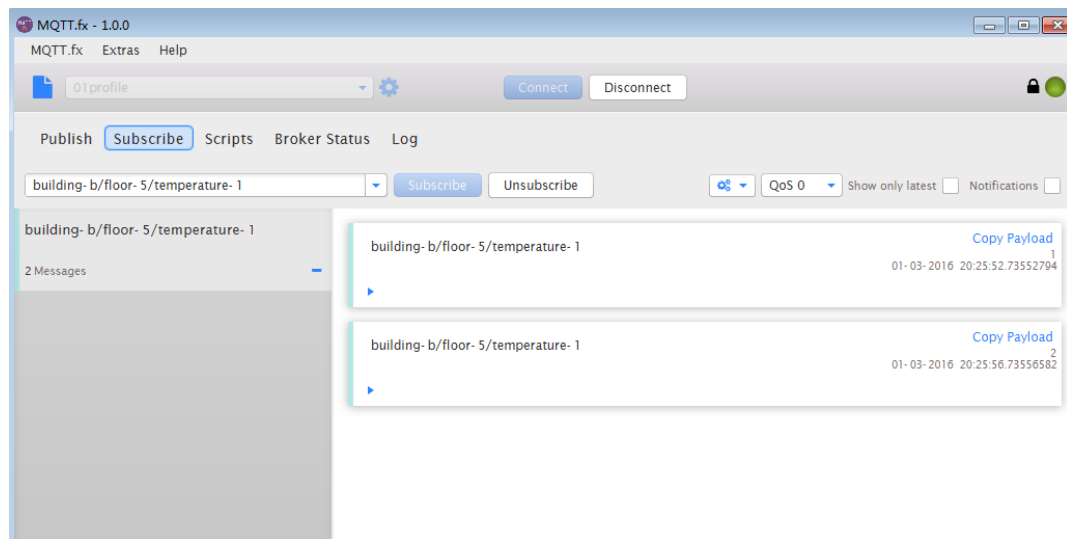
3.成功连接后，即可开始订阅消息。

打开Subscribe标签，填写主题topic，例如[building-b/floor-5/temperature-1](#)，选择默认的QoS 0，点击“Subscribe”进行订阅操作。

4.发布消息。

打开Publish标签，填写主题topic，例如[building-b/floor-5/temperature-1](#)，选择默认的QoS 0，点击“Publish”进行发布操作。

5.返回Subscribe界面，即可看到已接收的订阅消息，参见下图。



5.6.3 应用IoT Hub服务

Paho是Eclipse基金会提供的开源MQTT客户端实现，可以很好的支持百度云物接入IoT Hub服务以实现设备互联和物联网应用。我们通过[NetBeans](#)来展示如何通过代码来与物接入服务发送或者接受消息。

提供一个示例，例如订阅一栋办公楼内第五层的温度值。

新建一个Java应用程序类型的Maven项目，右击“依赖关系”选择添加依赖关系，查询org.eclipse.paho，并加入对org.eclipse.paho.client.mqttv3的依赖。

新建Thermometer类，源代码如下：

```
package com.baidu.iot;

import java.io.InputStream;
import java.security.KeyStore;
import java.security.cert.Certificate;
import java.security.cert.CertificateFactory;
import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.TrustManagerFactory;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttMessage;

public class Thermometer {

    public static void main(String[] args) throws Exception {
```

```

String endpoint = "hostname";    //输入创建实例endpoint返回的hostname
String username = "yourendpoint/yourthing"; //输入创建thing返回的username
String password = "Dm3yyv0Hb7zt/uRwsPgfsfsgbnj7CxuVMc+uDbf4j960="; //输入创建principal返回的password

String topic = "building-b/floor-5/temperature-1"; //订阅的消息主题，本例是指订阅b号楼第五层的温度

TrustManagerFactory tmf = TrustManagerFactory.getInstance("X509");
tmf.init((KeyStore)null);
TrustManager[] trustManagers = tmf.getTrustManagers();

SSLContext ctx = SSLContext.getInstance("TLS");
ctx.init(null, trustManagers, null);

MqttConnectOptions options = new MqttConnectOptions();
options.setCleanSession(true);
options.setUserName(username);
options.setPassword(password.toCharArray());
options.setSocketFactory(ctx.getSocketFactory());

MqttClient client = new MqttClient(endpoint, "java-client");
client.connect(options);

MqttMessage message = new MqttMessage();
message.setPayload("15".getBytes());
client.publish(topic, message);

client.disconnect();
}
}

```

运行成功后，即可在MQTT.fx查看订阅的消息。

5.7 常用操作

[查看已创建物接入服务](#)

- 查看所有已创建的服务，目前每个账户能创建100个endpoint，每个endpoint下可创建10000个thing、10000个policy和10000个principal。如果需要更多配额，请提交工单申请。

```
$ bce.py iot list-endpoint [-q QUERY] [-o {desc,asc}] [-b ORDER_BY] [-n PAGE_NO] [-s PAGE_SIZE]
```

返回示例如下：

```
{
  "order": "desc",
  "orderBy": "createtime",
  "pageNo": 1,
  "pageSize": 50,
  "result": [
    {
      "accountUuid": "d58120db764647d9a14dbb0f969f8c50",
      "createTime": "2016-04-07T03:13:18Z",
      "endpointName": "03endpoint",
      "hostname": "03endpoint.mqtt.iot.gz.baidubce.com:1884"
    }
  ],
  "totalCount": 1
}
```

- 查看设备列表（如：空调数量）

```
$ bce.py iot list-thing -e ENDPOINT_NAME [-q QUERY] [-o {desc,asc}] [-b ORDER_BY]
[-n PAGE_NO] [-s PAGE_SIZE]
```

- 查看身份列表

```
$ bce.py iot list-principal -e ENDPOINT_NAME [-t THING_NAME] [-q QUERY] [-o {desc,asc}]
[-b ORDER_BY] [-n PAGE_NO] [-s PAGE_SIZE]
```

- 查看设置的策略，包括主题和权限。

```
$ bce.py iot list-policy -e ENDPOINT_NAME [-p PRINCIPAL_NAME] [-q QUERY] [-o {desc,asc}]
[-b ORDER_BY] [-n PAGE_NO] [-s PAGE_SIZE]
```

返回如下示例：

```
[
  {
    "endpointName": "endpointname",
    "permissions": [
      {
        "operations": [
          "PUBLISH",
          "SUBSCRIBE"
        ],
        "permissionUuid": "8cec32ea-3154-42f5-ac50-51ccdb07783f",
        "policyUuid": "5d0cff9f-408c-4009-a28a-2313754da201",
        "topic": "building-b/floor-5/temperature-1"
      }
    ],
  },
]
```

```

        {
            "operations": [
                "SUBSCRIBE"
            ],
            "permissionUuid": "d67baab5-6bbf-4b97-96e9-3ff0bfd78f04",
            "policyUuid": "5d0cff9f-408c-4009-a28a-2313754da201",
            "topic": "building-b/floor-5/temperature-2"
        }
    ],
    "policyName": "01policy"
}
]

```

绑定身份和策略

- 绑定principal、policy

```
$ bce.py iot attach-principal-policy -e ENDPOINT_NAME -p PRINCIPAL_NAME -c POLICY_NAME
```

- 绑定thing-principal

```
$ bce.py iot attach-thing-principal -e ENDPOINT_NAME -t THING_NAME -p PRINCIPAL_NAME
```

解绑已组合服务

- 解绑principal、policy，将一个principal的权限移除。

```
$ bce.py iot remove-principal-policy -e ENDPOINT_NAME -p PRINCIPAL_NAME -c POLICY_NAME
```

- 解绑thing-principal，将一个设备的principal移除。

```
$ bce.py iot remove-thing-principal -e ENDPOINT_NAME -t THING_NAME -p PRINCIPAL_NAME
```

删除服务

- 删除已创建的服务

```
$ bce.py iot delete-endpoint -e ENDPOINT_NAME
```

- 删除设备

```
$ bce.py iot delete-thing -e ENDPOINT_NAME -t THING_NAME
```

- 删除身份

```
$ bce.py iot delete-principal -e ENDPOINT_NAME -p PRINCIPAL_NAME
```

- 删除设置的策略

```
$ bce.py iot delete-policy -e ENDPOINT_NAME -c POLICY_NAME
```

获取物接入服务资源

- 获取已创建的服务

```
$ bce.py iot get-endpoint -e ENDPOINT_NAME
```

- 获取设备

```
$ bce.py iot get-thing -e ENDPOINT_NAME -t THING_NAME
```

- 获取身份

```
$ bce.py iot get-principal -e ENDPOINT_NAME -p PRINCIPAL_NAME
```

- 获取设置的策略

```
$ bce.py iot get-policy -e ENDPOINT_NAME -c POLICY_NAME
```

重置密码

```
\$ bce.py iot regenerate-principal -e ENDPOINT_NAME -p PRINCIPAL_NAME [-t {all,password,cert}] [-o OUTPUT_DIR]
```

5.8 版本更新记录

- IoT Hub CLI工具包 [2016-03-08] 版本号0.1.0

通过CLI完成 IOT Hub 服务的创建、删除和列举等功能。

- IoT Hub CLI工具包 [2016-03-22] 版本号0.2.0

更新创建策略命令，支持一个策略拥有一组权限列表。

- IoT Hub CLI工具包 [2016-04-15] 版本号0.3.1

更新默认参数。

第6章

API参考

6.0.1 目录

- [介绍](#)
 - [简介](#)
 - [调用方式](#)
 - * [概述](#)
 - * [通用约定](#)
 - * [公共请求头](#)
 - * [公共响应头](#)
 - * [响应状态码](#)
 - * [通用错误返回格式](#)
 - * [公共错误码](#)
 - * [签名认证](#)
 - * [签名生成算法](#)
 - [多区域选择](#)
- [认证](#)
 - [认证](#)
 - [鉴权](#)
- [动作](#)
 - [给一个Thing添加一个Principal](#)
 - [从一个Thing移除一个Principal](#)
 - [给一个Principal添加一个Policy](#)
 - [从一个Principal移除一个Policy](#)
- [Endpoint](#)
 - [获取endpoint列表](#)
 - [获取指定的endpoint信息](#)
 - [创建endpoint](#)

- 删除endpoint
- Thing
 - 获取thing列表
 - 获取指定的thing信息
 - 创建thing
 - 删除thing
- Principal
 - 获取principal列表
 - 获取指定的principal信息
 - 创建principal
 - 重新生成密钥
 - 删除principal
- Policy
 - 获取policy列表
 - 获取指定的policy信息
 - 创建policy
 - 删除policy
- Permission
 - 获取policy下所有topic信息
 - 获取指定topic的信息
 - 在policy下设置topic
 - 更新已有的topic设置
 - 删除已有的topic
- Client
 - 获取指定MQTT客户端在线状态
 - 获取所有MQTT客户端在线状态
 - Publish Message

6.1 介绍

6.1.1 简介

物接入是全托管的云服务，可以在智能设备与云端之间建立安全的双向连接，并通过主流的物联网协议（如MQTT）通讯，实现从设备端到云端以及从云端到设备端的安全稳定的消息传输。IoT Hub API主要提供Thing、Principal、Policy和Topic的创建、删除、查询等功能，以Restful API的形式提供。

6.1.2 调用方式

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

通用约定

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

头域（Header）	是否必须	说明
Authorization	必须	包含Access Key与请求签名
Host	必须	包含API的域名
Content-Type	可选	application/json; charset=utf-8

公共请求头

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

公共响应头

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

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

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

消息体定义

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

错误返回示例

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

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

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

公共错误码

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

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

签名字符串格式

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

其中：

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

签名生成算法 有关签名生成算法的具体介绍，请参看[鉴权认证机制](#)。

6.1.3 多区域选择

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

6.2 认证

6.2.1 认证

相对URI	HTTP 方式
/v1/auth/authenticate/password	POST

请求参数

名称	类型	是否必选	含义
password	String	Y	身份(principal)的密钥/ak
username	String	Y	thing的username

返回参数

名称	类型	含义
endpointName	String	thing 所属 的 endpoint 的 Name
endpointUuid	String	thing 所属 的 endpoint 的 Uuid
principalUuid	String	thing在这次认证中所用的 principal的uuid

6.2.2 鉴权

相对URI	HTTP 方式
/v1/auth/authorize	POST

请求参数

名称	类型	是否必选	含义
principalUuid	String	Y	需要鉴权的 principal的Uuid
action	ENUM	Y	动作。目前定义了5个操作（和apollo中对应）:CONNECT: 主体与 broker 建立连接CREATE: 主体创建 TopicSEND: 主体向特定 Topic 发送 messageRECEIVE: 主体从特定 Topic接收messageCONSUME: clean session=false 时替代receive

名称	类型	是否必选	含义
topic	String	N	这次操作对应的topic。SEND, RECEIVE, CONSUME操作必须要传Topic

返回参数

无特殊返回参数。

6.3 动作

6.3.1 给一个Thing添加一个Principal

相对URI	HTTP 方式
/v1/action/attach-thing-principal	POST

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	thing 所属的 endpoint的名称
thingName	String	Y	thing的名称
principalName	String	Y	principal的名称

返回参数

无特殊返回参数。

请求示例

```
POST /v1/action/attach-thing-principal HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
{
  "principalName": "principal-1"
  "endpointName": "endpoint-1"
  "thingName": "thing-1"
```

```
}
```

返回示例

```
HTTP/1.1 201 Created
x-bce-request-id: fecfa1bd-aa94-42b0-99b1-b33e4b1b1c88
Content-Type: application/json;charset=UTF-8
{
  "message": "ok"
}
```

6.3.2 从一个Thing移除一个Principal

相对URI	HTTP 方式
/v1/action/remove-thing-principal	POST

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	thing 所属的 endpoint 的名称
thingName	String	Y	thing 的名称
principalName	String	Y	principal 的名称

返回参数

无特殊返回参数。

请求示例

```
POST /v1/action/remove-thing-principal HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
{
  "principalName": "principal-1"
  "endpointName": "endpoint-1"
  "thingName": "thing-1"
}
```


返回示例

```
HTTP/1.1 201 Created
x-bce-request-id: 46465c45-a6da-4211-ac11-ebbb9e0538da
Content-Type: application/json;charset=UTF-8
{
  "message": "ok"
}
```

6.3.3 给一个Principal添加一个Policy

相对URI	HTTP 方式
/v1/action/attach-principal-policy	POST

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	principal 所属的 endpoint 的名称
principalName	String	Y	principal 的名称
policyName	String	Y	thing 的名称

返回参数

无特殊返回参数。

请求示例

```
POST /v1/action/attach-principal-policy HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
{
  "principalName": "principal-1"
  "endpointName": "endpoint-1"
  "policyName": "policy-1"
}
```

返回示例

HTTP/1.1 201 Created

x-bce-request-id: 2030375e-b56e-442b-b106-84778ae46f3b

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

```
{
  "message": "ok"
}
```

6.3.4 从一个Principal移除一个Policy

相对URI	HTTP 方式
/v1/action/remove-thing-principal-policy	POST

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	principal 所属的 endpoint 的名称
principalName	String	Y	principal 的名称
policyName	String	Y	thing 的名称

返回参数

无特殊返回参数。

请求示例

POST /v1/action/remove-principal-policy HTTP/1.1

host: iot.gz.baidubce.com

authorization: {authorization}

content-type: text/plain;charset=UTF-8

```
{
  "principalName": "principal-1"
  "endpointName": "endpoint-1"
  "policyName": "policy-1"
}
```

返回示例

HTTP/1.1 201 Created

x-bce-request-id: b998337c-a4fb-4bc1-b557-39d2b0fd27e7

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

```
{
  "message": "ok"
}
```

6.4 Endpoint

6.4.1 获取endpoint列表

相对URI	HTTP 方式
/v1/endpoint	GET

请求参数

名称	类型	是否必选	默认值	说明
order	ENUM['desc' , 'asc']	N	desc	排序的方式，不区分大小写
orderBy	String	N	createTime	另外一个支持排序的字段是name。
pageNo	Int	N	1	页码
pageSize	Int	N	50	每页item个数，最大值200
q	String	N	-	模糊查询的内容。目前支持name字段模糊查询

返回参数

名称	类型	含义
accountUuid	String	创建者的Uuid
creadteTime	String	创建时间
endpointName	String	endpoint名称
mqttHostname	String	mqtt协议的url（非加密）

名称	类型	含义
mqttTlsHostname	String	mqtt协议的url (加密)
uuid	String	系统自动生成的一个endpoint的唯一值
websocketHostname	String	websocket协议的url

请求示例

```
GET /v1/endpoint HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: a9000b0d-c13c-4241-9ee1-0876f750716f
Content-Type: application/json;charset=UTF-8
{
  "totalCount": 2,
  "result": [
    {
      "mqttHostname": "endpoint-1.xxx.iot.baidubce.com:8061",
      "accountUuid": "myaccount",
      "hostname": "endpoint-1.xxx.iot.baidubce.com:61614",
      "endpointName": "endpoint-1",
      "mqttTlsHostname": "endpoint-1.xxx.iot.baidubce.com:61614",
      "createTime": "2016-08-31T03:36:24Z",
      "websocketHostname": "endpoint-1.xxx.iot.baidubce.com:8064",
      "uuid": "2013ffab-c17e-4657-839d-941bbe6c6c84"
    },
    {
      "mqttHostname": "endpoint-2.xxx.iot.baidubce.com:8061",
      "accountUuid": "myaccount",
      "hostname": "endpoint-2.xxx.iot.baidubce.com:61614",
      "endpointName": "endpoint-2",
      "mqttTlsHostname": "endpoint-2.xxx.iot.baidubce.com:61614",
      "createTime": "2016-08-25T08:43:42Z",
      "websocketHostname": "endpoint-2.xxx.iot.baidubce.com:8064",
      "uuid": "9e65bb32-731b-4b25-982b-13a12a0ff35e"
    }
  ]
}
```

```

],
"order": "desc",
"orderBy": "createtime",
"pageSize": 50,
"pageNo": 1
}

```

6.4.2 获取指定的endpoint信息

相对URI	HTTP 方式
/v1/endpoint/{endpointName}	GET

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint的名称

返回参数

名称	类型	含义
accountUuid	String	创建者的Uuid
creadteTime	String	创建时间
endpointName	String	endpoint名称
hostname	String	mqtt连接时所需的host
mqttHostname	String	mqtt协议的url（非加密）
mqttTlsHostname	String	mqtt协议的url（加密）
uuid	String	系统自动生成的一个endpoint的唯一值
websocketHostna me	String	websocket协议的url

请求示例

```

GET /v1/endpoint/endpoint-1 HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8

```

返回示例

```

HTTP/1.1 200 OK
x-bce-request-id: f494dff9-7ece-43be-a509-b8c1cf56e13f
Content-Type: application/json;charset=UTF-8
{
  "hostname": "endpoint-1.xxx.iot.baidubce.com:61614",
  "mqttHostname": "endpoint-1.xxx.iot.baidubce.com:8061",
  "createTime": "2016-08-31T03:36:24Z",
  "uuid": "2013ffab-c17e-4657-839d-941bbe6c6c84",
  "accountUuid": "mycount",
  "websocketHostname": "endpoint-1.xxx.iot.baidubce.com:8064",
  "mqttTlsHostname": "endpoint-1.xxx.iot.baidubce.com:61614",
  "endpointName": "endpoint-1"
}

```

6.4.3 创建endpoint

相对URI	HTTP 方式
/v1/endpoint	POST

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint的名称

返回参数

名称	类型	含义
accountUuid	String	创建者的Uuid
creadteTime	String	创建时间
endpointName	String	endpoint名称
hostname	String	mqtt连接时所需的host
mqttHostname	String	mqtt协议的url（非加密）
mqttTlsHostname	String	mqtt协议的url（加密）
uuid	String	系统自动生成的一个endpoint的唯一值

名称	类型	含义
websocketHostname	String	websocket协议的url

请求示例

```
POST /v1/endpoint HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
{
  "endpointName": "endpoint-1"
}
```

返回示例

```
HTTP/1.1 201 Created
x-bce-request-id: a80aba01-f81e-4dfc-a57b-76b05a30ee07
Content-Type: application/json;charset=UTF-8
{
  "mqttHostname": "endpoint-1.xxx.iot.baidubce.com:8061",
  "accountUuid": "myaccount",
  "hostname": "endpoint-1.xxx.iot.baidubce.com:61614",
  "endpointName": "endpoint-1",
  "mqttTlsHostname": "endpoint-1.xxx.iot.baidubce.com:61614",
  "createTime": "2016-08-31T03:36:24Z",
  "websocketHostname": "endpoint-1.xxx.iot.baidubce.com:8064",
  "uuid": "2013ffab-c17e-4657-839d-941bbe6c6c84"
}
```

6.4.4 删除endpoint

相对URI	HTTP 方式
/v1/endpoint/{endpointName}	DELETE

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint的名称

[返回参数](#)

无特殊返回参数。

[请求示例](#)

```
DELETE /v1/endpoint/endpoint-1 HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
```

[返回示例](#)

```
HTTP/1.1 204 No Content
x-bce-request-id: 50f11d37-54ea-44a5-9863-94441906b9bc
Content-Type: application/json;charset=UTF-8
```

6.5 Thing

6.5.1 获取thing列表

相对URI	HTTP 方式
/v1/endpoint/{endpointName}/thing	GET

[请求参数](#)

名称	类型	是否必选	默认值	说明
endpointName	String	Y	-	endpoint 的名称
order	ENUM['desc' , 'asc']	N	desc	排序的方式，不区分大小写
orderBy	String	N	createTime	另外一个支持排序的字段是 name。
pageNo	Int	N	1	页码
pageSize	Int	N	50	每页 item 个数，最大值200

名称	类型	是否必选	默认值	说明
q	String	N	-	模糊查询的内容。目前支持name字段模糊查询

返回参数

名称	类型	含义
username	String	用于mqtt认证
thingName	String	thing的名称
endpointName	String	所属的Endpoint
createTime	String	创建时间

请求示例

```
GET /v1/endpoint/endpoint-1/thing HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
```

返回示例

```
HTTP/1.1 200 Created
x-bce-request-id: c6903ab2-bf5e-4afb-8bbf-2e537edd3c49
Content-Type: application/json;charset=UTF-8
{
  "totalCount": 1,
  "result": [
    {
      "username": "endpoint-1/thing-1",
      "thingName": "thing-1",
      "endpointName": "endpoint-1",
      "createTime": "2016-08-31T05:12:39Z"
    }
  ],
  "order": "desc",
  "orderBy": "createtime",
  "pageSize": 50,
```

```
"pageNo": 1
}
```

6.5.2 获取指定的thing信息

相对URI	HTTP 方式
/ v1/ endpoint/ {endpointName}/ thing/ {thingName}	GET

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint的名称
thingName	String	Y	thing的名称

返回参数

名称	类型	含义
username	String	用于mqtt认证
thingName	String	thing的名称
endpointName	String	所属的Endpoint
createTime	String	创建时间

请求示例

```
GET /v1/endpoint/endpoint-1/thing/thing-1 HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
```

返回示例

```
HTTP/1.1 200 Created
x-bce-request-id: 360e61be-36fd-4607-b226-199fd6bc11bd
Content-Type: application/json;charset=UTF-8
{
  "username": "endpoint-1/thing-1",
```

```
"thingName": "thing-1",  
"endpointName": "endpoint-1",  
"createTime": "2016-08-31T05:12:39Z"  
}
```

6.5.3 创建thing

相对URI	HTTP 方式
/v1/endpoint/{endpointName}/thing	POST

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint的名称
thingName	String	Y	thing的名称

返回参数

名称	类型	含义
username	String	用于mqtt认证
thingName	String	thing的名称
endpointName	String	所属的Endpoint
createTime	String	创建时间

请求示例

```
POST /v1/endpoint/endpoint-1/thing HTTP/1.1  
host: iot.gz.baidubce.com  
authorization: {authorization}  
content-type: text/plain;charset=UTF-8  
{  
  "thingName": "thing-1"  
}
```

返回示例

```
HTTP/1.1 201 Created
```

```
x-bce-request-id: 22b69f0d-11ad-4547-a1dd-20ac37282f63
Content-Type: application/json;charset=UTF-8
{
  "username": "endpoint-1/thing-1",
  "thingName": "thing-1",
  "endpointName": "endpoint-1",
  "createTime": "2016-08-31T05:12:39Z"
}
```

6.5.4 删除thing

相对URI	HTTP 方式
/ v1/ endpoint/ {endpointName}/ thing/ {thingName}	DELETE

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint的名称
thingName	String	Y	thing的名称

返回参数

无特殊返回参数。

请求示例

```
DELETE /v1/endpoint/endpoint-1/thing/thing-1 HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
```

返回示例

```
HTTP/1.1 204 No Content
x-bce-request-id: 367d9dd5-4513-412d-a4f8-eacfe37dbaf3
Content-Type: application/json;charset=UTF-8
```

6.6 Principal

6.6.1 获取principal列表

相对URI	HTTP 方式
/v1/endpoint/{endpointName}/principal? thingName={thingName}	GET

请求参数

名称	类型	是否必选	默认值	说明
endpointName	String	Y	-	endpoint 的名称
thingName	String	N	-	principal 所属的 thing 名称，可选参数，可查询与指定 thing 绑定的 principal
order	ENUM['desc' , 'asc']	N	desc	排序的方式，不区分大小写
orderBy	String	N	createTime	另外一个支持排序的字段是 name。
pageNo	Int	N	1	页码
pageSize	Int	N	50	每页 item 个数，最大值200
q	String	N	-	模糊查询的内容。目前支持 name 字段模糊查询

返回参数

名称	类型	含义
principalName	String	principal 名称
endpointName	String	所属的 Endpoint 名称
createTime	String	创建时间

请求示例

```
GET /v1/endpoint/endpoint-1/principal HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: e20e06f3-d0da-4668-ac67-ce62047e3beb
Content-Type: application/json;charset=UTF-8
{
  "totalCount": 1,
  "result": [
    {
      "principalName": "principal-1",
      "endpointName": "endpoint-1",
      "createTime": "2016-08-31T06:09:29Z"
    }
  ],
  "order": "desc",
  "orderBy": "createtime",
  "pageSize": 50,
  "pageNo": 1
}
```

6.6.2 获取指定的principal信息

相对URI	HTTP 方式
/v1/endpoint/{endpointName}/principal/{principalName}	GET

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint名称
principalName	String	Y	principal名称

返回参数

名称	类型	含义
principalName	String	principal名称
endpointName	String	所属的Endpoint名称
createTime	String	创建时间

请求示例

```
GET /v1/endpoint/endpoint-1/principal/principal-1 HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: e294804d-d946-4a89-a671-317d4eaeae6c
Content-Type: application/json;charset=UTF-8
{
  "principalName": "principal-1",
  "endpointName": "endpoint-1",
  "createTime": "2016-08-31T06:09:29Z"
}
```

6.6.3 创建principal

相对URI	HTTP 方式
/v1/endpoint/{endpointName}/principal	POST

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint名称
principalName	String	Y	principal名称

返回参数

名称	类型	含义
principalName	String	principal名称
endpointName	String	所属的Endpoint
password	String	principal密钥
privateKey	String	PEM格式的私钥
cert	String	PEM格式的证书
createTime	String	创建时间

请求示例

```
POST /v1/endpoint/endpoint-1/principal HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
{
  "principalName": "principal-1"
}
```

返回示例

```
HTTP/1.1 201 Created
x-bce-request-id: 9f913092-f0ef-4018-9055-e02f2db4464f
Content-Type: application/json;charset=UTF-8
{
  "principalName": "principal-1",
  "endpointName": "endpoint-1",
  "privateKey": "-----BEGIN RSA PRIVATE KEY-----  \r\nMIICWwIBAAKBgQC98S6X8nCB/
3AdsK3uXpx7YfCP/.....-----END RSA PRIVATE KEY-----\r\n",
  "password": "XM5kW9bZhRTWc/mag8TYZuVlUfdaLUju+kdwh9dfGKo=",
  "createTime": "2016-08-31T06:09:28Z",
  "cert": "-----BEGIN CERTIFICATE-----  \r\nMIIC4zCCAcCCQC3v0MerfCjdjANBgkqhkiG9w0BAQsFADBqMQs
\r      .....+LIJhgeE/2s67BdGEGH0jWhtdi1KT1   Og1A/t\r\n\r\nqhn/EexhgR6CgLNmxV4+tPN2CrJtL6c6j2moVq5dtP0g6V
\r\n\r\n1I0dXSIO1qwOmwBYBPBmVQlQS1lKaA=\r\n\r\n-----END CERTIFICATE-----\r\n"
}
```

6.6.4 重新生成密钥

相对URI	HTTP 方式
/v1/endpoint/{endpointName}/principal/{principalName}	POST

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint名称
principalName	String	Y	principal名称
target	String	N	可选项: all, password, cert

返回参数

名称	类型	含义
principalName	String	principal名称
endpointName	String	所属的Endpoint
password	String	principal密钥
privateKey	String	PEM格式的私钥
cert	String	PEM格式的证书

请求示例

```
POST /v1/endpoint/endpoint-1/principal/principal-1 HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
```

响应示例

```
HTTP/1.1 201 Created
x-bce-request-id: 9f913092-f0ef-4018-9055-e02f2db4464f
Content-Type: application/json;charset=UTF-8
{
  "principalName": "principal-1",
  "endpointName": "endpoint-1",
```

```
    "privateKey": "-----BEGIN RSA PRIVATE KEY-----\\r\\nMIICXgIBAAKBgQCuoZEOP+3c/ur8x0bVzVFHBAPx2gCi1X+r/q/5B4EDDp33+wZe\\r\\n+tJ6MfbkgZbvNxEZ...\\n88WhhVIDdneYTe8wIJ1PVBZG0kv6c\\r\\n-----END RSA PRIVATE KEY-----\\r\\n",
    "password": "pN04VRhK2h0BJpp3ayyHuy+dY0t7HFA1W1rtnoRJ76g=",
    "cert": "-----BEGIN CERTIFICATE-----\\r\\nMIIC4zCCAcsCCQDHBYYIQT8cajANBgkqhkiG9w0BAQsFADBqMQswCQYJKoZIhvcNAQELAQE\\r\\n...\\nrSe0T4miCW7kXLNwqNW7SwB7NZ4GgG8nsK1lFSfx2hK7a41vvEl8+xiszhJlBHhF\\r\\n\\nAGhpPDgBUcgy77G4W3+kg0XRdVm77vI=\\r\\n-----END CERTIFICATE-----\\r\\n"
  }
}
```

6.6.5 删除principal

相对URI	HTTP 方式
/v1/endpoint/{endpointName}/principal/{principalName}	DELETE

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint名称
principalName	String	Y	principal名称

返回参数

无特殊返回参数。

请求示例

```
DELETE /v1/endpoint/endpoint-1/principal/principal-1 HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
```

返回示例

```
HTTP/1.1 204 No Content
x-bce-request-id: 7dda6e6a-08cd-4301-90df-a5c0f7a502e8
Content-Type: application/json;charset=UTF-8
```

6.7 Policy

6.7.1 获取policy列表

相对URI	HTTP 方式
/ v1/ endpoint/ {endpointName}/ policy? principalName={principalName}	GET

请求参数

名称	类型	是否必选	默认值	说明
endpointName	String	Y	-	endpoint名称
principalName	String	N	-	Policy 所 属 的 principal名称
order	ENUM['desc' , 'asc']	N	desc	排序的方式， 不区分大小写
orderBy	String	N	createTime	另外一个支持 排序的字段是 name
pageNo	Int	N	1	页码
pageSize	Int	N	50	每 页 item 个 数，最大值200
q	String	N	-	模糊查询的内 容。目前支持 name字段模糊 查询

返回参数

名称	类型	含义
policyName	String	policy名称
endpointName	String	所属的Endpoint
createTime	String	创建时间

请求示例

GET /v1/endpoint/endpoint-1/policy HTTP/1.1

```
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 9bcf7816-f997-4152-ad50-7c06dbb41bae
Content-Type: application/json;charset=UTF-8
{
  "totalCount": 1,
  "result": [
    {
      "endpointName": "endpoint-1",
      "policyName": "policy-1",
      "createTime": "2016-08-31T06:26:40Z"
    }
  ],
  "order": "desc",
  "orderBy": "createtime",
  "pageSize": 50,
  "pageNo": 1
}
```

6.7.2 获取指定的policy信息

相对URI	HTTP 方式
/ v1/ endpoint/ {endpointName}/ policy/ {policyName}	GET

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint名称
policyName	String	Y	policy名称

返回参数

名称	类型	含义
policyName	String	policy名称

名称	类型	含义
endpointName	String	所属的endpoint名称
createTime	String	创建时间

请求示例

```
GET /v1/endpoint/endpoint-1/policy/policy-1 HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 6f0106e3-06e0-4eb3-844e-02284539d14e
Content-Type: application/json;charset=UTF-8
{
  "endpointName": "endpoint-1",
  "policyName": "policy-1",
  "createTime": "2016-08-31T06:26:40Z"
}
```

6.7.3 创建policy

相对URI	HTTP 方式
/v1/endpoint/{endpointName}/policy	POST

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint名称
policyName	String	Y	policy名称

返回参数

名称	类型	含义
policyName	String	policy名称
endpointName	String	所属的endpoint名称
createTime	String	创建时间

请求示例

```
POST /v1/endpoint/endpoint-1/policy HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
{
  "policyName": "policy-1"
}
```

响应示例

```
HTTP/1.1 201 Created
x-bce-request-id: e9f39305-d67c-45ee-9130-5e50e567fc8d
Content-Type: application/json;charset=UTF-8
{
  "endpointName": "endpoint-1",
  "policyName": "policy-1",
  "createTime": "2016-08-31T06:26:40Z"
}
```

6.7.4 删除policy

相对URI	HTTP 方式
/ v1/ endpoint/ {endpointName}/ policy/ {policyName}	DELETE

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint名称
policyName	String	Y	policy名称

返回参数

无特殊返回参数。

请求示例

```
DELETE /v1/endpoint/endpoint-1/princy/princy-1 HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
```

返回示例

```
HTTP/1.1 204 No Content
x-bce-request-id: e2e12a78-32cf-4da5-8fd8-87d7f76e3295
Content-Type: application/json;charset=UTF-8
```

6.8 Permission

6.8.1 获取policy下所有topic信息

相对URI	HTTP 方式
/v1/endpoint/{endpointName}/permission?policyName={policyName}	GET

请求参数

名称	类型	是否必选	默认值	说明
endpointName	String	Y	-	endpoint名称
policyName	String	Y	-	policy名称
order	ENUM['desc' , 'asc']	N	desc	排序的方式，不区分大小写
orderBy	String	N	createTime	仅支持基于createTime排序
pageNo	Int	N	1	页码
pageSize	Int	N	50	每页item个数，最大值200

返回参数

名称	类型	含义
operations	List[ENUM]	允许的操作 list
permissionUuid	String	permission的ID
policyUuid	String	policy的ID
topic	String	操作对应的Topic
createTime	String	创建时间

请求示例

```
GET /v1/endpoint/endpoint-1/permission?policyName=policy-1 HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: eaa235de-f71c-4ed4-baa0-8b380ffb90ba
Content-Type: application/json;charset=UTF-8
{
  "totalCount": 1,
  "result": [
    {
      "policyUuid": "f1615eb2-9ff7-439f-b9db-8c2c5a5476b9",
      "operations": [
        "PUBLISH",
        "SUBSCRIBE"
      ],
      "topic": "topic1",
      "createTime": "2016-08-31T06:44:01Z",
      "permissionUuid": "ba8313a8-b2ed-4079-8160-00fc168d6d9c"
    }
  ],
  "order": "desc",
  "orderBy": "createtime",
  "pageSize": 50,
  "pageNo": 1
}
```


6.8.2 获取指定topic的信息

相对URI	HTTP 方式
/v1/endpoint/{endpointName}/permission/{permissionUuid}	GET

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint名称
permissionUuid	String	Y	permission的ID

返回参数

名称	类型	含义
operations	List[ENUM]	允许的操作 list
permissionUuid	String	permission的ID
policyUuid	String	policy的ID
topic	String	操作对应的Topic
createTime	String	创建时间

请求示例

```
GET /v1/endpoint/endpoint-1/permission/ba8313a8-b2ed-4079-8160-00fc168d6d9c HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 139eedb4-14f9-4512-94fb-bbac2b334bf3
Content-Type: application/json;charset=UTF-8
{
  "policyUuid": "f1615eb2-9ff7-439f-b9db-8c2c5a5476b9",
  "operations": [
```

```

        "PUBLISH",
        "SUBSCRIBE"
    ],
    "topic": "topic1",
    "createTime": "2016-08-31T06:44:01Z",
    "permissionUuid": "ba8313a8-b2ed-4079-8160-00fc168d6d9c"
}

```

6.8.3 在policy下设置topic

相对URI	HTTP 方式
/v1/endpoint/{endpointName}/permission	POST

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint名称
policyName	String	Y	所属的Policy名称
operations	List[ENUM]	Y	允许的操作list
topic	String	Y	操作对应的Topic

返回参数

名称	类型	含义
operations	List[ENUM]	允许的操作 list
permissionUuid	String	permission的ID
policyUuid	String	policy的ID
topic	String	操作对应的Topic
createTime	String	创建时间

请求示例

```

POST /v1/endpoint/endpoint-1/permission HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8

```

```
{
  "policyName": "policy-1"
  "operations": ["PUBLISH", "SUBSCRIBE"]
  "topic": "topic1"
}
```

返回示例

HTTP/1.1 201 Created

x-bce-request-id: 6fb3044f-b5ef-4a8f-a416-51fa0e48f510

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

```
{
  "policyUuid": "f1615eb2-9ff7-439f-b9db-8c2c5a5476b9",
  "operations": [
    "PUBLISH",
    "SUBSCRIBE"
  ],
  "topic": "topic1",
  "createTime": "2016-08-31T06:44:01Z",
  "permissionUuid": "ba8313a8-b2ed-4079-8160-00fc168d6d9c"
}
```

6.8.4 更新已有的topic设置

相对URI	HTTP 方式
/ v1/ endpoint/ {endpointName}/ permis- sion/{permissionUuid}	PUT

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint名称
permissionUuid	String	Y	permission的ID
operations	List[ENUM]	Y	允许的操作list
topic	String	Y	操作对应的Topic

返回参数

名称	类型	含义
operations	List[ENUM]	允许的操作 list
permissionUuid	String	permission的ID
policyUuid	String	policy的ID
topic	String	操作对应的Topic
createTime	String	创建时间

请求示例

```
PUT /v1/endpoint/endpoint-1/permission/ba8313a8-b2ed-4079-8160-00fc168d6d9c HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
{
  "operations":["PUBLISH"]
  "topic":"topic2"
}
```

返回示例

```
HTTP/1.1 201 Created
x-bce-request-id: 853459a6-b933-4546-8bac-8a174516e83f
Content-Type: application/json;charset=UTF-8
{
  "policyUuid": "f1615eb2-9ff7-439f-b9db-8c2c5a5476b9",
  "operations": [
    "PUBLISH"
  ],
  "topic": "topic2",
  "createTime": "2016-08-31T06:44:01Z",
  "permissionUuid": "ba8313a8-b2ed-4079-8160-00fc168d6d9c"
}
```

6.8.5 删除已有的topic

相对URI	HTTP 方式
/v1/endpoint/{endpointName}/permission/{permissionUuid}	DELETE

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint名称
permissionUuid	String	Y	permission的ID

返回参数

无特殊返回参数。

请求示例

```
DELETE /v1/endpoint/endpoint-1/permission/ba8313a8-b2ed-4079-8160-00fc168d6d9c HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/plain;charset=UTF-8
```

返回示例

```
HTTP/1.1 204 No Content
x-bce-request-id: 44dcf34e-bc58-46a8-8492-cde566c69328
Content-Type: application/json;charset=UTF-8
```

6.9 Client

6.9.1 获取指定MQTT客户端在线状态

相对URI	HTTP 方式
/v2/endpoint/{endpointName}/client/{clientId}/status/online	GET

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint名称
clientId	String	Y	MQTT客户端ID

请求示例

```
GET /v2/endpoint/endpoint-1/client/abc/status/online HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/json;charset=UTF-8
```

返回示例

```
HTTP/1.1 200 Ok
x-bce-request-id: 367d9dd5-4513-412d-a4f8-eacfe37dbaf3
Content-Type: application/json;charset=UTF-8

true
```

6.9.2 获取所有MQTT客户端在线状态

相对URI	HTTP 方式
/ v2/ endpoint/ {endpointName}/ batch-client-query/status	POST

请求参数

名称	类型	是否必选	含义
endpointName	String	Y	endpoint名称
mqttID	Array	Y	MQTT客户端ID列表

返回参数

名称	类型	含义
clientId	String	MQTT客户端ID
online	Boolean	在线状态

请求示例

```
POST /v2/endpoint/endpoint-1/batch-client-query/status HTTP/1.1
host: iot.gz.baidubce.com
authorization: {authorization}
content-type: text/json;charset=UTF-8

["clientId1", "clientId2"]
```

返回示例

```
HTTP/1.1 200 Ok
x-bce-request-id: 367d9dd5-4513-412d-a4f8-eacfe37dbaf3
Content-Type: application/json;charset=UTF-8

[{"clientId": "clientId1", "online": true}, {"clientId": "clientId2", "online": false}]
```

6.9.3 Publish Message

MQTT客户端可以调用此API接口，以HTTP方式接入，实现消息发布（不支持消息订阅）。

调用此API不需要携带百度云签名认证字符串。

注意

建议使用HTTPS访问，HTTP用户名及密码泄露。

URI	HTTP 方式	Content Type	Return Type
https://api.mqtt.iot.gz.baidubce.com/v1/ proxy? qos=0&topic=topic1&retain=false	POST	application/ octet-stream	application/ json;charset=UTF-8

请求参数

名称	是否必选	含义
qos	Y	该消息的QoS取值，可选0或1
topic	Y	topic名称，客户端将向指定的topic发布消息

名称	是否必选	含义
retain	N	retain标记, 详细介绍请参看 保留消息

请求头参数

名称	是否必选	含义
auth.username	Y	用户名, 即: 实例名称/设备名称
auth.password	Y	密码, 即: 创建身份时获得的密钥

发布的具体消息内容放在HTTP Content中, 可以是二进制消息。消息长度不大于32K。

请求示例

```
POST /v1/proxy?qos=0&topic=topic1&retain=false HTTP/1.1
```

```
host: api.mqtt.iot.gz.baidubce.com
```

```
content-type: application/octet-stream
```

```
auth.username: test/test
```

```
auth.pasword: cNCcTh+xkth7Jk2EHwUo2+IhkMsDnQlmfMi93CsifY=
```

成功: 201, 失败: 4XX or 5XX

第7章

Java SDK文档

7.1 目录

- [概述](#)
- [安装SDK工具包](#)
- [快速入门](#)
- [创建IotHubClient](#)
- [endpoint操作](#)
 - [创建endpoint](#)
 - [获取endpoint列表](#)
 - [查看指定的endpoint信息](#)
 - [删除endpoint](#)
- [thing操作](#)
 - [创建thing](#)
 - [查看thing](#)
 - [删除thing](#)
- [principal操作](#)
 - [创建principal](#)
 - [查看principal](#)
 - [绑定指定的thing和principal](#)
 - [重新获得principal的证书和密码](#)
 - [删除principal](#)
- [policy操作](#)
 - [创建policy](#)
 - [查看policy](#)
 - [绑定指定的principal和policy](#)
 - [删除policy](#)

- [permission操作](#)
 - [创建permission](#)
 - [更新permission](#)
 - [查看permission](#)
- [版本说明](#)
 - [v0.10.13](#)

7.2 概述

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

7.3 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
│   ├── iothub         //IoT服务相关类
│   └── model           //IoT内部model，如Request或
Response

```

```

|           └─ IotHubClient.class           //IoT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

```

7.4 快速入门

通过Java SDK创建一个物接入应用的流程如下：

1. [创建IotHubClient](#)
2. [创建endpoint](#)
3. [创建thing](#)
4. [创建principal](#)
5. [绑定指定的thing和principal](#)
6. [创建policy](#)
7. [绑定指定的principal和policy](#)
8. [创建permission](#)

7.5 创建IotHubClient

用户可以参考如下代码新建一个IotHubClient：

```

// 初始化相关参数
String AK = "ACCESS_KEY_ID";
String SK = "SECRET_ACCESS_KEY";
String ENDPOINT = "iot.gz.baidubce.com";

//使用物接入相关参数
String TEST_ENDPOINT_NAME = "sdk_test_endpoint_01";
String TEST_THING_NAME = "sdk_test_thing_01";
String TEST_PRINCIPAL_NAME = "sdk_test_principal_01";
String TEST_POLICY_NAME = "sdk_test_policy_01";
String TEST_TOPIC = "abc";

//创建物接入client

```

```
BceClientConfiguration config = new BceClientConfiguration()
    .withCredentials(new DefaultBceCredentials(AK, SK))
    .withEndpoint(ENDPOINT);
IotHubClient iotHubClient = new IotHubClient(config);
```

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

参数说明

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

参数	说明
connectionTimeo utInMillis	建立连接的超时时间（单位：毫秒）
localAddress	本地地址
maxConnections	允许打开的最大HTTP连接数
proxyDomain	访问NTLM验证的代理服务器的Windows域名
proxyHost	代理服务器主机地址
proxyPassword	代理服务器验证的密码
proxyPort	代理服务器端口
proxyPreemptive Authentica tio nEnabled	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
socketBufferSiz eInBytes	Socket缓冲区大小
socketTimeoutIn Millis	通过打开的连接传输数据的超时时间（单位：毫秒）
userAgent	用户代理，指HTTP的User-Agent头

7.6 endpoint操作

7.6.1 创建endpoint

创建endpoint之前应先[创建IotHubClient](#)。

用户可以参考如下代码新建一个endpoint:

```
//用一个字符串创建endpoint  
QueryEndpointResponse responseQuery = iotHubClient.createEndpoint(TEST_ENDPOINT_NAME);
```

7.6.2 获取endpoint列表

用户可以参考如下代码获取endpoint列表:

```
//列出该用户下所有endpoint相关信息 (ps. 所有的list操作均可加分页参数)  
ListResponse responseList = iotHubClient.listEndpoints();
```

7.6.3 查看指定的endpoint信息

用户可以参考如下代码查看endpoint:

```
//列出该用户下指定名字的endpoint相关信息  
QueryEndpointResponse responseQuery = iotHubClient.queryEndpoint(TEST_ENDPOINT_NAME);
```

7.6.4 删除endpoint

用户可以参考如下代码删除endpoint:

```
//删除endpoint  
BaseResponse responseBase = iotHubClient.deleteEndpoint(TEST_ENDPOINT_NAME);
```

7.7 thing操作

7.7.1 创建thing

创建thing之前应先[创建endpoint](#)。

请参考以下代码创建thing:

```
//在指定的endpoint下面创建thing  
QueryThingResponse responseQuery = iotHubClient.createThing(TEST_ENDPOINT_NAME, TEST_THING_NAME);
```

7.7.2 查看thing

请参考以下代码查看thing:

```
//列出指定endpoint下所有的thing
ListResponse responseList = iotHubClient.listThings(TEST_ENDPOINT_NAME);

//列出指定的某个endpoint下指定的thing的相关信息
QueryThingResponse responseQuery = iotHubClient.queryThing(TEST_ENDPOINT_NAME, TEST_THING_NAME);
```

7.7.3 删除thing

请参考以下代码删除thing:

```
//删除指定的thing
BaseResponse responseBase = iotHubClient.deleteThing(TEST_ENDPOINT_NAME, TEST_THING_NAME);
```

7.8 principal操作

7.8.1 创建principal

创建principal之前应先[创建endpoint](#)。

请参考以下代码创建principal:

```
//创建principal (response里有证书、密码)
CreatePrincipalResponse responseCreate = iotHubClient.createPrincipal(TEST_ENDPOINT_NAME, TEST_PRIN
```

7.8.2 查看principal

请参考以下代码查看principal:

```
//列出所有的principal
ListResponse responseList = iotHubClient.listPrincipals(TEST_ENDPOINT_NAME);
```

查看指定thing下的principal。如果thing下没有绑定principal，以下操作返回的结果为空。

```
//列出指定thing下面所有的principal
ListResponse responseList = iotHubClient.listPrincipals(TEST_ENDPOINT_NAME, TEST_THING_NAME);
```

7.8.3 绑定指定的thing和principal

请参考以下代码绑定指定的thing和principal:

```
//绑定指定的thing和principal
IoTClient.attachThingToPrincipal(TEST_ENDPOINT_NAME, TEST_THING_NAME, TEST_PRINCIPAL_NAME);

//解除thing和principal的绑定关系
BaseResponse response = IoTClient.removeThingToPrincipal(TEST_ENDPOINT_NAME, TEST_THING_NAME, TEST_PRINCIPAL_NAME);
```

7.8.4 重新获得principal的证书和密码

请参考以下代码重新获得principal的证书和密码:

```
//重新获取principal, 可重新获得证书、密码
CreatePrincipalResponse responseCreate = IoTClient.regenerateCert(TEST_ENDPOINT_NAME, TEST_PRINCIPAL_NAME);
```

7.8.5 删除principal

请参考以下代码绑定删除principal:

```
//删除principal
BaseResponse responseBase = IoTClient.deletePrincipal(TEST_ENDPOINT_NAME, TEST_PRINCIPAL_NAME);
```

7.9 policy操作

7.9.1 创建policy

请参考以下代码创建policy:

```
//创建policy
QueryPolicyResponse responseQuery = IoTClient.createPolicy(TEST_ENDPOINT_NAME, TEST_POLICY_NAME);
```

7.9.2 查看policy

请参考以下代码查看policy:

```
//列出指定endpoint下所有policy
ListResponse responseList = IoTClient.listPolicy(TEST_ENDPOINT_NAME);
```

```
//列出指定principal下的所有policy
responseList = iotHubClient.listPolicy(TEST_ENDPOINT_NAME, TEST_PRINCIPAL_NAME);

//获取指定的policy信息
QueryPolicyResponse responseQuery = iotHubClient.queryPolicy(TEST_ENDPOINT_NAME, TEST_POLICY_NAME);
```

7.9.3 绑定指定的principal和policy

请参考以下代码绑定指定的principal和policy:

```
//绑定指定的principal和policy
iotHubClient.attachPrincipalToPolicy(TEST_ENDPOINT_NAME, TEST_PRINCIPAL_NAME, TEST_POLICY_NAME);

//解除指定的principal和policy的绑定关系
BaseResponse response = iotHubClient.removePrincipalToPolicy(TEST_ENDPOINT_NAME, TEST_PRINCIPAL_NAME, TEST_POLICY_NAME);
```

7.9.4 删除policy

请参考以下代码删除指定的policy:

```
//删除指定的policy
BaseResponse responseBase = iotHubClient.deletePolicy(TEST_ENDPOINT_NAME, TEST_POLICY_NAME);
```

7.10 permission操作

7.10.1 创建permission

创建permission之前,必须先完成以下操作:

1. [创建endpoint](#)
2. [创建thing](#)
3. [创建principal](#)
4. [绑定指定的thing和principal](#)
5. [创建policy](#)
6. [绑定指定的principal和policy](#)

请参考以下代码创建permission:

```
//准备operation参数,可以添加"PUBLISH"或"SUBSCRIBE",也可以都加
```



```
List<Operation> operations = new ArrayList<Operation>();
operations.add(Operation.PUBLISH);
operations.add(Operation.SUBSCRIBE);

//创建permission参数包括endpoint、policy、操作类型、topic, response里有permissionUuid
QueryPermissionResponse response = iotHubClient.createPermission(TEST_ENDPOINT_NAME,
    TEST_POLICY_NAME,
    operations,
    TEST_TOPIC);

String permissionUuid = response.getPermissionUuid();
```

7.10.2 更新permission

请参考以下代码更新permission：

```
//更新permission, 不需要更新的参数填null,
QueryPermissionResponse response = iotHubClient.updatePermission(TEST_ENDPOINT_NAME, permissionUuid)
```

7.10.3 查看permission

请参考以下代码查看permission：

```
//列出指定policy下所有permission
ListPermissionResponse responseList = iotHubClient.listPermission(TEST_ENDPOINT_NAME, TEST_POLICY_NAME);

//获取指定permission信息
QueryPermissionResponse responseQuery = iotHubClient.queryPermission(TEST_ENDPOINT_NAME, permissionUuid)
```

7.11 版本说明

7.11.1 v0.10.13

首次发布。

第8章

常见问题

8.1 目录

- 产品配置操作问题
 - 物接入中是否能批量创建设备？
 - 物接入实例列表中为什么出现了不是我自己创建的实例？
 - 物接入的数据能存储到哪里？
 - 物接入中数据存储主题无法填写？
- 产品规格及使用限制
 - 物接入上传的消息大小有限制吗？
 - 每个百度云账号可以创建多少个实例？
 - 每个实例下可以创建多少个策略、身份和设备？
 - 每个实例下允许的最大连接数？
 - 每个实例下的物接入设备与实际连接数是什么关系，限额都是多少？
- 计费相关问题
 - 物解析和物管理的实例中收发的消息数也算在物接入的计费套餐里吗？
- 客户端及MQTT SDK相关问题
 - 请问是否有基于FreeRTOS或者RTX操作系统的MQTT SDK源码？
 - 为什么publish的QOS等级为2后，则会断开服务？
 - 物接入是否支持消息缓存？
 - 与物接入服务连接成功，往一个主题发送消息，就直接断开。
 - 连接物接入服务时，出现连接协议错误。
 - 遗嘱消息的触发条件有哪些？
- API使用问题
 - 使用API连接物接入时，返回connection timeout错误。
 - 使用API连接物接入时，返回invalid ClientID

8.2 产品配置操作问题

8.2.1 物接入中是否能批量创建设备？

物接入中可以通过调用open API来批量创建设备，查看<https://cloud.baidu.com/doc/IOT/API文档>。

8.2.2 物接入实例列表中为什么出现了不是我自己创建的实例？

物管理和物解析在使用时，也会在物接入中创建实例，每个用户只会在开始使用物解析和物管理时创建一个物解析实例和物管理实例，物解析的实例名是parser_endpointxxxx，物管理的实例名是32位数字或字母。

8.2.3 物接入的数据能存储到哪里？

物接入的数据可以直接存储到BOS和Kafka里；如果物接入上传的是json格式的数据，也可以通过规则引擎设置规则，存储到时序数据库、Kafka和转发至另一个物接入主题。

8.2.4 物接入中数据存储主题无法填写？

物接入的存储需要结合百度云的其他产品使用，如百度Kafka和BOS，请先开通相对应的服务。

8.3 产品规格及使用限制

8.3.1 物接入上传的消息大小有限制吗？

物接入上传的单条消息大小限制是32KB，超过32KB的消息会被丢弃。如果用户上传的消息大于32KB，请在上传前将消息切割成小包。

但在计费上，每512Bytes算一条。用户使用的消息条数=实际发送长度/512Bytes，计算结果向上取整。

8.3.2 每个百度云账号可以创建多少个实例？

每个百度云账号最多可以创建100个实例。如果需要更多配额，请提交工单申请。

8.3.3 每个实例下可以创建多少个策略、身份和设备？

每个实例下可创建10000个设备、10000个策略和10000个身份。如果需要更多配额，请提交工单申请。

8.3.4 每个实例下允许的最大连接数？

单实例最大并发连接数：10000个，即最多可同时连接10000个实体设备。

8.3.5 每个实例下的物接入设备与实际连接数是什么关系，限额都是多少？

每个实例下的设备对应一套独立的用户名和密钥，每个实例下最多可创建10000个设备。用户可以使用同一套用户名+身份密钥，建立多个连接。每个实例最多同时连接10000个实体设备，这10000个实体设备可以对应相同的物接入设备，即所有设备共享相同的用户名+身份密钥；也可以对应不同的物接入设备。

8.4 计费相关问题

8.4.1 物解析和物管理的实例中收发的消息数也算在物接入的计费套餐里吗？

是的。

8.5 客户端及MQTT SDK相关问题

8.5.1 请问是否有基于FreeRTOS或者RTX操作系统的MQTT SDK源码？

通过paho官网查看，提供了多个版本的C/C++ client，有的是支持posix标准的（unix、linux、windows），有的是支持嵌入式系统的。可以参考这个连接：<https://www.eclipse.org/paho/clients/c/embedded/>

8.5.2 为什么publish的QOS等级为2后，则会断开服务？

目前暂不支持Qos=2的服务。

8.5.3 物接入是否支持消息缓存？

支持。当客户端连接物接入服务时，如果Clean Session位被设置为false，则该连接被认为是持久连接，其具体表现为：当该客户断开后，任何订阅的主题和QoS被设置为1或2的信息都会保存，直到该客户端再次连接上server端，物接入服务支持将该消息保留24小时。若“clean session”被设置为true，当该客户断开后，所有的订阅主题都会被移除。

本方法不推荐大量使用，如果需要保存大量数据，推荐将物接入的数据转发到Kafka。

8.5.4 与物接入服务连接成功，往一个主题发送消息，就直接断开。

由于策略权限设置没有包括这个主题的pub权限，需要修改策略来解决权限问题。

8.5.5 连接物接入服务时，出现连接协议错误。

大多是由于选择的端口和期望的协议不一致，正确端口号是TCP：1883， TLS：1884，Websocket:8884。

8.5.6 遗嘱消息的触发条件有哪些？

遗嘱（Will Message）消息必须被存储在服务端并且与这个网络连接关联。网络连接关闭时，服务端必须发布这个遗嘱消息，除非服务端收到Client发送的DISCONNECT报文。

遗嘱消息发布的条件，包括但不限于：

- 服务端检测到了一个I/O错误或者网络故障。
- 客户端在保持连接（Keep Alive）的时间内未能通讯。
- 客户端没有先发送DISCONNECT报文直接关闭了网络连接。
- 由于协议错误服务端关闭了网络连接。

8.5.7 MQTT客户端网络连接异常，但在keepalive时间内恢复，客户端是否需要重新建立MQTT连接？

需要，如果MQTT客户端异常断开，都需要重新建立MQTT连接并重新订阅主题。

如果MQTT客户端cleansession=false，连接异常断开后，服务器会维护session信息，重新连接后不需要再订阅主题。

8.6 API使用问题

8.6.1 使用API连接物接入时，返回connection timeout错误。

1. 当前客户端的网络不通，需要测试客户端网络是不是好的，可以先访问一下一些公网的网站，比如www.baidu.com，如果访问成功的话，就表示网络ok，如果这个时候还是出现这个错误，估计有可能是防火墙关闭了特定端口，目前我们的MQTT提供服务的端口包括1883 (tcp)，1884 (tls)，8884 (websocket)，简单的测试办法就是telnet xxx.mqtt.iot.gz.baidubce.com port，其中xxx.mqtt.iot.gz.baidubce.com是IoT Hub返回的域名，port是使用协议对应的端口，如果是TCP，就用1883。
2. 如果客户端网络没有问题的话，telnet也不通，这个时候就可能是电信运营商的问题，电信运营商有时候会把特定IP的端口给封闭，导致这些服务受到影响，你可能需要获取解析的IP地址，然后自己一个一个的尝试，获取解析的IP地址：nslookup xxx.mqtt.iot.gz.baidubce.com，然后使用telnet来测试：telnet xxxx_ip port。
3. 如果每一个IP的telnet连接都是失败的话，这个时候很可能就是云端的服务出问题，你需要告知百度云的技术支持团队，可以提交工单，一般来说这种情况的可能性极低。

8.6.2 使用API连接物接入时，返回invalid ClientID

1. 填写的clientID不合法，我们ClientID支持的长度是128，超过之后会报错，clientID的格式必须是下面这些字符的组合” 0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ-”
2. 还有一种情况，就是我们的服务有并发连接数限制，当你的并发连接数超过上限的时候（目前是每个实例最大10000个并发连接），我们会返回这个错误给客户端。

第9章

MQTT协议介绍

9.1 目录

- [关于本章](#)
- [MQTT协议是什么？](#)
- [MQTT协议如何工作？](#)
 - [如何将消息正确送达？](#)
 - [如何使用通配符订阅多个主题？](#)
 - [如何确保消息已被送达？](#)
 - [什么是临终遗嘱？](#)

9.2 关于本章

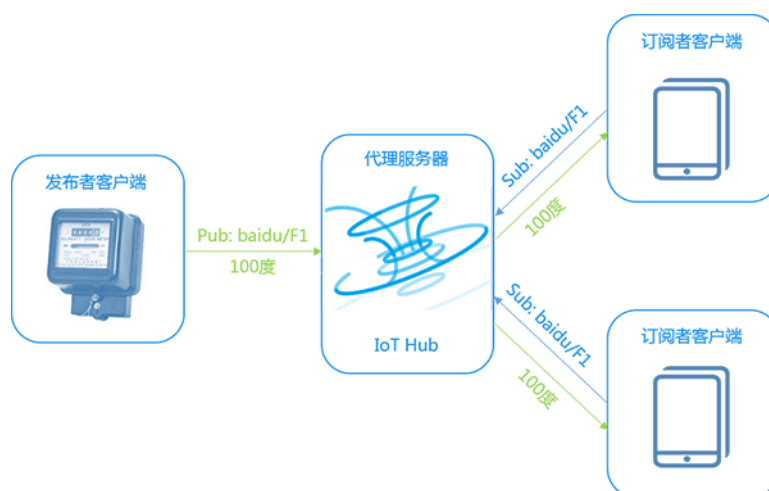
IoT Hub通过MQTT协议，在设备和云端之间建立安全的双向链接，为了帮助用户更好的使用IoT Hub，本章为用户介绍与MQTT相关的基本原理和概念。

9.3 MQTT协议是什么？

MQTT (Message Queuing Telemetry Transport Protocol) 的全称是消息队列遥感传输协议的缩写，是一种基于轻量级代理的发布/订阅模式的消息传输协议，运行在TCP协议栈之上，为其提供有序、可靠、双向连接的网络连接保证。

9.4 MQTT协议如何工作？

MQTT采用代理的发布/订阅模式实现了发布者和订阅者的解耦(decouple)，因此，在MQTT协议中有三种角色：代理服务器、发布者客户端以及订阅者客户端，其中发布者和订阅者互不干扰，也就是说发布者和订阅者互不知道对方的存在，它们只知道代理服务器，代理服务器负责将来自发布者的消息进行存储处理并将这些消息发送到正确的订阅者中去。



代理服务器 (Server)

代理服务器可以是一个程序或者设备，作为发送消息的客户端和请求订阅的客户端之间的中介。其主要作用是接收发布者客户端发布的应用信息，然后将信息转发给符合条件的订阅者客户端。

百度云天工智能物联网平台为用户提供了代理服务器的功能。

客户端 (Client)

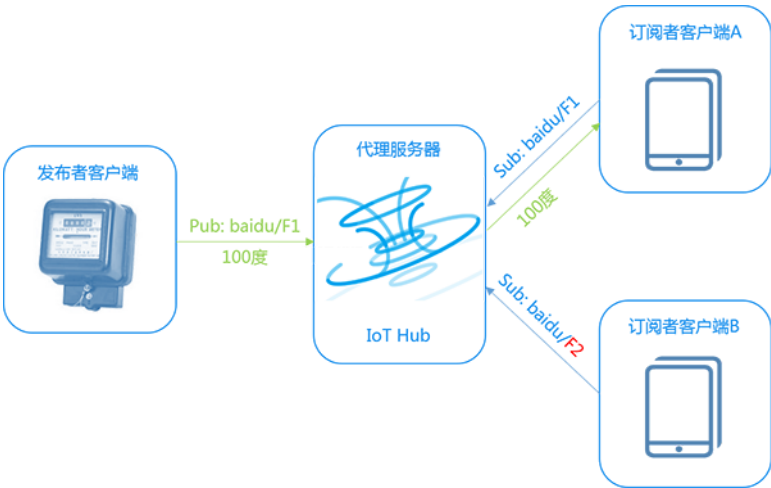
客户端指使用MQTT协议的程序或设备。客户端包括发布者客户端和订阅者客户端，同一个客户端可以即是发布者也是订阅者。客户端可以发布消息给其它相关客户端，也可以订阅其它客户端发布的消息。

因为每个用户的设备和使用场景不同，通常用户需要自己开发客户端软件。MQTT官方提供了Client SDK，可以帮助客户快速开发MQTT客户端。

百度云也为用户提供了websocket，可用于模拟客户端，帮助用户进行业务测试和验证。

9.4.1 如何将消息正确送达？

MQTT通过“主题”实现将消息从发布者客户端送达至接收者客户端。“主题”是附加在应用消息上的一个标签，发布者客户端将“主题”和“消息”发送至代理服务器，代理服务器将该消息转发至每一个订阅了该“主题”的订阅者客户端，如下图所示：



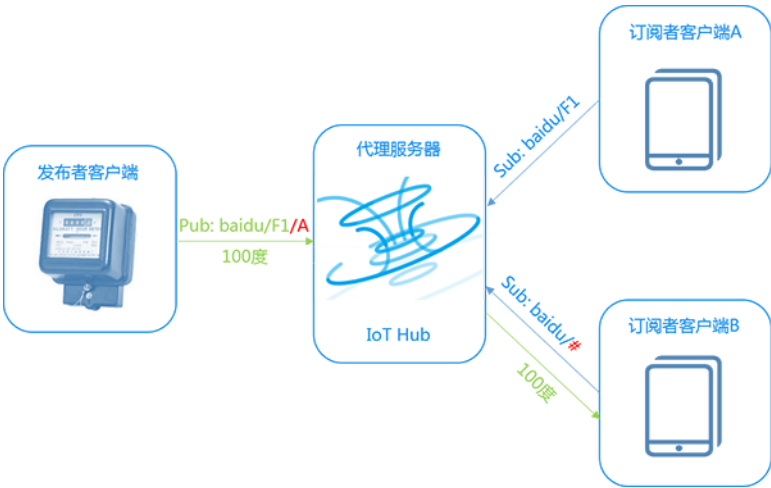
一个主题名可以由多个主题层级组成，每一层通过“/”斜杠分隔开，如上图所示，订阅者客户端A将主题过滤器设置为“baidu/F1”；订阅者客户端B将主题过滤器设置为“baidu/F2”。发布者客户端向“baidu/F1”发布消息，因此只有订阅者客户端A可以接收到该消息。

主题过滤器指客户端在订阅时包含的一个表达式，用于表示相关的一个或多个主题。

9.4.2 如何使用通配符订阅多个主题？

如果用户需要一次订阅多个具有类似结构的主题，可以在主题过滤器中包含通配符。通配符只可用在主题过滤器中，在发布应用消息时的主题名不允许包含通配符，主题通配符有两种：

- \#: 表示匹配>=0个层次，比如a/#就匹配a/b，a/b/c（不能匹配a/，后面必须有其它主题）。单独的一个#表示匹配所有，不允许a#或a/#/c等形式。
- +: 表示匹配一个层次，例如a/+匹配a/b，a/c，不匹配a/b/c。单独的一个+是允许的，但a+为非法形式。



9.4.3 如何确保消息已被送达？

发布者客户端通过设置PUBLISH报文中的QoS标志位，对于客户端发布的消息提供三种服务质量等级，如下：

- QoS=0，协议对此等级应用信息不要求回应确认，也没有重发机制，这类信息可能会发生消息丢失或重复，取决于TCP/IP提供的尽最大努力交互的数据包服务。
- 最少一次(At least once delivery)：QoS=1，确保信息到达，但消息重复可能发生，发送者如果在指定时间内没有收到PUBACK控制报文，应用信息会被重新发送。
- 仅仅一次(Exactly once delivery)：QoS=2，最高级别的服务质量，消息丢失和重复都是不可接受的。

9.4.4 什么是临终遗嘱？

MQTT协议利用KeepAlive机制在客户端异常断开时发现问题。当客户端断开时（例如：电量耗尽、系统崩溃或者网络断开），代理服务器会采取相应措施。

客户端设置“临终遗嘱”（LWT）信息后，当代理服务器检测到客户端离线后，就会发送保存在特定主题上的LWT信息，让其它订阅该主题的客户知道该节点已经意外离线。

第10章

MQTT客户端使用指南

10.1 目录

- [Websockets Client](#)
- [MQTT.fx](#)
 - [连接IoT Hub服务](#)
 - [订阅消息](#)
 - [发布消息](#)

10.2 Websockets Client

Websockets Client是百度云基于浏览器开发的MQTT客户端测试工具。用户完成物接入配置后，可以通过该工具测试连接性。

有关Websockets Client的使用方法请参看[连接测试](#)

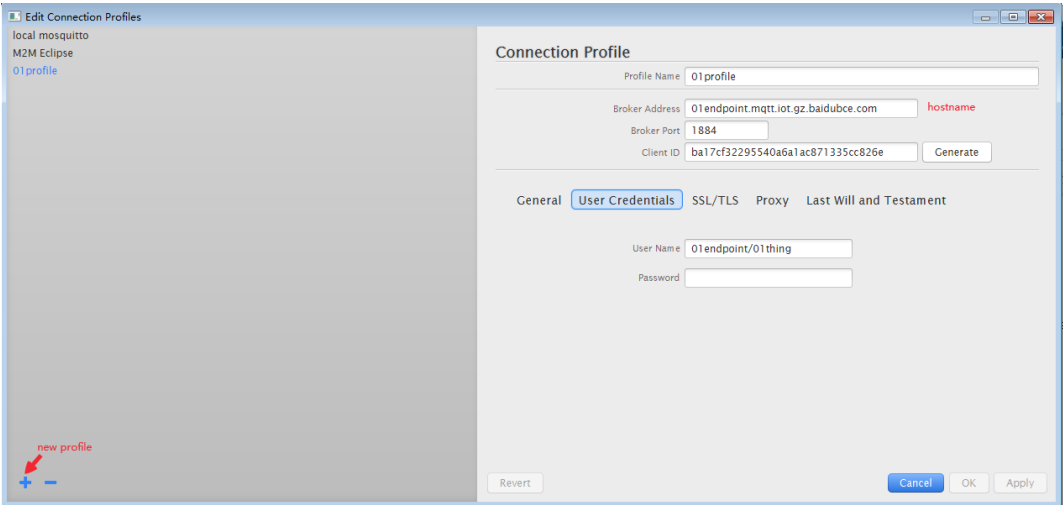
10.3 MQTT.fx

MQTT.fx 是目前主流的mqtt客户端，可以快速验证是否可以与IoT Hub 服务交流发布或订阅消息。设备将当前所处的状态作为MQTT主题发送给IoT Hub，每个MQTT主题topic具有不同等级的名称，如“建筑/楼层/温度。” MQTT代理服务器将接收到的主题topic发送给给所有订阅的客户端。

10.3.1 连接IoT Hub服务

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客户端的设置页面，点击 “+” 按键，创建一个新的配置文件。



* 填写Connection profile相关信息：

参数名称	说明
profile name	配置文件名称
Broker Address	创建endpoint后返回的hostname
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

2. 选择User Credential，输入创建 IoT Hub 服务返回的username/password，参考[配置实例](#)。
3. 如果您选择SSL安全认证方式连接IoT Hub 服务，需要配置SSL/TLS安全认证，勾选 [Enable SSL/TLS](#)，选择[CA signed server certificate](#)认证。
如您选择TCP连接，无需配置SSL安全认证，执行第4步骤即可。
4. 点击 “Apply” 按键，完成客户端配置。
5. 返回MQTT客户端界面，选择新创建的配置文件，点击 “connect” 按键连接服务。

10.3.2 订阅消息

注意：

IoT Hub一个主题支持的层级最多是9层（也就是最多只能出现8个斜线“/”）。

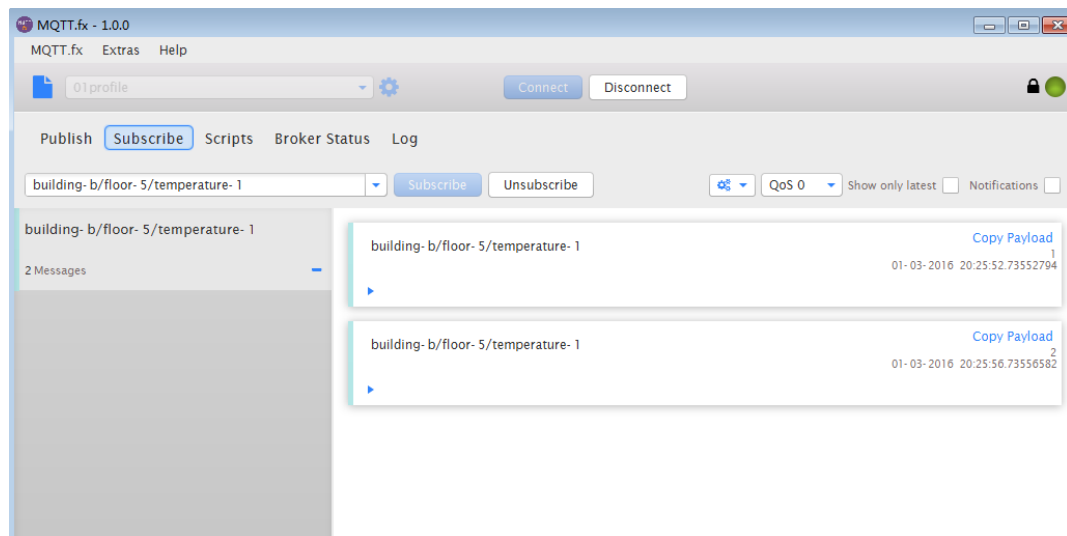
成功连接IoT Hub 服务后，即可开始订阅消息。

打开Subscribe标签，填写主题topic，例如**building-b/floor-5/temperature-1**，选择默认的QoS 0，点击“Subscribe”进行订阅操作。

10.3.3 发布消息

打开Publish标签，填写主题topic，例如**building-b/floor-5/temperature-1**，选择默认的QoS 0，点击“Publish”进行发布操作。

返回Subscribe界面，即可看到已接收的订阅消息，参见下图。



第11章

MQTT Client SDK

物接入与Paho（即MQTT Client SDK）完全兼容，如果开发者需要开发MQTT客户端，可到[Paho官方网站](#)下载SDK并获取帮助文档，详情如下：

Client	MQTT 3.1	MQTT 3.1.1	LWT	SSL/TLS	Automated Re-connect	Offline Buffering	Message Persistence	WebSocket Support	Socket MQTT Support	Standard Blocking API	Non-Blocking API	High Availability
Java	√	√	√	√	√	√	√	√	√	√	√	√
Python	√	√	√	√	√	√	×	√	√	√	√	×
JavaScript	√	√	√	√	√	√	√	√	×	×	√	√
GoLang	√	√	√	√	√	√	√	√	√	×	√	√
C	√	√	√	√	√	√	√	×	√	√	√	√
.Net (C#)	√	√	√	√	×	×	×	×	√	×	√	×
Android Service	√	√	√	√	√	√	√	√	√	×	√	√
Embedded C/C++	√	√	√	√	×	×	×	×	√	√	√	×

第12章

MQTT客户端代码示例

12.1 目录

- [C代码示例](#)
 - [下载TLS认证文件](#)
 - [下载并执行示例代码](#)
 - [C代码示例](#)
- [python代码示例](#)
 - [下载TLS认证文件](#)
 - [下载并执行示例代码](#)
- [Java代码示例](#)

12.2 C代码示例

12.2.1 下载TLS认证文件

物接入支持SSL/TLS加密传输方式，保障用户的数据传输安全。用户在执行示例代码前，需先下载TLS认证文件，并在代码中指定认证文件的存放路径。

下载[TLS认证文件](#)，并将认证文件保存至示例代码路径下。

12.2.2 下载并执行示例代码

下载[MQTT-c压缩包](#)，解压MQTT-c，目录结构如下：

MQTT-c

```
├─ include
├─ lib
├─ src
│   ├── ConnectorSync.c
│   ├── PublisherSync.c
│   └── SubscriberSync.c
├─ Makefile
└─ root_cert.pem
```

1. 打开src/PublisherSync.c，配置以下参数：

参数名称	解释
PRIVATE_FILE	输入认证文件所在目录
USER	创建物接入设备后返回的用户名，参见 创建物接入设备 中的步骤3
PWD	创建身份后返回的密钥，参见 创建物接入身份
publisher	用来标识设备的ID，用户可自己定义，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。client id只支持英文大小写字母，数字0-9，中划线和下划线，不支持其它字符。

```
#define PRIVATE_FILE "./root_cert.pem"//认证文件
#define USER "smart-sensor/ldw"
#define PWD "O7hrHKUYJLqxwajP/5/2fqdZ8KIDZ/aR4/CWrmRt6Gg="

int main(int argc, char* argv[]) {
    // argv[1] 为host:port
    // argv[2] 为topic
    // argv[3] 为message
    MQTTClient client;

    if (argc != 4) {
        PRINT("Usage: publish [host:port] [topic] [payload]\n");
        PRINT("Cleansession 1, Qos 1\n");
        return -1;
    }
    // 检测认证文件的可用性
    if (0 == access(PRIVATE_FILE, W_OK)) {
        // 设置认证文件
        setCertification(argv[1], PRIVATE_FILE);
    } else {
        PRINT ("Certification not exist, use normal connection.");
    }

    client = createPublisher(argv[1], "publisher", USER, PWD);
    if (NULL == client) {
        return -1;
    }
    PRINT("Start to publish message[%s] by topic : %s\n", argv[3], argv[2]);
    publishMsg(&client, argv[2], argv[3]);
    PRINT("=====\n");

    // 关闭链接
    closeConnection(&client);

    return 0;
}
```

2. 打开src/SubscriberSync.c，配置以下参数：

参数名称	解释
PRIVATE_FILE	输入认证文件所在目录
USER	创建物接入设备后返回的用户名，参见 创建物接入设备 中的步骤3
PWD	创建身份后返回的密钥，参见 创建物接入身份
clientId	用来标识设备的ID，用户可自己定义，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。client id只支持英文大小写字母，数字0-9，中划线和下划线，不支持其它字符。

```

#define PRIVATE_FILE "root_cert.pem" //
#define USER "smart-sensor/ldw"
#define PWD "O7hrHKUYJLqxwajP/5/2fqdZ8KIDZ/aR4/CWrmRt6Gg="

int main(int argc, char* argv[]) {
    char buffer[1024]; // 1KB space
    int messageLen = 0;
    int ret;
    char *host = argv[1];
    char *topic = argv[2];
    char *clientId = "subscriber";
    MQTTClient client;
    if (argc != 3) {
        PRINT("Usage: Subscriber [host:port] [topic]\n");
        PRINT("Clean session 1, Qos 1\n");
        return -1;
    }
    // 设置认证文件
    setCertification(host, PRIVATE_FILE);
    PRINT("Start connect and subscribe on topic : %s\n", topic);
    client = createSubscriber(host, clientId, topic, USER, PWD);
    if (NULL == client) {
        return -1;
    }
    // sub消息
    while (TRUE) {
        ret = startSubscribe(&client, topic, buffer, &messageLen);
        if (0 == ret && messageLen > 0) {
            PRINT("Topic: %s ,receive Message: %s\n", topic, buffer);
        }
    }
}

```

3. 打开Makefile文件:

编辑LIB_PATH和INCLUDE_PATH, 路径为MQTT -c文件的当前存储位置。

```
LIB_PATH = /home/iot/MQTT-c/lib
```

```
INCLUDE_PATH = /home/iot/MQTT-c/include
```

编辑后执行`{\color{emcolor}\textbf{make all}}`编译文件, 生成“PublisherSync”和“SubscriberSync”文件。

4. 运行“SubscriberSync”文件, 参照命令格式Subscriber [host:port] [topic], 执行订阅操作:

```
./SubscriberSync ssl://yourendpointname.mqtt.iot.gz.baiduce.com:1884 topic
```

5. 运行“PublisherSync”文件, 参照命令格式publish [host:port] [topic] [payload], 执行发布操作:

```
./PublisherSync ssl://yourendpointname.mqtt.iot.gz.baiduce.com:1884 topic publishmessage
```

返回message, 说明发布成功。

说明：
如果提示“libpaho-mqtt3cs.so.1”无法找到，请执行`export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/iot/MQTT-c/lib`。

12.3 C#代码示例

前提条件：

- 下载并安装virtual studio 2012。
- 下载[mqtt-net压缩包](#)，解压mqtt-net，即可使用vs2012运行。如果您需要自建工程，按照以下步骤操作：

1. 打开vs 2012，新建“项目>控制台应用程序”，选择.NET Framework 4.5版本。
2. 菜单栏选择“工具>库程序包管理器>程序包管理控制台”，执行[Install-Package M2Mqtt](#)命令，自动下载并安装mqtt文件。

说明：
如果无法下载mqtt文件，请打开vs2012“工具>选项>包管理器>程序包源”，添加一个本地目录作为程序包源，下载[m2mqtt.4.3.0.nupkg](#)文件，复制到刚才添加的本地目录中。当前项目右键选择“管理NuGet程序包”，找到您添加的程序包源，安装m2mqtt程序包即可。

3. 添加表头，

```
using uPLibrary.Networking.M2Mqtt;  
  
using uPLibrary.Networking.M2Mqtt.Messages;
```

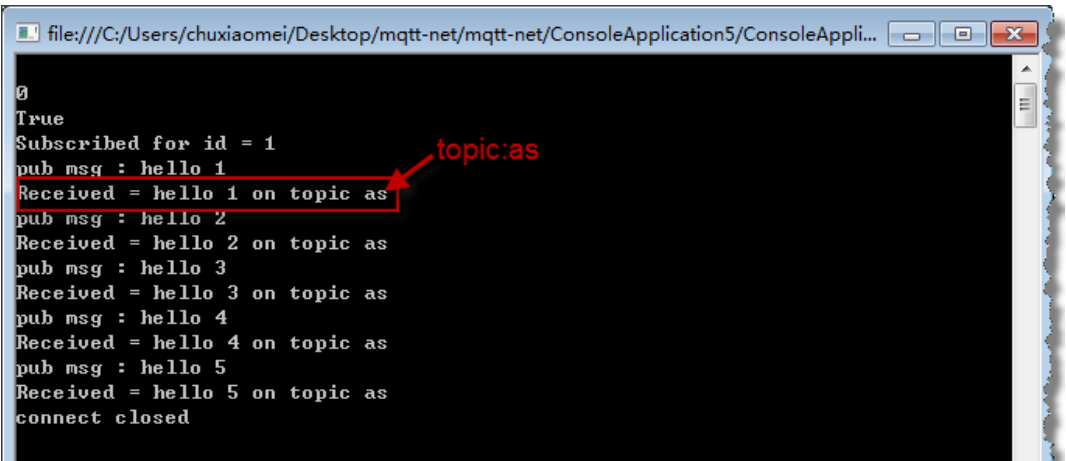
4. 参考sample code的main函数，将创建好的实例参数输入：

参数名称	解释
user	创建物接入设备后返回的用户名，参见 创建物接入设备 中的步骤3
pwd	创建身份后返回的密钥，参见 创建物接入身份
endpoint	实例地址，参见 创建物接入实例

参数名称	解释
port	实例的端口号。端口1883，不支持传输数据加密；端口1884，支持SSL/TLS加密传输。
topic	订阅的主题内容，参见 创建物接入策略
clientid	用来标识设备的ID，用户可自己定义，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。client id只支持英文大小写字母，数字0-9，中划线和下划线，不支持其它字符。

```
string endpoint = "yourendpointname.mqtt.iot.gz.baidubce.com";
int port = 1884; // 端口默认1884
string user = "username"; //创建thing, 返回username
string pwd = "password"; //创建principal, 返回password
string clientid = Guid.NewGuid().ToString(); // 获取一个独一无二的id
string[] topic = new string[] { "yourtopic" }; //输出订阅发布的主题
```

- 5. publish和subscribe代码参考mqtt-net代码示例。
- 6. 运行程序，成功订阅和发布消息，如下图所示：



12.4 python代码示例

12.4.1 下载TLS认证文件

物接入支持SSL/TLS加密传输方式，保障用户的数据传输安全。用户在执行示例代码前，需先下载TLS认证文件，并在代码中指定认证文件的存放路径。

下载[TLS认证文件](#)，并将认证文件保存至示例代码路径下。

12.4.2 下载并执行示例代码

说明：

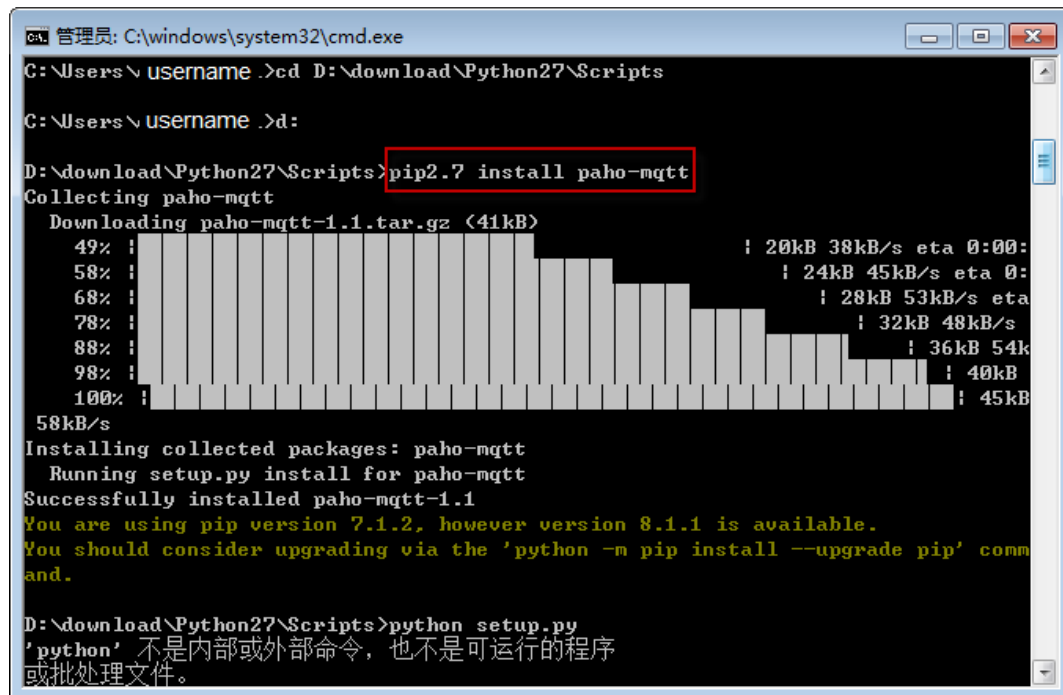
安装Python 2.7以上版本，暂时不支持Python 3.5版本。

下载[mqtt-py压缩包](#)，解压mqtt-py，目录结构如下：

```
mqtt-py
├── sub_py2.py           //订阅代码示例
├── pub_py2.py          //pub发布代码示例
└── paho-mqtt-1.1.tar.gz //mqtt client端
```

以下操作步骤以Windows为例：

1. 安装Paho MQTT Python Client。打开cmd命令行，输入命令[pip install paho-mqtt](#)，自动下载并安装Python Client，如下图所示：



用户也可以通过github下载Paho MQTT代码进行安装，具体操作如下：

```
git clone https://github.com/eclipse/paho.mqtt.python.git
cd org.eclipse.paho.mqtt.python.git
python setup.py install
```

关于Paho Python Client的详细介绍, 可查看[Paho官方网站](http://www.eclipse.org/paho/clients/python/)。

2. 安装完成后, 即可开始订阅消息。打开“sub_py2.py”文件, 填写配置参数。

参数名称	解释
trust	输入认证文件所在目录
user	创建物接入设备后返回的用户名, 参见 创建物接入设备 中的步骤3
pwd	创建身份后返回的密钥, 参见 创建物接入身份
endpoint	实例地址, 参见 创建物接入实例
port	实例的端口号。端口1883, 不支持传输数据加密; 端口1884, 支持SSL/TLS加密传输。
topic	订阅的主题内容, 参见 创建物接入策略
client_id	用来标识设备的ID, 用户可自己定义, 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。client id只支持英文大小写字母, 数字0-9, 中划线和下划线, 不支持其它字符。

代码示例如下:

```
import paho.mqtt.client as mqtt

trust = "C:\\\\Users\\username\\Desktop\\iot\\mqtt-py\\root_cert.pem" #开启TLS
                                #成功创建thing后返回的username
                                #成功创建principal后返回的password
                                #实例(endpoint)地址
                                #endpoint端口
                                #订阅的主题内容
                                #连接后返回0为成功
                                #qos
                                #topic: "+msg.topic+" Message: "+str(msg.payload))

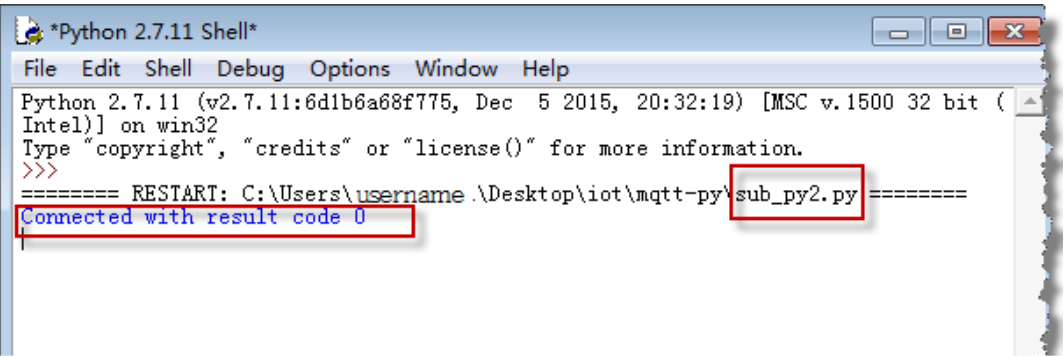
trust = "C:\\\\Users\\username\\Desktop\\iot\\mqtt-py\\root_cert.pem" #开启TLS
user = "01endpoint/01thing"
pwd = "07hrHKUYJLqxwajP/5/2fqdZ8KIDZ/aR4/CWrmRt6Gg="
endpoint = "01endpoint.mqtt.iot.gz.baidubce.com"
port = 1884
topic = "test iot service"

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe(topic, qos=1)

def on_message(client, userdata, msg):
    print("topic:"+msg.topic+" Message:"+str(msg.payload))
```

```
client = mqtt.Client(  
    client_id="test_mqtt_receiver_1", #用来标识设备的ID, 用户可自己定义, 在同一个  
实例下, 每个实体设备需要有一个唯一的ID  
    clean_session=True,  
    userdata=None,  
    protocol='MQTTv31'  
)  
  
client.tls_insecure_set(True) #检查hostname的cert认证  
client.tls_set(trust) #设置认证文件  
client.username_pw_set(user, pwd) #设置用户名, 密码  
client.on_connect = on_connect #连接后的操作  
client.on_message = on_message #接受消息的操作  
client.connect(endpoint, port, 60) #连接服务 keepalive=60  
client.loop_forever()
```

配置相关参数后，执行上例代码，返回0说明已连接IoT Hub服务，并成功订阅主题。



3. 开始发布消息，打开“pub_py2.py”文件，填写配置参数。

参数名称	解释
trust	输入认证文件所在目录
user	创建物接入设备后返回的用户名，参见 创建物接入设备 中的步骤3
pwd	创建身份后返回的密钥，参见 创建物接入身份
endpoint	实例地址，参见 创建物接入实例
port	实例的端口号。端口1883，不支持传输数据加密；端口1884，支持SSL/TLS加密传输。
topic	订阅的主题内容，参见 创建物接入策略

参数名称	解释
client_id	用来标识设备的ID，用户可自己定义，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。client id只支持英文大小写字母，数字0-9，中划线和下划线，不支持其它字符。

代码示例如下：

```
import time
import paho.mqtt.client as mqtt
import datetime

def on_publish(msg, rc):    #成功发布消息的操作
    if rc == 0:
        print("publish success, msg = " + msg)

def on_connect(client, userdata, flags, rc):    #连接后的操作 0为成功
    print("Connection returned " + str(rc))

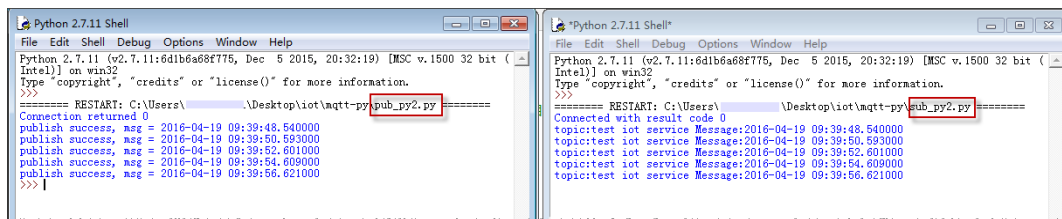
client = mqtt.Client(
    client_id="test_mqtt_sender_1", #用来标识设备的ID，用户可自己定义，在同一个实例下，每个实体设备需要有一个唯一的ID
    clean_session=True,
    userdata=None,
    protocol='MQTTv311'
)

trust = "C:\\\\Users\\username\\Desktop\\\\iot\\mqtt-py\\root_cert.pem" #开启TLS时的认证文件目录
user = "01endpoint/01thing"
pwd = "07hrHKUYJLqxwajP/5/2fqdZ8KIDZ/aR4/CWrmRt6Gg="
endpoint = "01endpoint.mqtt.iot.gz.baidubce.com"
port = 1884
topic = "test iot service"

client.tls_insecure_set(True) #检查hostname的cert认证
client.tls_set(trust) #设置认证文件
client.username_pw_set(user, pwd) #设置用户名，密码
client.connect(endpoint, port, 60) #连接服务 keepalive=60
client.on_connect = on_connect #连接后的操作
client.loop_start()
time.sleep(2)
```

```
count = 0
while count < 5: #发布五条消息
    count = count + 1
    msg = str(datetime.datetime.now())
    rc, mid = client.publish(topic, payload=msg, qos=1) #qos
    on_publish(msg, rc)
    time.sleep(2)
```

执行上例代码，返回0说明已连接IoT Hub服务，pub文件成功发布主题信息，sub文件成功接收topic，如下图所示：



12.5 Java代码示例

Java代码示例请参看[使用MQTT Client SDK模拟实体设备](#)。

第13章

百度天工 - 更懂行业的物联网平台

13.1 目录

- [天工简介](#)
- [天工架构](#)
- [应用天工](#)
 - [风电机组运维](#)
 - [智慧路灯](#)
 - [工业协议解析](#)

13.2 天工简介

物联网是指通过各种信息传感设备，实时采集任何物体或过程的各种信息，与互联网结合形成的一个巨大网络，实现物与物、物与人，所有的物品与网络的连接。物联网用途广泛，遍及智能交通、环境保护、公共安全、平安家居、智能消防、工业监测、环境监测、楼宇照明管控、水系监测、食品溯源等多个领域。但由于行业纷繁复杂，物联网从实施到应用都面临巨大的难度，主要体现在以下几个方面：

- 行业鸿沟：物联网被广泛使用在各个传统行业和新型行业中，行业之间的技术和产品天然就存在着鸿沟，而传统行业和互联网之间更是有着完全不同的技术栈和语言。
- 数据孤岛：多样的设备和协议，各式各样应用场景，随之造就了一个个数据孤岛，让我们难以看到企业运行的全貌，行业的趋势。片面的数据更是难以推动行业创新的步伐。
- 数据丢失：随着海量数据的价值被发掘，被越来越多运用到各类物联网场景，比如预测天气或是预测设备的保养周期，传统的数据计算和存储手段已经无法满足需求。目前绝大多数物联网数据未被使用。举例来说，在一个装有3万个传感器的油井设备上，只有1%的数据被使用，其他数据都被丢失，而且这1%的数据仅被用于发现和控制异

常状况，大量的数据尚未被挖掘出来用于优化和预测，只有后者才能提供最大的价值。

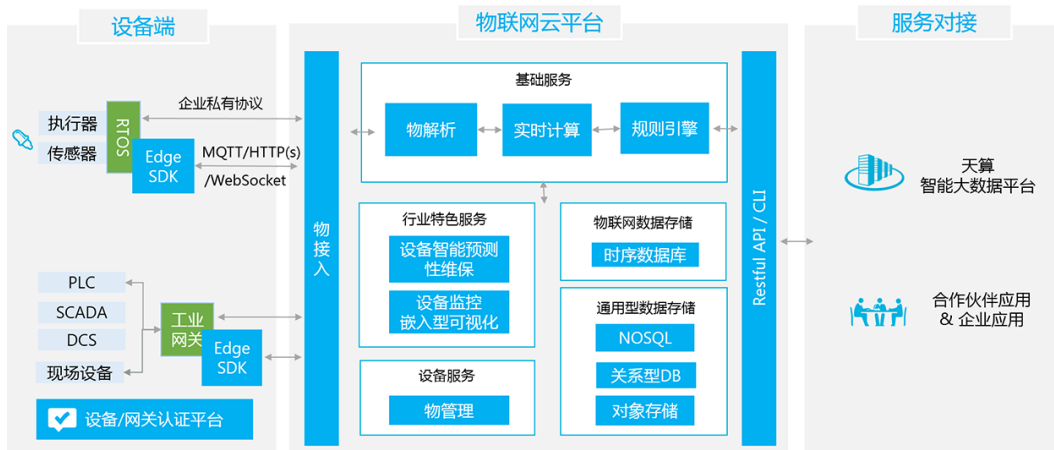
天工是基于百度云构建的、融合百度大数据和人工智能技术的“一站式、全托管”智能物联网平台，提供物接入、物解析、物管理、规则引擎、时序数据库、机器学习、MapReduce等一系列物联网核心产品和服务，帮助开发者快速实现从设备端到服务端的无缝连接，高效构建各种物联网应用（如数据采集、设备监控、预测性维保等）。

通过持续技术创新和不断积累行业经验，天工平台日益成为更懂行业的智能物联网平台，在工业制造、能源、零售O2O、车联网、物流等行业提供完整的解决方案。同时，基于天工平台设备认证服务，建立互信、共赢的生态合作机制，帮助行业用户快速实现万物互联的商业价值。

天工，取自世界上第一部关于农业和手工业生产的综合性著作《天工开物》，强调人类要和自然相协调。百度云天工平台，以使能为目标，将和各行各业一起，共同开创物联网云时代。

13.3 天工架构

目前，天工平台的服务主要由物接入、物解析、物管理、规则引擎和时序数据库组成，并可无缝对接百度云天算智能大数据平台及基础平台产品，可提供千万级设备接入的能力，百万数据点每秒的读写性能，超高的压缩率，端到端的安全防护。其基本架构如下图所示：



- **Edge SDK:** 百度云面向设备端提供的SDK，可以安装在单机设备或企业网关上。安装了SDK的设备只需要配置一个云端生成的密钥便可以完成与云端连接，实现与云端通讯配置。Edge SDK支持SSL方式连接，保证用户数据安全。
- **物接入:** 物接入是全托管的云服务，可以在智能设备与云端之间建立安全的双向连接，并通过主流的物联网协议（如MQTT）通讯，实现从设备端到云端以及从云端到设备端的安全稳定的消息传输。
- **物管理:** 主要用于对接入云端的设备进行管理和操作。物管理需要与百度云的物接入服务配合使用，对接入云端的设备进行一站式设备管理，可应用于设备的层级管理、监测、遥控、固件升级和维护保养等各个场景。

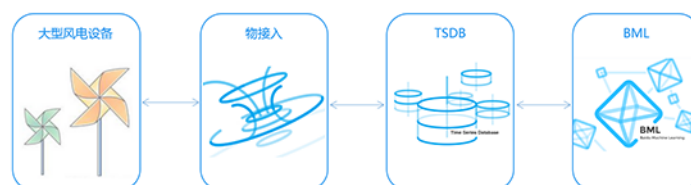
- 物解析：在云端为用户提供工业协议解析服务（比如Modbus和OPC UA）。当云端收到设备端返回的原始数据后，结合用户提供的设备通讯地址表，物解析服务可将数据解析成直接可用于存储和分析的数据。
 - 规则引擎：作为百度云天工智能物联网平台的重要组件，用于将信息根据预先设置好的规则转发至白云的其它服务。用户可通过规则引擎设定消息处理规则，对规则匹配的消息采取相应的转发操作，如推送给手机APP等；也可以将设备消息无缝转发到时序数据库、百度Kafka和对象存储中进行存储。
 - 时序数据库：用于管理时间序列数据的专业化数据库。区别于传统的关系型数据库，时序数据库针对时间序列数据的存储、查询和展现进行了专门的优化，从而获得极高的数据压缩能力、极优的查询性能，特别适用于物联网应用场景。
 - 天算智能大数据平台：提供了完备的大数据托管服务、智能API、众多业务场景模板以及人脸识别、文字识别、语音识别等服务，帮助用户实现智能业务。
- 百度的大数据计算系统已成功应用于百度搜索、广告、百度地图、百度糯米、百度外卖等几乎全部的核心业务，经过多年的业务应用锤炼和技术演进，百度大数据计算系统的集群规模、计算能力均为国内第一。天工平台可与天算智能大数据平台实现无缝对接，助力企业快速具备海量数据分析能力。
- 合作伙伴应用&企业应用：提供了API接口，可与企业应用或其他第三方平台进行对接。

13.4 应用天工

13.4.1 风电机组运维

风电机组投入使用后将不间断的向电网输送电量，每台大型发电机组每天产生的经济效益可能超过万元。如何更好的维护设备，减少设备的不可用时间，已成为每一个风力发电企业关注的首要问题。在传统情况下，设备的维护主要通过定期巡检、保养等方法。该方法可能带来大量人力、物力成本的浪费，同时也无法保证巡检后的设备能否继续正常运行至下一次巡检。

百度云天工智能物联网平台可帮助企业实现精确的故障预测，如下图所示：



根据风电行业的实际经验，对影响风电设备使用寿命的主要数据进行长期采集，风电机组产生的数据是典型的时间序列，更适合存储至TSDB数据库。从历史监控数据中找出故障发生前的数据，通过百度机器学习（BML）进行模型训练，得到设备发生故障前的特征和趋势，并应用该模型对设备故障进行预测。

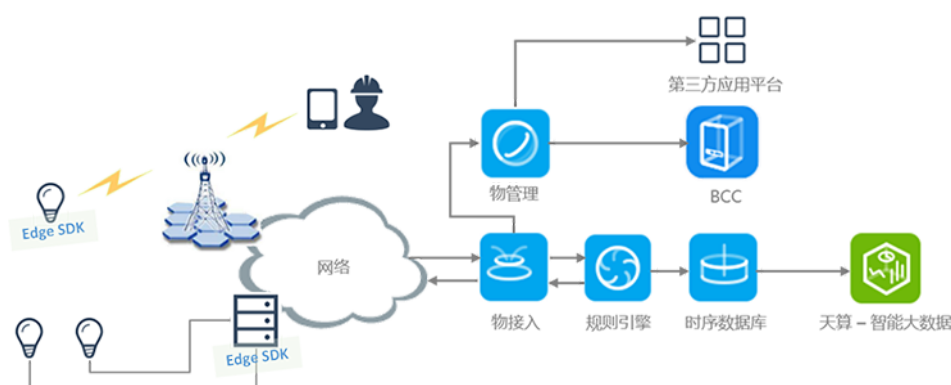
通过应用百度云“天工平台+人工智能”解决方案，可提前12小时预测风电机组设备故障，故障预测召回率可达99%，有效的减少了风电机组故障带来的经济损失以及定期巡检造成的资源浪费。

13.4.2 智慧路灯

城市路灯管理属于一个典型的物联网场景，该场景可以复制到很多其它的领域。

传统情况下，路灯管理基本上靠人力，例如：巡查、维护，甚至是靠人工控制路灯开关；而城市路灯数量巨大，分布复杂，对于管理部门来说可能已经投入了大量的人力成本，但仍然会出现故障发现和抢修不及时的情况。同时在传统情况下，由于缺少路灯的运行环境（例如：电压、温度），使用寿命等数据，很难有针对性改进使用方法，进而延长使用寿命，也很难指定准确的维护策略和备件策略。

如下图所示为百度云天工智能物联网平台智慧路灯解决方案架构。



路灯接入可以采用多种方式，例如：3G/4G，电力载波，LPWAN等，无论采用哪种方式，用户只需在接入设备中预装Edge SDK，都能轻松打通路灯和百度云之间的双向安全通道，实现将路灯接入百度云的物接入服务。路灯可以实时将设备状态、电压电流、环境温度、地理坐标等信息发送至物接入服务。同时在工程师的智能终端上预装APP对接百度云，可实时上报工程师的地理位置坐标。

物接入接收到路灯上送的消息后，可将消息分别转发至物管理和规则引擎服务。用户可在规则引擎上制定策略，实现以下操作：

- 当路灯下线或电压电流超过阈值时，检索距离现场最近的维护工程师，并自动发送告警、路灯坐标等信息给指定的维护工程师。工程师的智能终端可对接百度地图服务，自动在地图上显示待维修路灯的位置。
- 将电压电流、环境温度等信息转发至时序数据库，并对接天算大数据平台，用于后续的数据挖掘（可参考[风电机组运维案例](#)）。

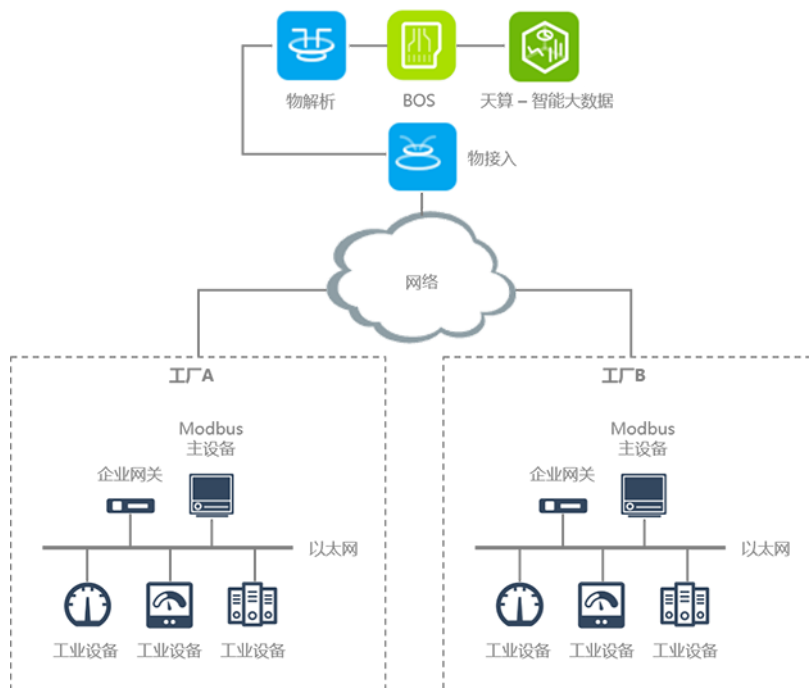
用户可在物管理服务中对路灯进行分层管理，可以将每个路灯划分到不同的“市/区/街道”下，后续可在不同维度下对路灯进行操作，例如：统一关闭/打开某一条街道下的路灯。同时物管理服务提供了API接口，可方便用户自研或对接第三方管理平台。

13.4.3 工业协议解析

Modbus协议是应用于电子控制器上的一种通用语言，它已经成为通用工业标准。通过该协议，不同厂商生产的控制设备可以连成工业网络，进行集中监控。通过对Modbus协议报文进行解析，用户可以获取到工业生产过程中的各种数据信息。

传统情况下，对协议报文的解析是在用户本地完成的。为了完成协议解析，用户需要在生产环境中额外部署解析设备。如果用户的生产环境分布在不同的地理位置，不同工厂之间会形成一个个信息孤岛；由于没有统一的数据存储中心，可能得到片面的分析结果，没法做到真正的协同生产。同样，由于分布在不同位置，用户无法统一设置和刷新信息采集策略，导致采集效率和灵活性大大折扣。

物解析在云端为用户提供工业协议（比如Modbus和OPC UA）解析服务。用户只需将原始数据在未经协议解析的情况下发送转发到云端，然后使用云端服务将原始数据进行解析，如下图所示。



在企业网关集成IoT Edge SDK后，只需要配置一个云端生成的密钥便可以完成与云端连接，将工业设备接入云端。用户可在云端统一配置设备数据的采集频率和规则，这样可以极大的降低部署时间和成本，加快工厂设备数据采集部署速度，提高协议解析的效率和灵活性。

解析后的数据可直接存储到BOS中，随着数据量的增长，云端支持的海量存储使企业无须再担心本地存储不够。同时，海量解析后的数据直接对接天算大数据平台，深入进行数据挖掘，实现预测维保、备件管理等功能。