

物联网开发平台 入门指引





【版权声明】

©2013-2025 腾讯云版权所有

本文档(含所有文字、数据、图片等内容)完整的著作权归腾讯云计算(北京)有限责任公司单独所有,未经腾讯云事先明确书 面许可,任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵 犯,腾讯云将依法采取措施追究法律责任。

【商标声明】

🔗 腾讯云

及其它腾讯云服务相关的商标均为腾讯云计算(北京)有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标,依法 由权利人所有。未经腾讯云及有关权利人书面许可,任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等 行为,否则将构成对腾讯云及有关权利人商标权的侵犯,腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况,部分产品、服务的内容可能不时有所调整。 您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定,除非双方另有约定,否则,腾讯云对本 文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务,及相应的技术售后服务,任何问题请联系 4009100100或95716。



文档目录

入门指引

公共实例快速入门

企业实例快速入门

企业实例入门概述

创建企业实例及产品设备

设备接入及上报数据

平台下发控制指令

平台转发消息至用户业务系统

快速体验平台

MQTT.fx 快速接入物联网开发平台

使用MQTT物模型接入平台

使用自定义透传上下行消息

云端控制设备入门

平台转发消息至用户 HTTP 服务

入门指引 公共实例快速入门

最近更新时间: 2025-05-28 10:22:32

操作场景

在物联网开发平台创建智能灯产品和设备,并连接物联网开发平台,通过腾讯连连小程序绑定设备,进行远程控制灯的亮度、颜 色、开关,并实时获取智能灯上报的数据。

前提条件

为了通过下面的步骤快速理解该业务场景,需要做好以下准备工作:

- 拥有一台物理或虚拟的 Linux 环境,可以编译、运行 light_demo 程序。
- light_demo 在 Linux 环境下测试和验证,主要基于 Ubuntu 16.04 版本,gcc-5.4(建议至少 gcc-4.7+)。

△ 注意:

物联网开发平台从2024年6月20日起,新注册物联网开发平台的用户需购买公共实例激活码才可使用公共实例,在此时间之前注册的用户并已开通公共实例的用户则不受影响,依然享有免费额度。若您需要使用公共实例,您可直接在 线上 购买,单击"公共实例",根据您的量产设备数量购买激活码数量。若用户有商业化的设备接入与量产需求,也可以通 过接口的商务联系我们。

操作步骤

控制台操作

新建产品

- 1. 登录物联网开发平台,选择公共实例。
- 2. 选择产品开发菜单,单击新建产品进入产品开发页面。
- 3. 在新建产品页面,填写产品基本信息。
 - 产品名称: 输入"智能灯"或其他产品名称。
 - 产品品类:选择智慧城市 > 公共事业 > 路灯照明。
 - 设备类型:选择**设备**。
 - 通信方式:选择Wi-Fi。



⊷品品类 *	标已定义标准	物模型	免 包含免开发面核	扳		精輸入品类
	智慧农业	•	▲ 公共事业	•	○ 路灯照明	【标 [无] ▲
	智慧生活	×	环境监测	•	○水表	无 无
	智能城市	Þ	消防安全	•	○ 电表	无无
	智能制造	•			○ 燃气表	无 无
	全屋智能	×			○ 两轮车	标无
	其他行业	×				
				_		_
	已选择品类: 智能	能城市/公	▼ 公共事业 / 路灯照明			•
写产品信息	已选择品类: 智能	能城市/公	*			•
笔写产品信息 "品名称 *	已选择品类:智能 智能灯	能城市/公	*			·
笔写产品信息 "品名称 *	已选择品类:智慧 智能灯 支持中文、英文、	能城市 / 公	* \$共事业 / 路灯照明 下划线、空格 (非首)	尾字符)、「	中英文括号、-、@、、/的	组合,最多不超过40个字符
55 产品信息 品名称 * 备类型	已选择品类:智慧 智能灯 支持中文、英文、 设备	能城市 / 2 . 数字、1 网关	★ 大事业 / 路灯照明 下划线、空格(非首) 子设备	尾字符)、「	中英文括号、-、@、乀 /的	组合,最多不超过40个字符
每一一日日日 一日名称 * 一日名称 * 一日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日	 已选择品类:智能 智能灯 支持中文、英文、 设备 Wi-Fi 	能城市 / 2 . 数字、] 网关	 、共事业 / 路灯照明 下划线、空格(非首) 子设备 	尾字符)、「	中英文括号、-、@、、 /的	u合,最多不超过40个字符
15 产品信息 "品名称 *	 B选择品类:智能 智能灯 支持中文、英文、 设备 Wi-Fi 请根据业务场景」 	能城市 / 2 . 数字、 网关 正确选择所	★ <	尾字符)、「	中英文括号、-、 @、 、 /的; 東产品开发	· 组合,最多不超过40个字符
百产品信息 品名称 *	 已选择品类:智慧 智能灯 支持中文、英文、 设备 Wi-Fi 请根据业务场景」 物模型 	能城市 / 2 、数字、7 网关 自定义)	 、共事业 / 路灯照明 下划线、空格 (非首) 子设备 子设备 	尾字符)、「	中英文括号、-、@、、/的 读产品开发	▲
写产品信息 品名称*	 已选择品类:智慧 智能以丁 支持中文、英文、 设备 Wi-Fi 请根据业务场景」 物模型 洗填 	能城市 / 2 、数字、7 网关 目定义;	 、共事业 / 路灯照明 下划线、空格(非首) 子设备 子设备 <	尾字符) 、1	中英文括号、-、 @、 \ /的 读产品开发	组合,最多不超过40个字符
写产品信息 品名称*	 已选择品类:智慧 智能以丁 支持中文、英文、 设备 Wi-Fi 请根据业务场景ゴ 物模型 洗填 	能城市 / 2 、数字、	★共事业 / 路灯照明 下划线、空格(非首) 子设备 ▼ ■ <td>尾字符) 、1</td> <td>中英文括号、-、 @、 \ /的 读产品开发</td> <td>组合,最多不超过40个字符</td>	尾字符) 、1	中英文括号、-、 @、 \ /的 读产品开发	组合,最多不超过40个字符

4. 产品信息填写完成后,单击**新建产品**,即可完成新建产品。

5. 产品新建成功后,您可在产品列表页查看到"智能灯"。

定义产品物模型

选择"智能灯"类型后,系统会自动生成标准功能。



1	物模	型定义	(2) 设备开发 〉 (3) 交互	形发 〉 (4) 谈	昏调试 〉	5 批量投产			
导	入物模型	型 查看物模型JS	ON						物模型症
	标准〕	功能 (7) ⑦	自定义功能 (0) ⑦ 高级功能 (0)						
	添加杨	於住功能						请输	入功能名称或标识符搜索
		功能类型 👅	功能名称	标识符	数据类型 ▼	读写类型 ▼	数据定义		操作
		属性	电灯开关 必选	power_switch	布尔型	读写	0 - 关 1 -开		编辑 删除
		属性	颜色可选	color	枚举整型	读写	0 - Red 1 - Green 2 - Blue		编辑删除
		属性	亮度 可选	brightness	整数型	读写	数值范围: 0-100 初始值: 1 步长: 1 单位: %		编辑 删除
		属性	灯位置名称可选	name	字符串	读写	字符串长度: 0-64个字符		编辑删除
	•	事件	DeviceStatus 可选	status_report	信息	-	-		编辑删除
	•	事件	LowVoltage 可选	low_voltage	告警	-	-		编辑 删除
	•	事件	Hardware_fault 可选	hardware_fault	故障	-	-		编辑删除
	共 7	훘						10 ▼ 条/页 ⊮	◀ 1 /1页 ▶

创建设备

在设备调试页面中,单击新建设备,设备名称是:dev001。

新建设备	×	•
所属产品	智能灯	
设备名称★		
	支持英文、数字、下划线的组合,最多不超过48个字符	
	保存取消	

下载 Demo 程序

下载 lightdemo 例程

首先从 GitHub 下载,或执行下面的 git 命令。

git clone https://github.com/tencentyun/qcloud-iot-explorer-sdk-embedded-c.git



修改 Demo 程序

上述 git 命令执行成功后,会生成一个 qcloud-iot-sdk-embedded-c 目录。

- 1. 进入 qcloud-iot-explorer-sdk-embedded-c 目录。
- 2. 修改该目录下的 device_info.json 文件。

> vi device_info.json
{ "auth_mode":"KEY",
<pre>"productId" '", "productSecret":"YOUR_PRODUCT_SECRET", "deviceName":"",</pre>
<pre>"key_deviceinfo":{ "deviceSecret" },</pre>
<pre>"cert_deviceinfo": "devCertFile":"YOUR_DEVICE_CERT_FILE_NAME", "devPrivateKeyFile":"YOUR_DEVICE_PRIVATE_KEY_FILE_NAME" }</pre>

- 8. 将上图红色线框中的数据分别替换为控制台产品在设备调试阶段的参数信息,单击选择设备名称进入"设备详情页"中,查看参数并保存。
 - 产品 ID: 将控制台的产品 ID,复制到上图 productId。
 - 设备名称: 将控制台的设备名称,复制到上图 deviceName。
 - 设备密钥:将控制台的设备密钥,复制到上图 deviceSecret。

编译

- 1. 上述配置信息修改完成后,即可编译。
- 2. 在 qcloud-iot-sdk-embedded-c 目录下执行以下命令进行编译。

./cmake_build.sh

3. 编译成功后,会在 output/release/bin 目录下生成 light_data_template_sample 执行文件。

运行 Demo 程序

- 1. 进入 output/release/bin 目录。
- 2. 输入 ./light_data_template_sample 。
- 3. 运行成功后,系统输出示例如下:



Product_ID: BKDDAHRGRX, Device_Name: dev001 DBG|2019-05-07 21:51:33|HAL_TLS_mbedtls.c|HAL_TLS_Connect(209): Setting up the DBG|2019-05-07 21:51:33|HAL_TLS_mbedtls.c|HAL_TLS_Connect(251): Performing the INF | 2019-05-07 21:51:33 | HAL_TLS_mbedtls.c | HAL_TLS_Connect(269): connected with with id: ZPEm9 success data successfully INF|2019-05-07 21:51:33|light_data_template_sample.c|main(496): Cloud Device 21:51:33|light_data_template_sample.c|_register_data_template_property(370): data template property=power_switch registered. data template property=color registered. 21:51:33|light_data_template_sample.c|_register_data_template_property(370): INF|2019-05-07 21:51:33|light_data_template_sample.c|main(517): Register data DBG|2019-05-07 21:51:33|shadow_client.c|IOT_Shadow_Get(384): GET Request



```
Document={"clientToken":"BKDDAHRGRX-0", "payload":{"metadata":{"reported":
{}},"timestamp":1557236942,"version":0},"result":0,"timestamp":1557237093,"type
{"power_switch":0, "color":0, "brightness":0.000000, "name": "dev001"}},
"clientToken":"BKDDAHRGRX-1"}
DBG|2019-05-07 21:51:34|shadow_client.c|IOT_Shadow_Update(318): UPDATE Request
Document: {"version":0, "state":{"reported":
DBG|2019-05-07 21:51:34|mqtt_client_publish.c|qcloud_iot_mqtt_publish(337):
publish packetID=0|topicName=$template/operation/BKDDAHRGRX/dev001|payload=
{"type":"update", "version":0, "state":{"reported":
{"power_switch":0,"color":0,"brightness":0.000000,"name":"dev001"}},
"clientToken":"BKDDAHRGRX-1"}
INF|2019-05-07 21:51:34|light_data_template_sample.c|main(607): shadow
update response, response ack: 0
"clientToken":"BKDDAHRGRX-2"}
```

4. Light Demo 程序定时会上报数据到开发平台,数据格式如下:

{"version":1, "state":{"reported":
{"power_switch":0,"color":0,"brightness":0.000000,"name":"dev001"}},
"clientToken":"BKDDAHRGRX-2"}

5. 继续保持 Light Demo 程序处于运行状态,然后前往控制台查看该设备的数据。

查看设备状态

- 1. 保持 light Demo 程序为运行状态。
- 进入控制台 > 产品开发 > 设备调试,可查看到设备"dev001"的状态为"上线"状态,表示 Demo 程序已成功连接上开 发平台。
- 3. 单击查看,可进入设备详情页。



- 4. 单击设备属性,可查询设备上报到开发平台的最新数据及历史数据。
 - 当前上报数据的最新值:会显示设备上报的最新数据。
 - 当前上报数据的更新时间:显示数据的更新时间。

设备信息 在线调试	云端诊断日志 设备	云端日志 设备本地日志 扩	展信息		
物模型日志内容日志		自动刷新			
属性 事件 行为					
属性名称/属性标识符	2				
标识符	功能名称	历史数据	数据类型	最新值	更新时间
power_switch	电灯开关	查看	布尔型	-	-
color	颜色	查看	枚举整型	-	-
brightness	亮度	查看	整数型	-	-
name	灯位置名称	查看	字符串		-

5. 单击查看,可查看某个属性的历史上报数据。

查看设备通信日志

单击**设备日志**,可查询该设备某段时间范围的所有上下行数据。

- 上行: 上行指设备端上报到开发平台的数据。
- 下行: 下行指从开发平台下发到设备的数据。

设备信息	在线调试	云端诊断日志	设备云端日志 设备本地日	3志 扩展信息				
近30分钟	近1小时	今天 昨天	近3天 2024-04-02 14:34	~ 2024-04-02 15:04		多个关键字用空格隔开		Q
时间		类别	RequestID	内容			结果 🛈	
				当前列	表为空			

在线调试

1. 当 Light Demo 成功连接到物联网开发平台后,您可在控制台设备调试列表,单击调试,进入在线调试。



设备信息在线调试	云端诊断日志 设备云端日志 设备本地	归志 扩展信息				
下发指令			调试日志		清空日志 🗸 显示响应报文	、 🔽 自动刷新
属性调试 行为调用			时间	上下行	日志类型	
- 功能名称/标识符	期望值	实时数据	▼ 2024-04-02 1	5:07:15.651 🕴 下行	设备属性控制	
✔ 电灯开关(power_switch)		π	时间 202	24-04-02 15:07:15.651		
✔ 颜色(color)	Red v	Red	日志类型 设备 method "co	昏属性控制 ntrol™		
✓ 亮度(brightness)	- 68 + %	68	clientToken "v2 params {"po	: ower_switch":1,"color":0,"brig	1105" ghtness":68}	
灯位署名称(name)	0/64 支持英文字母、数字、常见半角符号组合					
鎹						

- 2. 将亮度设置为68,颜色设置为"Red",单击发送。
- 3. 查看 Light Demo 程序,可查看到成功接收到下发的数据。

4. 通信日志会显示如下日志,表示成功下发了指令到设备端。



5. 查看通信日志,即可查看到设备成功接收到下行指令,并上报最新数据到开发平台的详细日志。

腾讯连连小程序绑定设备

- 1. 点击刚创建的"智能灯"产品,在设备列表中,找到刚创建的设备。
- 2. 单击右侧的二维码,展示用于腾讯连连小程序绑定的二维码。



✓ 物模型	> 🕑 设备开发	> 交互开发	4 设备调试		inter f	
🚺 设备调试	是供真实、虚拟设备调试功能, (更于测试设备上报、接收数据是否]]	E常, 可创建测试设备后进行调试			
新建设备	虚拟设备调试			设备名	3称 ▼ 输入设备名称搜索 (2
设备名称	状态	激活时间	最后上线时间	操作	绑定网关	
u 1	未激活	-	-	调试 二维码 删	除 查看	



3. 打开腾讯连连微信小程序,通过**扫一扫**添加此设备,添加成功后,在设备列表中将看到此设备。



16:15	© * ≈ "I ■ I
测试 ~	••• ••
暂无天气信息	11
请先设置当前家庭位置	/ /
全屋	Ξ
推荐场景	
*	_
智能灯 离线	
开启公众号提醒,第一时间接受设备消息	去开启 X
	: *
首页 场景	发现 我的

4. 点击"智能灯"设备卡片,进入智能灯操作界面,可查看设备上报的数据和控制设备。



16:19 く 智能灯	·•• •
电灯开关 关	
▶ 电灯开关	
_{亮度} - 1%	+
Red Green Blue) e
e温 - 0%	+

🔗 腾讯云

企业实例快速入门 企业实例入门概述

最近更新时间: 2024-11-27 17:34:32

企业实例通常为产业类的物联网解决方案提供设备接入、消息通信、设备管理等能力。本文将以车载设备为例帮助用户快速了解 企业实例的基本功能,如何将车载设备接入企业实例,车载设备如何上报消息,如何通过平台下发控制消息到设备,以及如何将 设备消息转发至用户的业务系统。

前提条件

- 1. 注册腾讯云 账号,并完成 实名认证。
- 2. 开通物联网开发平台。

操作步骤

用户可按如下指引快速了解企业实例。

- 1. 创建企业实例及产品设备:如何开通企业实例及如何创建产品与设备。
- 2. 设备接入及上报数据:如何通过模拟程序接入平台并上报数据。
- 3. 平台下发控制指令:如何通过平台控制设备。
- 4. 平台转发消息至用户业务系统: 用户的业务系统如何接收物联网开发平台采集的设备数据和状态。



创建企业实例及产品设备

最近更新时间: 2024-11-27 17:34:32

本文介绍如何开通企业实例,如何在企业实例下创建产品与设备,为设备接入与设备上报数据做准备。

操作步骤

购买企业实例

- 1. 进入物联网开发平台的产品介绍页或者进入物联网开发平台控制台的实例列表页都可购买企业实例。
 - 在物联网开发平台 loT Explorer 产品介绍页面,单击**立即选购**。



○ 登录 物联网开发平台控制台,单击导航栏左侧菜单实例管理进入实例列表页。单击实例列表的新增实例进入实例购买页。

期联网开发平台 #智慧生活领域的物联网应用或基于平台进行物联网	羽垂直行业应用的开发		C C C C C C C C C C C C C C C C C C C
实例概况			推荐购买
企业实例数 O 个	公共实例数 1 ↑	即将到期 0 个	已到期 1 ↑ 工业、商业、服务业等产业类场景
实例列表 什么是实例 ① 新增实例 全部实例 ▼ 所有状态 ▼			公共实例 智能家居、智能穿戴萼消费 类 场景
会 公共实例 ④ 运行中 购买激活码 实例D: inc An 创建时间: 2020-12-29 11:04:52	全业实例 (文例D: int 创建时间: 2021-08-02 11:14:22 到期时间: 2023-08-02 11:14:22	续费 升配	快捷入口
已注册设备 剩余可注册设备	已注册设备 剩余可注册设备		设备SDK 设备迁移 产品共享 日
62 ↑ 148 ↑	1 ↑		Ξ
			产品动态更多

2. 在实例购买页面,根据您的业务需要选择实例配置规格,单击**立即购买**进入商品信息确认页面。



○ 实例类型:选择"企业实例"。

腾讯云

- 实例单元规格:选择对应的规格版,根据实际业务需求选择"入门版"、"标准版"或者"高性能版"。
- 实例单元数:默认选择1,当选择其他单元数,下方的"实例规格说明"会展示对应单元数的实例规格。
- 时长: 该实例的使用时长,可先选择1个月。
- 3. 在订单确认信息页面核对订单信息,确认无误后单击去支付 > 确认支付即可完成实例购买。



确认产品信息,返回修改配置							
下单说明 请确认产品信息后提交订	单,如有优惠券可在支付时选择使用, <mark>最</mark>	冬实付金额以支付订单时为准					
产品清单							
◇ 预付费产品 (1)					应付合计 元		
产品名称	配置	类型	单价	数量时长	订单金额		
实时互动-物联版新购	实例版本:入 门版 单元数: 1	新购)元/月	x1 1个月	元		
┃ 选择优惠券 代金券					优惠券抵扣 元 王		
 2 使用代金券抵扣 0.00元兑换优惠券 第有 1 张代金券,其中 1 张与订单中产品相关,本次有 1 张可用。特权用户最多可用 10 张代金券,其中满减券最多一张。 ●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●							
				ロ 选择优惠券 文付金額	王 元 去支付		

() 说明:

如您有腾讯云相关代金券,您可勾选"使用代金券"在结算时进行抵扣。

查看企业实例

1. 购买成功后,在物联网开发平台的实例列表将查看到购买成功的企业实例。点击企业实例,如下图红色线框中会显示所购买的 企业实例信息,查看系统自动分配的实例 ID,单元数等实例规格。



例列表 什么是实例 ① 新增实例 全部实例 ▼ 所有状态 ▼		公共实例 智能家居、智能穿戴等消费 类 场景
 	激活码	 并配 快捷λ口 (快捷λ□
已注册设备 剩余可注册设备 25 ↑ 7185 ↑	E 备 ! 没备 ■ 4	设备SDK 设备迁移 产品共享 产品动态 更
· · · · · · · · · · · · · · · · · · ·		 ・控制台改版 ・新增实时音视频增值服务
	创建时间: 2024-03-21 10:41:27 到期时间: 2024-04-21 10:41:43	入门文档 ・ MQTT.fx 快速接入物联网开发平台
	同时可在线设备数 消息上下行 TPS 峰值 500 个 100 条砂	 ・ 云端控制设备入门 ・ 平台转发消息至用户 HTTP 服务

2. 单击企业实例入门版,则进入该企业实例信息概览页,在概览页用户可通过资源状态查看所购买实例的规格信息。

列信息			Ø 常用功能	帮助文档
企业实例 入口版 明于交通出行、工业、能源、服务业等	产业物联网应用,资源弹性高可靠,助力企业高	該效低成本实现物联应用方案。		
常用功能			2 隐藏常用	功能
产品开发 创建产品将设备接入平台	设备管理 创建或管理现有设备	固件升级 远程批量升级设备固件	规则引擎 设备消息转发至用户业务系统	
实例信息				
实例ID i i i i i i i i i i i i i i i i i i i	实例备注 - 🖍	过期时间 2024-04-21 10:41:43	每日赠送消息数 100万条	
实例状态 即将到期	创建时间 2024-03-21 10:41:27	实例单元数 1	产品总数 2个	
资源状态			升配 钧	卖费
同时在线设备数上限	注册设备数 🛈	消息上下行 TPS 🚯	消息转发 TPS 🚯	
500 个	2 / 5000 个	0 / 100 条/秒	0 / 150 条/秒	



创建产品

- 1. 在实例列表页单击对应企业实例,进入实例内页,单击左侧菜单的**产品开发**,进入产品列表。
- 2. 单击新建产品,显示新建产品表单界面。

^空 品开发 / 新建产		 帮助文档 🖸
● 请选择产品品	类	
产品品类 *	杨 已定义标准物模型 免 包含免开发面板 请输入品类 Q	
	智慧农业 → ▲ 其他品类 → ▲ ● 自定义产品 注 法 * ● 目定义产品 注 * ● 自定义产品 注 * ● ● 自定义产品 注 * ● ● 自定义产品 注 * ● ● 自定义产品 * ● ● ■ ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	
• 填写产品信息	已选择品类:其他行业 / 其他品类 / 自定义产品	9
产品名称 *	车载终端	2
设备类型 通信方式•	支持中文、英文、数字、下划线、空格(非首尾字符)、中英文括号、-、@、\、/的组合,最多不超过40个字符	
通信方式 *	2G/3G/4G ▼ 请根据业务场景正确选择产品的通信方式,否则会影响后续产品开发	

产品信息设置如下:

- 产品品类:选择**其他行业 > 其他品类 > 自定义品类**。
- 产品名称: 输入"车载终端"或其他名称。
- 设备类型:选择"设备"。
- 通信方式:选择"2G/3G/4G",表示设备是通过蜂窝网络接入物联网平台。
- 数据协议:选择"物模型"。

3. 单击"新建产品"按钮,系统将为用户创建产品。

物模型定义

1. 选择已经创建成功的产品,点击**车载终端**,进入产品详情页。



产品开发					Ø 功能		帮助文档 🖸
功能指引 产品是具备相同	功能的同型号设备的集合,可	可以通过以下几步完成开发				必隐	藏指引
1		2		3	4		
产品物模型定义 约定设备与平台交互的模式 查看文档	>	设备开发 按接入协议要求将设备接入到平 查看文档	台	> 设备调试 调试设备与云端的通信是否正常 查看文档	> 批量投 设备通过 查看文档	产 调试与测试,即可投产	
新建产品					按产品名称 ▼	请输入产品名称	Q
产品名称	产品ID	产品品类	设备类型	状态 ▼	创建时间	操作	
车载终端	6	用户自定义	设备	开发中	2024-02-01 16:42:11	删除	
测试	Ē.	用户自定义	设备	开发中	2024-01-23 21:29:33	删除	
开关	16	智慧生活-电工照明-一路开 关	设备	开发中	2023-07-03 17:43:15	删除	9
							C

2. 单击**导入物模型**,点击 下载 tbox.json 来下载tbox.json文件。使用文本编辑器打开 tbox.json,复制 tbox.json 文件内 容到下图的文本框中,单击**导入**。系统将成功导入物模型定义。

导入物模型	×
① 注意:导入新的JSON后原产品的物模型将会被覆盖	
您可以通过 JSON 对产品的物模型进行定义后导入平台,格式规范请查看文档	
请将要导入的物模型对应的JSON粘贴到此文本框	
导入取消	

3. 物模型导入成功后,可以查看该产品已从 JSON 文件中成功导入的物模型。



产品开发	/ 车载设备						帮助文档 亿
导入物模	型查看物模型	별JSON					物模型定义帮助 II
标准	功能 (2) ⑦	自定义功能 (15) 💿	高级功能 (0)				
添加目	自定义功能						请输入功能名称或标识符搜索Q
	功能类型 🔻	功能名称	标识符	数据类型 ▼	读写类型 🔻	数据定义	操作
	属性	速度	speed	浮点型	只读	数值范围: 0-500 初始值: 0 步长: 1 单位: -	编辑 删除
	属性	方向	direction	浮点型	只读	数值范围: 0-400 初始值: 0 步长: 1 单位: -	编辑删除
	属性	高程	altitude	浮点型	只读	数值范围: -1000-20000 初始值: 0 步长: 1 单位: 米	编辑 删除
	属性	信号强度	signal_value	浮点型	只读	数值范围: 0-500 初始值: 0 步长: 1 单位: -	編組 删除
共 15	5 条					10 ▼ 条/页	5 /2页 ▶ ▶

创建设备

1. 单击**设备调试 > 新建设备**按钮。

产品开发 / 车载设备							0.000	帮助文档 🖸
车载设备 ♪ 开发中							j	更多信息
产品 ID 设备数量	F N 🗖	产品密钥 动态注册 ③		ì	设备类型 自动创建设备 🛈	设备		
✓ 物模型定义	> 🕑 设备开发 >	3 设备调试	4 批量投产					
() 设备调试提供真	实调试功能,便于测试设备上报、接收数	据是否正常,可创建测试设备质	三进行调 试					
新建设备						设备名称 ▼	输入设备名称搜索	Q
设备名称	状态	激活时	间	最后上线时间		操作		
dŧ	离线	2024-0	02-19 10:51:25	2024-02-19 12	:06:59	调试 删除		
								9
								6

2. 在弹出的新建设备窗口中,输入设备名称 dev001,单击保存按钮。



	7
所属产品	车载设备
设备名称*	dev001
	支持英文、数字、下划线的组合,最多不超过48个字符

3. 系统成功创建设备后,自动生成其设备密钥。

设备接入及上报数据

最近更新时间: 2024-11-27 17:34:32

使用 Windows 模拟程序上报数据,通过该模拟程序模拟车载设备连接物联网平台,并上报车载设备采集的相关数据到平台。用 户可以快速体验设备如何接入及上报物模型数据至平台,以及如何在控制台查看上报的设备数据。

前提条件

1. 已开通企业实例。

腾讯云

2. 已按 创建企业实例及产品设备 指引在用户自有账号下创建完产品、导入对应的物模型以及创建完一个设备。

() 说明:

使用本文的 Windows 设备模拟程序,必须要按指引导入对应的物模型格式,用户也可自由定义物模型,该模拟程序可 根据根据 JSON 文件中的物模型定义随机上报物模型属性、事件数据至平台。

操作步骤

下载 MQTT 客户端模拟程序

- 1. 若用户系统为 Windows 系统且版本为 Win10 及以上,则可以下载平台提供的模拟程序到 PC 端。 点击下载 mqtt 客户端模拟程序。
- 2. 下载模拟程序到用户 Windows 本地磁盘并解压缩。

运行模拟程序上报数据

- 1. 进入 Windows 终端命令行,进入模拟程序解压目录。假设用户解压目录为 e:\mqtt_client。注意,解压后将看到 mqtt_connect.exe 和 tbox.json 两个文件。
- 2. 运行以下命令。

```
e:\mqtt_client>mqtt_connect.exe -s tbox.json -i 用户产品id -n 用户创建的DeviceName -k 设备密钥
-d 用户产品id.iotcloud.tencentdevices.com -l 50
```



3. 上述命令行中的用户产品id、用户创建的DeviceName、设备密钥分别从控制台上创建的设备信息页中复制粘贴。如下图所

设备数量	1		动态注册 🛈		自动创建设备(_
• 物模型定义	> 📀	设备开发	3 设备调试	〉 ④ 批量投产			
← dev001							
设备信息	在线调试	云端诊断日志	设备云端日志 谈	设备本地日志 扩展信息			
设备信息	在线调试	云端诊断日志	设备云端日志 谈	会备本地日志 扩展信息			
设备信息 设备信息 _{设备名称}	在线调试 : dev001	云端诊断日志	设备云端日志 	谷本地日志 扩展信息 FT XKN KN G	所属产品	车载设备	
设备信息 设备信息 设备名称 设备密钥	在线调试 : dev001 匠 oJpE	云端诊断日志 wPQ== [ī]	设备云端日志 ら 产品の 设备创建时间	A番本地日志 扩展信息 FT XKN FD 2024-02-19 10:50:37	所属产品 最后上线时间	车载设备 2024-02-19 12:06:59	
 设备信息 设备信息 设备名称 设备密钥 激活时间 ① 	在线调试 : dev001 匠 oJpC 2024-02-19 10:51	云端诊断日志 wPQ== 匝 :25	设备云端日志 3 产品D 设备创建时间 设备状态	谷本地日志 扩展信息 FT XKN に 2024-02-19 10:50:37 周线	所属产品 最后上线时间 固件版本	车载设备 2024-02-19 12:06:59 -	

命令行参数	描述
-s	创建产品导出的物模型 JSON 文件,模拟程序将根据该 JSON 文件定义的属性、事件随机 上报数据至平台。
-i	指定产品 ID,从控制台创建的产品 ID。
-n	指定设备名称。注意,动态注册一般是由设备端随机生成设备名称,无需在控制台创建设备。
-k	指设备密钥,从控制台所创建的设备信息页中获取设备密钥。
-d	MQTT 服务器地址,替换为自己的产品 ID,平台 MQTT 服务地址为 "产品 ID.iotcloud.tencentdevices.com"。
-1	指定循环次数,循环上报数据次数达到后模拟程序自动关闭 MQTT 连接并退出。

4. 运行成功后,模拟程序将打印如下日志,用户无需关闭窗口,模拟程序将会随机上报数据至平台。

INF INF INF DBG DBG INF DBG DBG DBG DBG	2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19	12:06:5% 12:06:5% 12:06:5% 12:06:5% 12:06:5% 12:06:5% 12:07:00 12:07:00 12:07:00 12:07:00 12:07:00 12:07:00	<pre>9] qcloud_iot_device.c iot_device_info_set(56): SDK_Ver: 3.2.1-51515dd499d674200f7e5958d39a454fb79487c4, Product_ID: F N, Device_Name: dev001 9) qcloud_iot_device.c iot_device_info_set(56): SDK_Ver: 3.2.1-51515dd499d674200f7e5958d39a454fb79487c4, Product_ID: F N, Device_Name: dev001 9) MAL_ICP_win.c HAL_TCP_Connect(105): connected with TCP server: F N totcloud.tencentdevices.com:1883 9) mqtt_client.c IOT_MUT_Construct(134): mqtt connect with id: ufHBm success 9) mqtt_client_subscribe.c qcloud_iot_mqtt_subscribe(147): topicName=%thing/down/property/I V/dev001 packet_id=9775 9) data_template_client.c _template_mqtt_event_handler(275): template subscribed successfully, packet-id=9776 9) dqtt_client_subscribe.c qcloud_iot_mqtt_subscribe(313): Sync_device_data_successfully, packet-id=9776 9) dqtt_client_subscribe.c qcloud_iot_mqtt_subscribe(313): Sync_device_data_successfully, packet-id=9776 9) dqtt_client_demo.c event_handler(275): template subscribed successfully, packet-id=9776 9) dqtt_client_cl_template_client.c _template_check_subscribe(313): Sync_device_data_successfully, packet-id=9776 9) dqtt_template_client.c _template_devent_handler(275): template_subscribed_successfully, packet-id=9776 9) dqtt_template_client.c _template_devent_handler(275): template_subscribes/successfully, packet-id=9777 9) dqta_template_client.c _template_devent_handler(275): template_subscribes/successfully 9) dqtt_client_subscribe.c qcloud_iot_mqtt_subscribes/synce_sthing/down/action/F N/dev001 packet_id=9777 9) dqta_template_client.c _template_mqtt_event_handler(275): template_subscribes/synce_sthing/down/action/F N/dev001 packet_id=9777 9</pre>
INF INF DBG INF	2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19	12:07:00 12:07:00 12:07:00 12:07:00 12:07:00) mqtt_connect_demo.c event_handler(2353): subscribe success, packet-id=9777 data_template_client.c _template_check_subscribe(313): Sync device data successfully) mqtt_connect_demo.c _usr_init(2878): cloud Device Construct Success) mqtt_connect_demo.c _usr_init(2386): add your init code here) mqtt_connect_demo.c _usr_init(2386): add your init(255): data_template_property=cid_registered.
INF DBG INF INF INF INF	2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19	12:07:0 12:07:0 12:07:0 12:07:0 12:07:0 12:07:0 12:07:0 12:07:0	l mgtt_connect_demo.cregister_data_template_property(2455): data_template_property=jac_registered.) mgtt_connect_demo.cregister_data_template_property(2455): data_template_property=speed_registered. 1 mgtt_connect_demo.cregister_data_template_property(2455): data_template_property=direction_registered. 1 mgtt_connect_demo.cregister_data_template_property(2455): data_template_property=altitude_registered. 1 mgtt_connect_demo.cregister_data_template_property(2455): data_template_property=altitude_registered. 1 mgtt_connect_demo.cregister_data_template_property(2455): data_template_property=signal_value_registered. 1 mgtt_connect_demo.cregister_data_template_property(2455): data_template_property=signal_value_registered.
INF INF INF INF INF INF	2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19	12:07:01 12:07:01 12:07:01 12:07:01 12:07:01 12:07:01 12:07:01 12:07:01	l mqtt_connect_demo.c _register_data_template_property(2455): data template property=fuel registered. Imqtt_connect_demo.c _register_data_template_property(2455): data template property=analog registered. Imqtt_connect_demo.c _register_data_template_property(2455): data template property=longitude registered. Imqtt_connect_demo.c _register_data_template_property(2455): data template property=longitude registered. Imqtt_connect_demo.c _register_data_template_property(2455): data template property=fuitude registered. Imqtt_connect_demo.c _register_data_template_property(2455): data template property=report_time_registered. Imqtt_connect_demo.c _register_data_template_property(2455): data template property=report_time_registered. Imqtt_connect_demo.c _main(2692): Register data_template_propertys_success
INF INF DBG F	2024/02/19 2024/02/19 2024/02/19 N=0"}	12:07:01 12:07:01 12:07:01	l mgtt_connect_demo.cl_register_data_template_action12326): data template_action-device_control registered. mgtt_connect_demo.clmain(2702): Register data template actions Success mgtt_client_publish.clgcloud_iot_mgtt_publish(347): publish packetID=0 topicName=\$thing/up/property/F N/dev001 payload={"method":"get_status", "clientToken":"
DBG DBG ata t_t DBG	2024/02/19 2024/02/19 ":{"reporte ime":"00000 2024/02/19	12:07:01 12:07:01 d":{"dire 000000000 12:07:01	l data_template_client.c _get_status_reply_ack_cb(211): replyAck=0 data_template_client.c _get_status_reply_ack_cb(215): Received_Json_Document=["method":"get_status_reply","clientToken":"FI (N=0", "code":0," status": "success","d ccion":0.49, "longitude":-359999951, "analog: ["id":"0000000000000049, "name": "0000049", "value":"00000000000000000000000000000000000
DBG Q INF	2024/02/19 V-1", "pa 2024/02/19	12:07:02 rams":{"c 12:07:02	<pre>lmgtt_clent_publish.c qcloud_iot_mqtt_publish(347): publish packetID=0 topicName=\$thing/up/property/F N/dev001 payload={"method":"report", "clientToken":"F id":1,"lac":1} 2 mqtt_connect_demo.c main(2758): data template reporte success</pre>
DBG "FT INF	2024/02/19 QUQP3XKN-2" 2024/02/19 ess"}	12:07:02 , ″event] 12:07:02	2 mqtt_client_publish.clqcloud_iot_mqtt_publish(340): publish topic seq=9778 topicName=\$thing/up/event/F N/dev001 payload={"method":"event_post", "clientToken": [d":"over_speed_alarm_event", "type":"alert", "timestamp":1708315622000, "params":{"position":1,"area_id":1}} 2 mqtt_connect_demo.c[OnReportRepIyCallback(2440): recv report reply response, reply ack: 0, ("method":"report_reply","clientToken":"F V-1","code":0,"status":"s
INF DBG DBG DBG DBG	2024/02/19 2024/02/19 2024/02/19 2024/02/19 2024/02/19	12:07:02 12:07:02 12:07:02 12:07:02 12:07:02	2 mqtt_connect_demo.c event_handler(2367): publish success, packet-id=9778 2 data_template_event.c _on_event_reply_callback(107): recv: {"method':"event_reply", "clientToken":"FI N-2", "code":0, "status":"", "data":{}} 2 data_template_event.c _on_event_reply_callback(126): eventToken:F N-2 code:0 2 mqtt_connect_demo.c event_post_cb(2218): Reply: {"method":"event_reply", "clientToken":"F N-2", "code":0, "status":"", "data":{}} 2 data_template_avent_ch_traverse_avent_like(14): eventToken[E N-2] relevend 2 data_template_avent_traverse_avent_like(14): eventToken[E N-2] relevend

控制台查看设备上报数据

查看物模型日志数据

- 1. 进入该模拟设备控制台页面,进入**设备云端日志 > 物模型日志 > 属性。**
- 2. 当模拟程序在运行时,将会在如下页面查看到模拟程序上报的最新属性值以及对应的更新时间。



物構型日志の家日ま						
属性事件行为	为					
属性名称/属性标识符	Q					?
示识符	功能名称	历史数据	数据类型	最新值	更新时间	
id	基站码	查看	整数型	27	2024-02-19 12:08:34.262	
с	地区区域码	查看	整数型	27	2024-02-19 12:08:34.262	
peed	速度	查看	浮点型	0.24	2024-02-19 12:08:17.438	
irection	方向	查看	浮点型	0.24	2024-02-19 12:08:17.438	
ltitude	高程	查看	浮点型	-999.76001	2024-02-19 12:08:17.438	
ignal_value	信号强度	查看	浮点型	0.24	2024-02-19 12:08:17.438	
atellite_count	GNSS定位卫星数	查看	整数型	24	2024-02-19 12:08:17.438	
iel	油量	查看	浮点型	0.24	2024-02-19 12:08:17.438	
nalog	描圳景	杏玉	结构体	{"id":"0000000000000000000000000000000000	2024 02 10 12:08:17 438	

查看内容日志

- 1. 进入该模拟设备控制台页面,进入**设备云端日志 > 内容日志 > 属性**。
- 当模拟程序在运行后并成功上报数据后,将会在如下页面查看到模拟程序上行的 JSON 报文以及平台返回至设备的下行 JSON 报文。详细的物模型协议格式可参考 物模型协议。



设备信息 在线调试 物模型日志 内容日志	云端诊断日志 上下线日志	设备云端日志 ② ● 自动刷新	扩展信息
日志类型	• t	ppic \$thing/up/property/F N/dev001,	\$thing/down/pr 🔻
近30分钟 近1小时	今天昨天	近3天 近7天 近30天 202	4-02-19 00:00 ~ 2024-02-19 23:59
时间	通讯类型	Topic	通信内容
2024-02-19 12:08:34.288	下行	\$thing/down/property/l .KN/dev001	{"method":"report_reply","clientToken":"F V-33","code":0,"status":"success"}
2024-02-19 12:08:34.252	上行	\$thing/up/property/F (N/dev001	{"method":"report", "clientToken":"F N-33", "params":{"cid":27,"lac":27}}
2024-02-19 12:08:31.058	下行	\$thing/down/property/F V/dev001	{"method":"report_reply", "clientToken":"F N-32", "code":0, "status":"success"}
2024-02-19 12:08:31.016	上行	\$thing/up/property/F N/dev001	{"method":"report", "clientToken":"F N-32", "params":{"cid":26}}
2024-02-19 12:08:17.501	下行	\$thing/down/property/F	{"method":"report_reply","clientToken":"F N-29","code":0,"status":"success"}
2024-02-19 12:08:17.426	上行	\$thing/up/property/t KN/dev001	{"method":"report", "clientToken":"F N-29", "params": ("cid":24,"lac":24,"speed":0.240000,"direction":0.240000,"altitude":-999,760010,"signal_value":0.24000 0,"satellite_count":24,"fuel":0.240000,"analog": ("id":"000000000000000000024","name":"024","value":"00000000000000000000024"},"iongitude ":-359999976,"latitude":-360000000.000000,"report_time":"00000000000000000000024"}
2024-02-19 12:08:14.246	下行	\$thing/down/property/f N/dev001	{"method":"report_reply","clientToken":"F N-28","code":0,"status":"success"}

查看设备上线/离线日志

- 1. 进入该模拟设备控制台页面,进入**设备云端日志 >上下线日志**,可查询设备上下线行为以及下线原因。
- 当模拟程序在运行后并成功上报数据后,将会在如下页面查看到设备连接到平台的事件,动作类型为"上线";若模拟程序主动关闭或者因心跳超时,则动作类型都显示为"下线",具体的原因如下图红色线框显示。



在线调试	云端诊	浙日志	设备云站	尚日志	设备本地日志	志 扩展信息			
内容日志	上下约	纪志	φ 🔾	自动刷新	Ī				
近1小时	今天	昨天	近3天	近7天	近30天	2024-02-19 00:00	~ 2024-02-19 23:59	ö	
	动作		详细信息						
2024-02-19 12:08:35.929 下线			Device disconnect,last active time:2024-02-19 12:08:34						
06:59.752	上线		Device conn	Device connect,version:4,keepalive:240,cleansession:1,clientip:1					
00:53.357	下线		Device keep	alive timeou	ut,last active time	e:2024-02-19 10:54:52]		
2024-02-19 10:51:25.995 上线			Device connect,version:4,keepalive:240,cleansession:1,clientip:1						
	 本线调试 内容日志 近1小时 35.929 35.929 35.357 31.25.995 	在线调试 云端衫 内容日志 上下約 近1小时 今天 动作 动作 8:35.929 下线 06:59.752 上线 01:25.995 上线	在线调试 云端诊断日志 内容日志 上下线日志 近1小时 今天 昨天 动作 动作 8:35.929 下线 0 96:59.752 上线 1 0:53.357 下线 1 51:25.995 上线 1	在线调试 云端诊断日志 设备云如 内容日志 上下线日志 • • • • • • • • • • • • • • • • • • •	在线调试 云端诊断日志 设备云端日志 内容日志 上下线日志 (*) ① 自动刷新 近1小时 今天 昨天 近3天 近7天 动作 详细信息 335,929 下线 Device disconnect, last at	在线调试 云端诊断日志 设备云端日志 设备本地日式 内容日志 上下线日志 (*) ● 自动刷新 近1小时 今天 昨天 近3天 近7天 近30天 动作 详细信息	在线调试 云端诊断日志 设备云端日志 设备本地日志 扩展信息 内容日志 上下线日志 (*) ● 自动刷新 近1小时 今天 昨天 近3天 近30天 2024-02-19 00:00 动作 详细信息	在线调试 云端诊断日志 设备云端日志 设备本地日志 扩展信息 内容日志 上下线日志 (*) </td <td>在线调试 云端诊断日志 设备云端日志 设备本地日志 扩展信息 内容日志 上下线日志 ・ ● 自动刷新 近1小时 今天 昨天 近3天 近7天 近30天 2024-02-19 00:00 - 2024-02-19 23:59 ご 近1小时 今天 昨天 近3天 近7天 近30天 2024-02-19 00:00 - 2024-02-19 23:59 ご 10:53.552 F线 Device disconnect, last active time:2024-02-19 12:08:34 10:53.357 F线 Device connect, version: 4, keepalive:240.cleansession: 1, clientip: 1 61:25.995 上线 Device connect, version: 4, keepalive:240.cleansession: 1, clientip: 1</td>	在线调试 云端诊断日志 设备云端日志 设备本地日志 扩展信息 内容日志 上下线日志 ・ ● 自动刷新 近1小时 今天 昨天 近3天 近7天 近30天 2024-02-19 00:00 - 2024-02-19 23:59 ご 近1小时 今天 昨天 近3天 近7天 近30天 2024-02-19 00:00 - 2024-02-19 23:59 ご 10:53.552 F线 Device disconnect, last active time:2024-02-19 12:08:34 10:53.357 F线 Device connect, version: 4, keepalive:240.cleansession: 1, clientip: 1 61:25.995 上线 Device connect, version: 4, keepalive:240.cleansession: 1, clientip: 1

控制台查看设备轨迹日志

平台提供了云端轨迹日志查询功能,当设备出现无法上报数据或无法接收控制指令或规则引擎无法转发消息时,便于用户分析哪 个环节出现问题,用于定位是否是设备侧或用户业务系统相关问题。

1. 进入该模拟设备控制台页面,进入**设备云端日志 > 云端诊断日志**,选择合适的时间范围。

2. 当模拟程序在运行后并成功上报数据后,将会在如下页面查看到模拟设备在平台的端到端行为轨迹。

3. 若某个行为出现失败,失败的原因分析可参考 异常原因。



设备信息	在线调试	云端诊	断日志	设备云站	耑日志	设备本地日志	扩展信息			
近30分钟 2024-02-19 00	近1小时 :00 ~2024-	今 天 02-19 23:59	昨天	近3天 �	近7天 自动刷新	近30天		多个关键字用空格隔开		Q
时间		类别			RequestID		内容		结果()	
2024-02-19 12	:08:35.929	STAT	US				Device disconnect, last active	e time:2024-02-19 12:08:34	SUCC	
2024-02-19 12	:08:34.286	MES	SAGE				Publish message to device: t \$thing/down/property/F	opic: V/dev001, QOS: 0, ID: 0	SUCC	
2024-02-19 12	:08:34.259	MES	SAGE				Device publish message to topic:\$thing/up/property/F	N/dev001,QOS:0,ID:1000	SUCC	
2024-02-19 12	:08:34.257	MES	SAGE				Send message to RuleEngin topic:\$thing/up/property/F	e, V/dev001	SUCC	
2024-02-19 12	:08:31.058	MES	SAGE				Publish message to device: t \$thing/down/property/F	opic: V/dev001, QOS: 0, ID: 0	SUCC	
2024-02-19 12	:08:31.022	MES	SAGE				Device publish message to topic:\$thing/up/property/F	N/dev001,QOS:0,ID:1000	SUCC	
2024-02-19 12	:08:31.021	MES	SAGE				Send message to RuleEngin topic:\$thing/up/property/F	e, N/dev001	SUCC	



平台下发控制指令

最近更新时间: 2024-11-27 17:34:32

车载设备接入到平台后,一般会通过用户的业务系统发起控制设备的操作。例如远程设置车载设备的某个参数,可以通过物联网 开发平台提供的云 API 或在线调试功能下发控制消息至设备侧。

前提条件

体验下发控制指令需要提前做好以下准备工作:

- 按企业实例快速入门创建对应产品,并导入物模型。
- 能够成功使用 Windows 模拟程序模拟设备连接平台。

使用 API Explorer 控制设备

- 1. 登录 物联网开发平台 后,访问 设备远程控制 API ,可查看平台提供的 API 服务,单击"点击调试",可进入 API Explorer 在线 API 调试工具。
- 进入设备远程控制 API 调试页面,输入必选参数。然后单击右侧区域的"发送请求"按钮。API Explorer 将返回响应结果。

注意:在发送请求前,请确保被控设备已成功连接平台。

- 输入参数 Region:选择华南地区 (广州) ap-guangzhou。
- 输入参数 ProductId: 请输入按企业实例快速入门指引,在平台生成的对应产品 ID。
- 输入参数 DeviceName: 请输入连接到平台的 DeviceName。
- 输入参数 Data: 请输入物模型 JSON,例如{"win_switch":1}



API Explorer 物	勿联网开发平台 (IOTE	XPLORER) 🔻		产品体验,您说了算 用户之声 区
搜索接口,支持中英文搜索	索 Q	ControlDeviceData iotexplorer 2019-04-23 查看API文档	「赞 😔 吐槽	在线调用 代码示例 CLI示例 签名示例 文档说明 数据模拟
固件升级相关接口	~	更多选项 ▼		问题反馈
设备管理相关接口	^			
批量禁用启用设备		输入参数		注意:通过API发送请求等同于真实操作,请小心进行 点击下面的"发送请求"按钮,系统会让POST的请求方法发送您在左侧填写的参数到对应
批量解绑子设备		Region (j)		的接口,该操作等同于真实操作,建议您仔细阅读产品计费文档了解费用详情,同时系统 会给您展示请求之后的结果 脑应头盖相关信息 供您调试 参考
发布RRPC消息		华南地区(广州) ap-guangzhou		
设备透传指令控制		参数输入方式		发送请求 请求耗时: 893ms
发布广播消息		表单 JSON 参数	如推荐	
获取指定网关设备的子设	设备列表	ProductId [*] 🛈 😔		响应结果 响应头 真实请求
获取设备绑定的用户列表	Ę	CT MARTINE TO A CONTRACT OF A	8	(r_+
获取产品的设备列表		DeviceName [*] 🥡		Kesponse : ("Data": "",
生成单个设备绑定的签名	3	dev001	8	"RequestId": "519486c9-d37c-4c3f-91e4-5261866e3e24", "Result": "(\"Sent\":1,\"pushResult\":0)"
直接绑定设备和家庭				}
查询绑定到家庭的网关设	设备的子设备列表	/"win switch":11	0	查看 519486c9-d37c-4c3f-91e4-5261866e3e24 的诊断信息 区
获取网关绑定的子设备列	问表	{ win_switch . i}		78
查询设备绑定的网关设备	z	Method (选填) [*] 🚯		8
查看设备详情		string		E P
批量删除设备		DeviceId (选填) [*] 3		E
删除设备		string		U
创建设备		DataTimestamp (选填) [*] ①		
展示英文接口 😳 吐	±槽	发起调用 调用历史 展示所有参数 ▼		

3. 若输入参数输入正确,而且远控的设备已连接物联网开发平台,即为在线状态,API Explorer显示的相应结果为如下报文。



控制台查看下行控制消息

- 1. 进入控制台产品开发,选择目标产品进入设备调试,单击模拟接入的设备名称。
- 2. 选择**设备云端日志 > 内容日志 > 属性**,筛选设备下行时间范围。



产品开发 / 车载 /		, 4035-02	i 🔾		×××× 自动创建设备 ①		← 回到旧版	帮助文档 🖸
 ✓ 物模型定义 ✓ dev001 	→ 设备开发	3 设备调试	>	(4) 批量投产				
设备信息 在约 物模型日志	式调试 云端诊断日志 内容日志 上下线日志	设备云端日志 ① 自动刷新	设备本地日元	5. 扩展信息				
日志类型属性	▼ t	opic Sthing/up/property/F	i/d 近30 天	ev001, \$thing/down/pr •	2024-02-19 12 ⁻ 08			
时间	通讯类型	Торіс		通信内容				
2024-02-19 12:08:34.2	288 下行	\$thing/down/property/F	KN/dev	v001 {"method":"report_r	reply","clientToken":"F K	N-33","code":0,"status":"suo	ccess"}	9
2024-02-19 12:08:34.2	252 上行	\$thing/up/property/FTC	/dev00	1 {"method":"report",	"clientToken":"F" N-33"	, "params":{"cid":27,"lac":27	7}}	4
2024-02-19 12:08:31.0	058 下行	\$thing/down/property/F	KN/dev	v001 {"method":"report_r	reply","clientToken":"F	N-32","code":0,"status":"suo	ccess"}	
2024-02-19 12:08:31.0	016 上行	\$thing/up/property/FTC	/dev00	1 {"method":"report",	"clientToken":"F N-32"	, "params":{"cid":26}}		E
2024-02-19 12:08:17.5	501 下行	\$thing/down/property/F	KN/dev	v001 {"method":"report_r	reply","clientToken":"F	N-29","code":0,"status":"suc	ccess"}	

3. 搜索"win_switch",设备远程控制 API 执行成功后,在控制台可以查询到对应的下行消息记录。



2024-02-19 12:08:17.501	下行	\$thing/down/property/F N/dev001	{"method":"report_reply", "clientToken":"F V-29", "code":0, "status":"success"}
2024-02-19 12:08:17.426	上行	Sthing/up/property/F KN/dev001	("method":"report", "clientToken":"F N-29", "params": ("cid":24,"lac":24,"speed":0.240000,"direction":0.240000,"altitude":-999.760010,"signal_value":0.24000 0,"satellite_count":24,"fuel":0.240000,"analog": ("di":"0000000000000000000024","name":"024","value":"00000000000000000000000024"),"longitude ":-359999976,"latitude":-360000000.000000,"report_time":"00000000000000000000000000000000000
2024-02-19 12:08:14.246	下行	\$thing/down/property/F	{"method":"report_reply", "clientToken":"F V-28", "code":0, "status":"success"}
2024-02-19 12:08:14.205	上行	Sthing/up/property/F KN/dev001	("method":"report", "clientToken":"F N-28", "params": ("cid":23,"lac":23,"speed":0.230000,"direction":0.230000,"altitude":-999.770020,"signal_value":0.23000 0,"satellite_count":23,"fuel":0.230000,"analog": {"id":"00000000000000000000023","name":"23","value":"0000000000000000000023"},"longitude":-3 59999977,"latitude":-360000000.000000}}
2024-02-19 12:08:11.624	下行	\$thing/down/property/F V/dev001	{"method":"control","clientToken":"v2530232175iDTeC::519486c9-d37c-4c3f-91e4- 5261866e3e24","params":{" win_ switch":1}}
2024-02-19 12:08:11.032	下行	\$thing/down/property/FN/dev001	{"method":"report_reply", "clientToken":"F N-27", "code":0, "status":"success"}
2024-02-19 12:08:10.992	上行	\$thing/up/property/F N/dev001	{"method":"report", "clientToken":"F N-27", "params": ("cid":22,"lac":22,"speed":0.220000,"direction":0.220000,"altitude":-999.780029,"signal_value":0.22000 0,"satellite_count":22,"fuel":0.220000,"analog": ("id":"00000000000000000000022","name":"2","value":"0000000000000000000022"},"longitude":-3599 99978}}

平台转发消息至用户业务系统

最近更新时间: 2024-11-27 17:34:32

操作场景

物联网解决方案通常需要实时获取设备上报到物联网平台的数据、状态等信息,再结合各自场景的业务数据来完成整套物联网解 决方案的闭环业务流程。本文档主要介绍如何将采集的车载设备数据使用平台的规则引擎消息转发至用户自建 HTTP 服务的能 力,将设备数据、状态转发到用户自建的 HTTP 服务,用户可根据此文档中的代码示例快速了解并构建自己的 HTTP 服务获取 设备数据。

前提条件

- 1. 已开通企业实例,模拟程序已接入平台并上报数据至平台调试通过。
- 2. 用户需提前注册开通 腾讯云 SCF 产品,通过 SCF 产品托管 HTTP 服务,将对应的 HTTP 服务接收地址准备好配置到规则引擎转发服务中。

() 说明:

用户基于腾讯云SCF产品部署好HTTP服务后,需要确保模拟程序连接平台并持续上报数据,HTTP服务才能打印接收 到的设备数据。

操作步骤

使用 SCF 部署 HTTP 服务

本文档使用腾讯云云函数(Serverless Cloud Function,SCF)产品快速搭建 Web 服务,来接收物联网平台规则引擎处理 后的数据。本示例只展示如何接收物联网平台的数据,进一步使用数据需要用户根据实际业务场景去处理。

在 SCF 平台基于 Express 快速搭建 Node 服务示例

1. 登录 SCF 控制台,选择 Serveless 应用,单击新建应用。

Serverless	Serverless 应用					
計 概览 ② 函数服务	 ・Web 建站全新体验 无改造部署, ・【联合特惠】全景录制,所见即序 	函数直接处理 HTTP 请求,体验产品写问卷,在 所得的录制模式,高度还原互动效果,免后期合品	有机会获得精美礼品! <u>产品文档>>> 🕻 问卷入口></u> 成,稳定支持高并发业务需求,更有实时音视频、	≥ 🖸 、云函数资源包,低至 1 元,立即	□領取>≥ 【	
② Serverless 应用 高级能力	新建应用			请选择您要进行过	滤的标签	Q Ø
◇ 层	应用名称 🕈	状态	标签	上次修改时间 🍣	操作	
I 函数套餐包	iot-test	⊘正常	http	2023-08-23 15:44:26	访问应用	
¹ A展館刀 【2】ASW工作流 ☑	共 1 条				10 ▼ 条/页 🛛 ◀ 1 /1 7	Į ► ►
						•
						Ċ
						1
						-
						-
三 给产品打个分 ③						



2. 选择 Web 应用下的 Express 框架模板,单击下一步。

						← 新建应用	erverless
							概览
		X	b应用	Wel	应用市场	创建方式	函数服务
		ĩЛ	模版或导入已有项目,快速部署 Web 应	通过	快速创建开箱即用的 Serverless 应用		Serverless 应用
							级能力
Q					请输入名称查询	框架选择	三、
							函数套餐包
查看详情	Express 框架 社区模版	查看详情	Nuxt.js 框架 社区模版	查看详情	Next.js 框架 社区模版		展能力
					· · · · · · · · · · · · · · · · · · ·		IASW工作流 ☑
SS	expres	г	TXUN	JS	NEXT.		PEB事件总线 ☑
迁移您的 Express 应用	基于云函数和 API 网关,快速迁移您	移您的 Nuxt.js 应用	基于云函数和 API 网关,快速迁和	您的 Next.js 应用	基于云函数和 API 网关,快速迁移航		
查看详情	Flask 框架 社区模版	查看详情	Egg 框架 社区模版	查看详情	Koa 框架 社区模版		
. al-	➡ E1aa1	~~			I		
•	Flask 框架 社区機廠	查看详情	Egg 框架 社区頻販	查看详情	Koa 框架 社区 佩版	下一步	经产品打不分 向

3. 按下图所示输入应用名,选择环境、地域,上传方式。配置结束后单击**完成**。

Serverless	← 新建应用		
₩ 節			
② 函数服务	基础配置		
珍 Serverless 应用	应用名	iot	
高级能力		最短2个字符,最长63个字符,只能包含小写字母、数字及分隔符*-"、且必须以小写字母开头,数字或小写字母结尾。	
	环境	开发环境 - dev ▼	
🛾 函数套餐包		为您的项目选择不同部署环境,实现开发、测试和生产环境的隔离。	
拓展能力	框架	Express 应用 ▼	
CI ASW工作流 II	地域		
⊗ EB事件总线 ☑			
	上传万式	● 第1時末 ● 第1時末 ●	Q
	高级配置	•	6
	自定义域名 🛈	「「倉用	œ
	-75, WA 27, 199		Ξ
	的数配直		
□ 给产品打个分 ③	取消完	RZ CONTRACTOR OF CONTRACTOR	

4. 等待部署完成后进入应用,在 API 网关中可以看到 URL。此 URL 就是您要在规则引擎中填写的 API 地址,假设此 URL 地址为: https://iot-api/ , 那么在本示例中您的 API 地址为 URL 加上接口路由,即: https://iot-api/test 。单击函数名称,进入函数服务。


	/ [1407 cm	
Serverless	- iot	Jev v	访问应用	汪钥应用
盟 概览	资源列表	开发部署 部署日志		
② 函数服务	其础信息			
② Serverless 应用	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	iat		
	实例名称	http-e: x		
高级能力	地域	. ap-guangzhou (广州)		
\$ 层	API网关			
💟 函粉套怒句	服务ID	service-dxjqn3v0 🗳		
	域名	service-d 0.gz.apigw.tencentcs.com		
拓展能力	环境	release		
「 /」 A SW /丁作法 p	URL			
	云函数			
🗞 EB事件总线 🖸	函数名称			
	命名空间	default		
	运行环境	Nodejs12.16		
	使用层	iot-layer(版本1)		
	内存	512MB		C
	超时时间	3秒		0
	固定出口IP			
		SERVERLESS = 1		4
	你位金			E
三 给产品打个分 ⊙				

5. 在函数服务中,可以看到如下图所示的对应的代码模板。单击 app.js,复制示例代码到 app.js 文件中,注意无需全覆盖。

Serverless	← http-e	x 正常	函数服务帮助文档 🖸
₩ 概览	函数管理	函数管理	版本: \$LATEST ▼ 操作 ▼
❷ 函数服务	版本管理	25.¥17.第 25.11.217 戶放頂 此物法自 口士杰达	
诊 Serverless 应用	别名管理		
高级能力	触发管理	提交方法 ⑦・在线编辑 ▼ 运行环境 Nodejs 12.16 Nodejs 12.16	开发教程 🗹 下载 ▼
\$ 层	函数 URL	ジ 编辑 选择 查看 转到 … ← →	
🛾 函数套餐包	监控信息	资源管理器 … JS app.js ×	未部署 🛄 …
拓展能力	日志查询	> vscode 1 const express = require('express')	
C ASW工作流 II	并发配额	<pre>> src 2 const path = require('path') > layer</pre>	in sansan Harran Marana Marana
№ EB事件总线 12	部署日志	<pre></pre>	
三 给产品打个分 🕥		19 }) 20 21 app.get('/user/:id', (reg. res) => {	



需要将下述代码复制到 app.js 的代码,请覆盖原 app.js 中的 const 开头的3行代码,原 app.js 中的 app.get 方法无需 覆盖。

```
app.use(bodyParser.json());
// 鉴权信息校验的代码在上文已经解释过,这里使用Middleware来简化代码
   next(); // 身份校验通过,继续执行后续的中间件和路由处理器
   res.status(403).send('Forbidden'); // 身份校验失败,返回403 Forbidden 状态码
// 注意接收规则引擎转发的数据接口为 POST 类型
app.post('/test', checkSignatureMiddleware, (req, res) => {
 const params = req.body.payload.params;
```

6. 上述步骤完成后,需要引入两个依赖包,分别是 body-parser 与 js-sha1。在代码编辑器的上方选择终端 > 新建终端,然
 后在底部终端中输入命令: cd src 进入 src 目录,接着分别输入命令 npm install body-parser 和
 npm install js-sha1 安装两个依赖包。

腾讯云



副 概定 函数管理 ② 函数服务 函数管理 版本管理 版本管理 別名管理 別名管理
高級能力 触发管理 运行 运行 送行 上口 ② 层 函数 URL 近 新分貨場 北



7. 等待依赖安装完成之后,单击下图红色线框中的**部署**,即可成功部署 Web 服务,SCF 会自动启动该 Web 服务,约5秒 - 10秒后系统会提示"部署成功"。





控制台配置规则引擎

创建规则引擎

- 1. 登录物联网开发平台控制台,选择用户购买的企业实例。
- 2. 单击左侧菜单数据流转 > 规则引擎,再次单击规则引擎列表页的新建规则。
 - 规则名称: 输入 "HTTP_Forward" 或其他内容。
 - 规则描述: 输入转发某类产品下设备数据的备注信息。

规则名称 *	HTTP_Forward
	支持英文、数字、下划线的组合,最多不超过32个字符
规则描述	将智能灯设备数据转发到第三方HTTP服务
	最多不超过256个字符

3. 规则基本信息填写完成后,单击确定,即可完成规则的创建。

配置规则引擎

🔗 腾讯云

筛选数据

- 1. 在规则引擎列表选择刚创建成功的规则 HTTP_Forward,单击规则名称或操作列的管理,进入规则引擎配置页面。
- 2. 单击下图筛选数据右侧的编辑,选择要转发的设备数据源。

规则引擎 / HTTP_Forward	帮助文档 亿
基本信息	编辑
规则名称 HTTP_Forward	
规则状态 已 禁用	
规则描述	
筛选数据 ①	编編 SoL機械
字段 Topic \$(productid)\$(devicename)/event	
祭件	
当前SQL SELECT FROM '\$(productid)\$(devicename)/event'	
行为操作	0
添加行为操作	C
	E3
转发错误行为操作	=
添加行为操作	4

- 3. 在编辑规则页面,输入或选择如下信息。
 - 字段:输入"*",表示将设备上报的所有 JSON 数据进行转发。
 - Topic: 首先选择对应的产品。选择入门文档中所创建的产品**车载设备**;第二步选择设备,请选择**全部设备**,表示规格引 擎将转发该产品下所有的设备;最后一步选择转发的数据,请选择**物模型属性上报**。



字段*	*	6
	【 仅支持**'、 ','、''、'('、')'、'_'、单引号、 空格、字母和数字,不为空,最多不 超过300个字符	
opic *	车载设备	
	全部设备	
	物模型属性上报	
	名称命名支持字母、数字、下划线、"("、")"、"\$"、"{"、"}"、"," 组合;不同层级之 间用 / 分层。+表示一级,使用/+/命名,不能/+aaa/;长度限制为1-64位。	7
条件	选填	
	不能有中文或中文字符,最多不超过300个字符	

4. 单击确定按钮,将返回配置规则页面,系统自动生成如下图所示对应的 TOPIC 及筛选数据 SQL。

规则引擎 / HTTP_Forward	帮助文档 🖸
基本信息	编辑
规则名称 HTTP_Forward	
规则状态 已禁用	
规则描述	
等选数据 ⑦	编辑 SQL调试
字段 •	
Topic Sthing/up/property/FT CKN/+	
条件	
当前SQL SELECT * FROM '\$thing/up/property/F: KN/+*	
行为趣作	9
· · · · · · · · · · · · · · · · · · ·	C
	m
	+

配置规则引擎

行为操作配置

1. 在规则引擎列表选择刚创建成功的规则 HTTP_Forward,单击规则名称或操作列的管理链接,进入规则引擎配置页面。



规则引擎 / HTTP_Forward	腾讯云IoT技术交流群 🖸 使用指南	3
基本信息	编辑	
规则名称 HTTP_Forward		
规则状态 已禁用		
规则描述 将智能灯设备数据转发到第三方HTTP服务		
		1
筛选数据⑦	编辑 SQL调试	
字段 ・		
Topic Sthing/up/proper		
条件		
当前SQL SELECT * FROM '\$thing/up/prop		
添加行为操作		
	רו	1

2. 下图行为类型选择 "数据转发到第三方服务 (Forward)"、选择 API 地址类型选择 "使用已有 HTTP 服务地址"。 HTTP 地址则为 此步骤 中复制 SCF 自动生成的 URL 地址,并在最后加上 /test 。类似 https://service.****.apigw.tencenttcs.com/release/test 。



添加规则	×	
 ● 将筛选后的数据转发到第三方HTTPS服务中。点击 查看文档 ☑ 了解如何开发HTTP服务接收物联网平台数据 		
行为类型		
数据转发到第三方服务 (Forward) ▼		
● 使用已有HTTP服务地址		
v 增加鉴权token		
Token: 🚯 *		
test 🥥		
保存取消		

3. 请勾选 "增加鉴权 Token", 且输入 test, 该 Token 必须与 SCF 函数服务中 app.js 里定义的 Token 值一致。

4. 单击保存,完成规则行为配置。

5. 返回规则引擎列表,将该规则的状态设置为 **启用**。

模拟验证

- 参见 "设备接入及上报数据",使用模拟程序上报数据。注意上报的数据相关产品需要与规则引擎筛选的目标产品、设备保持一致。若不一致 SCF 日志查询将无法打印输出数据。
- 2. 进入 SCF 的函数服务,单击日志查询,可查看到 SCF 打印的日志,可完整打印出设备上报的物模型属性。



腾讯云



快速体验平台 MQTT.fx 快速接入物联网开发平台

最近更新时间: 2024-10-13 10:58:21

操作场景

MQTT.fx 是目前主流的 MQTT 桌面客户端,它支持 Windows、 Mac、Linux 操作系统,可以快速验证是否可与物联网开 发平台(简称平台)进行连接,并发布或订阅消息。更多 MQTT 协议介绍请参见 MQTT 协议介绍。 本文档主要介绍如何使用 MQTT.fx 将设备连接到腾讯云物联网开发平台,如何通过 MQTT.fx 配置 MQTT Client ID、User Name 及 Password 等参数,如何向平台发布消息并订阅消息。本文以 MQTT.fx 1.7.1 for Windows 版本为例,不限于

V1.7.1,最新 MQTT.fx 5.3 同样支持。

操作步骤

下载 MQTT.fx 并连接平台

- 1. 下载并安装 MQTT.fx 客户端。
- 2. 打开 MQTT.fx 客户端程序,单击设置。
- 3. 进入设置页面,并单击"+",创建一个新的配置文件,输入自定义名称 Profile Name,Profile Type 选择 MQTT Broker。

MQTT.fx - 1.7.1					- 🗆 🗙
File Extras Help					
loTCloud 🗸 🗘 Connec	t Disconnect				•
Publish Subscribe Scripts Broker Status Log	Edit Connection Profiles Alot InTCloud	Profile Name	- - ×	0.50 0.51 0.52	Retained
	AZM Eclipse local mosquitto	MQTT Broker Modile Settings MQTT Broker Modiles Settings Broker Address Broker Addres Broker Address		061 062	Related
	创建 人名尔雷里文州				
	创建——个新的配直义件				



4. 填写 MQTT Broker Profile Settings 和 General 相关信息。

MQTT.fx - 1.7.1			- 🗆 X
File Extras Help			
ToTCloud 🗸 🔅 Connect	Disconnect		•
Publish Subscribe Scripts Broker Status Log	Distriction Profiles Aiot IsriCland MAM Edges Isral mosquito	Profile Name Profile Type MQTT Broker • MQTT Broker Profile Settings Broker Profile Settings Broker Profile Settings Client ID Generate Centeral User Credentials SSL/TLS Proxy LWT	Coll Odd Robed Co
		Connection Timeout 30 Koop Allev Interval 40 Clean Session V Auto Records Max Intight 50 MQTT Version V Use Default Clear Fublish History Clear Subscription History	
	+ -	Revert Canod OK Apply	

参数说明

参数	说明
Profile Name	用户自定义名称。
Broker Address	MQTT 服务器连接地址,广州域设备填入: PRODUCT_ID.iotcloud.tencentdevices.com,这里 PRODUCT_ID 为变量参数,用 户需填入创建产品时自动生成的产品 ID,例如 T****DS8G.iotcloud.tencentdevices.com。
Broker Port	MQTT 服务器连接端口,填入:1883。本文主要针对密钥认证类型的产品,端口必须是 1883,如果您想通过8883接口接入,建议使用证书认证型产品自行接入。
Client ID	MQTT 协议字段,按照物联网通信约束填入:产品 ID + 设备名, 如:"TXXXXDS8Gdev001",TXXXXDS8G 是产品 ID,dev001 是设备名称。
Connection Timeout	连接超时时间(秒)。
Keep Alive Interval	心跳间隔时间(秒)。
Auto Reconnect	断网自动重连。

5. 单击 User Credentials,填写 User Name 和 Password。

User Name: MQTT 协议字段,按照物联网通信约束填入:产品 ID + 设备名 + SDKAppID + connid+expiry。
 创建完产品即可在产品列表页和产品详情页查看 ProductID,例



如:"TO****DS8Gdev001;12010126;E4F3Q;1591948593",仅替换示例中的产品 ID + 设备名即可,后面的三 个参数本身由平台提供的设备接入 SDK 自动生成,也可由平台提供的 生成小工具 自动生成。

○ **Password**: Password 必须填写,用户可以使用平台提供的 生成小工具 自动生成 Password,也可以按照文档 手 动生成 Password 。

Edit Connection Profiles			– 🗆 X
Alot			
Io I Cloud M2M Eclipse	Profile Name		
local mosquitto	Profile Type	MQTT Broker	
	MQTT Broker Profile Settings		
	Broker Address		
	Broker Port	1883	
	Client ID		Generate
	General User Credentials	SSL/TLS Proxy LWT	
	User Name Password	•••••••	
+ -	Revert	(Cancel OK Apply

- 6. 完成以上步骤设置后,单击 Apply 和 OK 进行保存,并在配置文件框中选择刚才创建的文件名,单击 Connect。
- 7. 当右上角圆形图标为绿色时,说明已成功连接物联网开发平台,即可进行发布和订阅等操作。

MQTT.fx - 1.7.1		- o x
File Extras Help		
loTCloud	- 🔅 Connect Disconnect	•••
Publish Subscribe Scripts B	roker Status Log	
»	Publish	Q550 Q651 Q652 Retained @(*

生成 UserName 与 Password

1. 进入 控制台,单击"公共实例",创建项目后,单击项目名称进入产品开发页,单击新建产品,输入产品信息。

<u>注意:</u>			
产品品类需选择	"智慧生活/电工照明/灯",	,系统会自动创建物模型属性、	事件等。

🔗 腾讯云

新建产品	
产品名称*	
产品品类	支持中义、央义、数字、下划线、经格(非自尾字符)、中央义括号、-、@、\、的组合,最多不超过40个字符 标准品类 自定义品类
	智慧生活 / 电工照明 / 灯 🖍 😒
设备类型	设备 网关 子设备
通信方式 *	Wi-Fi 🔹
	请根据业务场景正确选择产品的通信方式,否则会影响后续产品开发
认证方式	密钥认证 证书认证
数据协议	物模型 自定义透传 ①
描述	选填
	最多不超过80个字符
确定	取消

✓ 物模型	〉 🕜 设备	яд > (交互开发 🔿	4 设备调试	\$ 5	批量投产
() 设备调	式提供真实、虚拟设备调调	(功能, 便于测试设备上持	役、接收数据是否正常,	可创建测试设备后进行;	周试	
新建设备	虚拟设备调试			设备名称	輸入设备名称搜索	Q
设备名称	状态	激活时间	最后上线时间	操作	绑定网关	

3. 下载网页小工具 生成小工具 并解压缩后,在目录中可查看到以下3个文件。



4. 单击平台设备调试中创建的设备名称,获取设备的三元组信息"产品 ID"、"设备名称"和"设备密钥"。

$\boldsymbol{\heartsuit}$	腾讯云	

产品开发 / 开关	
 ✓ 物模型 ✓ 设备开发 ✓ 交 	医互开发 〉 4 设备调试 〉 5 批量投产
← dev001	
() 从2023-08-01开始,日志存储时长将由原来的7天改成3天。	
设备信息 在线调试 云端诊断日志 设备云述	端日志 设备本地日志 扩展信息
设备信息	
设备名称	产品ID 422 值 所属产品 开关
设备密钥 J+: == 后	设备创建时间 2023-07-03 17:43:34 最后上线时间 -
激活时间 -	设备状态 未激活 固件版本 -
标签信息	

5. 打开 Chrome 浏览器,并打开 sign.html 文件,显示如下图。将上一步中"产品 ID"、"设备名称"、"设备密钥"信息分别复制到对应的 ProductID、DeviceName、DeviceSecret 文本框中,签名算法默认选择 HMAC-SHA256,单击 Generate,网页工具自动生成 UserName 与 Password。

请输入设备信息:	
ProductID:	
DeviceName:	
DeviceSecret:	
Hmac签名算法:	HMAC-SHA256 ~
	Generate
结果:	
UserName:	
Password:	

6. 复制自动生成的 User Name、Password 到 MQTT.fx 的 User Credentials 区域对应的文本框。



设备发布物模型消息

选择客户端 Publish Tab,输入主题名称、Qos 等级,单击 Publish 进行发布。

示例 Topic: \$thing/up/property/ProductID/DeviceName (设备上报数据到平台的物模型 Topic,实际体验需要将 ProductID 与 DeviceName 替换成用户在平台创建的对应内容)。发布消息数据格式规范及示例,可参见文档 物模型协议。

🛞 MQTT:fx - 1.7.1			
File Extras Help			
loTCloud • 🖉 Connect Disconnect			•••
Publish Subscribe Scripts Broker Status Log			
> Sthing/up/property/ Publish 输入设备具备发布权限的topic名称	Q050 Q051 Q052	Retained	0 (*
「method"report, 'dentTokent" 'timestump" prover_switch1, 'color1, 'toints: 'j] 」 此处输入设备发布topic的具体信息			

发布结果可通过控制台中对应的设备云端日志查询到上报记录。

品开发 / 智 (能灯												腾讯云loT技术交流群 🗹
✔ 物模型		父 设	备开发		文 交互开发		4	设备调试		5 批量投产			
dev001													
() 从2023-	08-01开始,	日志存储田	时长将由原来	的7天改成	3天。								
设备信息	在线	调试	云端诊断	日志	设备云端日志	设备	本地日志	扩展(言息				
物模型日志	。 内	容日志	上下线日;	志	ф 🔵 еа	刷新							
日志类型 雇	鮏		▼	topic	\$thing/up/prop	er	C/dev00	01, \$thing/do	own/pro	•			
30分钟	1小时	今天	昨天	近7	7天 2023-0	7-25 17:09	~ 2023-0	07-25 17:39	Ö				
时间			通讯类型	Т	Горіс			通信	内容				
2023-07-25	17:38:09.27	72	上行	\$	Sthing/up/propert		C/dev001	{ "me "brigh	ethod":"repo htness":32 }	rt", "clientToken":"123' }}	, "timestamp":162864	16783, "params":{ "p	ower_switch":1, "color":1,

设备订阅物模型消息

选择客户端 Subscribe Tab,输入订阅主题 Topic 名称、Qos 等级,单击 **Subscribe** 进行主题订阅。订阅结果可通过 <mark>控制</mark> 台 的设备日志查询。

示例 Topic: \$thing/down/property/ProductID/DeviceName (设备订阅物模型 Topic,常用于云端通过物模型下发控



制报文至设备端,实际体验需要将 ProductID 与 DeviceName 替换成用户在平台创建的对应内容)。



进入设备在线调试,设置开关,亮度后单击发送,平台将下发控制指令。



产品开发 / 智能灯					腾讯云IoT技术交流群 IZ 使用指南
 ◇ 物模型 > ◇ 设备; ← dev001 	开发 〉 🕑 交互开发) 🚺 设备调试	〉 (5) 批量投产		
① 从2023-08-01开始,日志存储时步	长将由原来的7天改成3天。				
设备信息 在线调试 z	云端诊断日志 设备云端日志	设备本地日志 扩展信	自息		
下发指令			调试日志	清空日志	显示响应报文 🔽 自动刷新
属性调试 行为调用			时间	上下行	日志类型
- 功能名称/标识符	期望值	实时数据	▼ 2023-07	-25 17:47:03.1 ↓ 下行	设备属性控制
✓ 电灯开关(power_switch)		Я	时间	2023-07-25 17:47:03.143	
✓ 亮度(brightness)	- 50 +	% 50	日志类型 method	设备属性控制 "control"	
✔ 颜色(color)	Red v	Red	clientToken params	"v21498 ("color":0,"name":"","power_switch"	»" :1,"brightness":50}
色温(color_temp)	- 0 +	% 0	▶ 2023-07	-25 17:46:21.8 ↓ 下行	设备属性控制
✓ 灯位置名称(name)	支持英文字母、数字、常见半角符号约	0/64 - 1合			
发送重置					

之后回到 MQTT.fx 窗口,可以看到云端刚才下发的控制报文数据:

😁 MQTT.fx - 1.7.1			
File Extras Help			
loTCloud - 🔅 Connect	Disconnect		n 😑 🕒
Publish Subscribe Scripts Broker Status Log			
\$thing/down/property/			QoS0 QoS1 QoS2 Autoscrol Qv
\$thing/down/propertydev001	Dump Messages Mute Unsubscribe	\$thing/down/propertydev001	1 Q050
		Sthing/down/property, /dev001	
		12:06:0210:15:09:48:4588103 ['method':"control","clientToken":"	Cio "params":{"name":"","power_switch":1,"color":0,"brightness":50)}
Topics Collector (0)	Scan Stop OC+		
		此处可以看到刚才在云端控制台下发的设备数据	



查看日志

在 MQTT.fx 上,单击 Log 查看操作日志和错误提示日志。





使用MQTT物模型接入平台

最近更新时间: 2024-08-14 17:01:21

操作场景

物联网开发平台提供了基于MQTT协议的物模型接入方式,假设一款智能灯接入到物联网开发平台,通过物联网开发平台可以远 程控制灯的亮度、颜色、开关,并实时获取智能灯上报到开发平台的数据。本文档主要指导您如何在物联网开发平台控制台接入 智能灯。

前提条件

为了通过下面的步骤快速理解该业务场景,需要做好以下准备工作:

- 申请物联网开发平台服务。
- 拥有一台物理或虚拟的 Linux 环境,可以编译、运行 light_demo 程序。
- light_demo 在 Linux 环境下测试和验证,主要基于 Ubuntu 16.04 版本,gcc-5.4(建议至少 gcc-4.7+)。

操作步骤

控制台操作

创建项目

- 1. 登录物联网开发平台控制台,选择平台默认开通的公共实例或用户购买的企业实例。
- 2. 进入项目列表页面,单击新建项目。
 - 项目名称: 输入"智能灯演示"或其它名称。
 - 项目描述:按照实际需求填写项目描述。

新建项目	×	;
项目名称 *	智能灯演示	
	支持中文、英文、数字、下划线的组合,最多不超过20个字符	
项目描述	选填	
	最多不超过80个字符	
	保存取消	

- 3. 项目基本信息填写完成后,单击保存,即可完成新建项目。
- 4. 项目新建成功后,即可新建产品。

新建产品

1. 进入该项目的产品列表页面,单击**新建产品**。



- 2. 在新建产品页面,填写产品基本信息。
 - 产品名称: 输入"智能灯"或其它产品名称。
 - 产品品类:选择"智慧城市">"公共事业">"路灯照明"。
 - 设备类型:选择"设备"。
 - 认证方式:选择"密钥认证"。
 - 通信方式:按需选择。
 - 其它都为默认选项。

	新建产品	×
产品名称 *	智能灯	
	支持中文、英文、数字、下划线的组合,最多不超过20个字符	
产品品类	智能城市 🔹 公共事业 💌 路灯照明	r
设备类型	设备 网关 子设备	
认证方式	证书认证 密钥认证	
通信方式	Wi-Fi 2G/3G/4G 5G BLE LoRaWAN 其它 ①	
数据协议	数据模板 自定义透传	
描述	选填	
	是实际规计的公学性	
	₩25小/2016 00 1-1-13	
	保存取消	

- 3. 产品信息填写完成后,单击保存,即可完成新建产品。
- 4. 产品新建成功后,您可在产品列表页查看到"智能灯"。

定义产品物模型

选择"智能灯"类型后,系统会自动生成标准功能。

腾讯云

智能灯
产品ID
产品品类
设备类型 认证方式 通信方式
数据协议创建时间

创建设备

在设备调试页面中,单击新建设备,设备名为 dev001。

新建设备		×
所属产品	智能灯	
设备名称 *		
	支持英文、数字、下划线的组合,最多不超过48个字符	
	保存取消	

下载 Demo 程序

下载 lightdemo 例程

首先从 GitHub 下载,或执行下面的 git 命令。

git clone https://github.com/tencentyun/qcloud-iot-explorer-sdk-embedded-c.git

修改 Demo 程序

上述 git 命令执行成功后,会生成一个 qcloud-iot-sdk-embedded-c 目录。

1. $\text{H}\lambda$ qcloud-iot-explorer-sdk-embedded-c **E**aarrow.

2. 修改该目录下的 device_info.json 文件。

> vi device_info.json



<pre>{ "auth_mode":"KEY",</pre>	
"productId" ["productSecret":"YO "deviceName":	UR_PRODUCT_SECRET", .",
<pre>"key_deviceinfo":{ "deviceSecret" },</pre>	
<pre>"cert_deviceinfo": "devCertFile":"YO "devPrivateKeyFil }</pre>	UR_DEVICE_CERT_FILE_NAME", e":"YOUR_DEVICE_PRIVATE_KEY_FILE_NAME"

- 3. 将上图红色线框中的数据分别替换为控制台产品在设备调试阶段,单击选择**设备名称**进入"设备详情页"中的参数并保存。
 - 产品 ID: 将控制台的产品 ID,复制到上图 productId。
 - 设备名称: 将控制台的设备名称,复制到上图 deviceName。
 - 设备密钥:将控制台的设备密钥,复制到上图 deviceSecret。

编译

- 1. 上述配置信息修改完成后,即可编译。
- 2. 在 qcloud-iot-sdk-embedded-c 目录下执行以下命令进行编译。

./cmake_build.sh

3. 编译成功后,会在 output/release/bin 目录下生成 light_data_template_sample 执行文件。

运行 Demo 程序

- 1. 进入 output/release/bin 目录。
- 2. 输入 ./light_data_template_sample。
- 3. 运行成功后,系统输出示例如下:

```
INF|2019-05-07 21:51:33|device.c|iot_device_info_set(65): SDK_Ver: 3.0.0,
Product_ID: BKDDAHRGRX, Device_Name: dev001
DBG|2019-05-07 21:51:33|HAL_TLS_mbedtls.c|HAL_TLS_Connect(204): Connecting to
/BKDDAHRGRX.iotcloud.tencentdevices.com/8883...
DBG|2019-05-07 21:51:33|HAL_TLS_mbedtls.c|HAL_TLS_Connect(209): Setting up the
SSL/TLS structure...
DBG|2019-05-07 21:51:33|HAL_TLS_mbedtls.c|HAL_TLS_Connect(251): Performing the
SSL/TLS handshake...
INF|2019-05-07 21:51:33|HAL_TLS_mbedtls.c|HAL_TLS_Connect(269): connected with
/BKDDAHRGRX.iotcloud.tencentdevices.com/8883...
```



INF|2019-05-07 21:51:33|mqtt_client.c|IOT_MQTT_Construct(115): mqtt connect with id: ZPEm9 success DBG|2019-05-07 21:51:33|shadow_client.c|_shadow_event_handler(63): shadow INF|2019-05-07 21:51:33|light_data_template_sample.c|event_handler(222): data successfully INF|2019-05-07 21:51:33|light_data_template_sample.c|main(496): Cloud Device topicName=\$thing/down/event/BKDDAHRGRX/dev001|packet_id=35314|pUserdata=(null) 21:51:33|light_data_template_sample.c|_register_data_template_property(370): data template property=color registered. data template property=brightness registered. 21:51:33|light_data_template_sample.c|_register_data_template_property(370): template propertys Success DBG|2019-05-07 21:51:33|mqtt_client_publish.c|qcloud_iot_mqtt_publish(337): publish packetID=0 topicName=\$template/operation/BKDDAHRGRX/dev001|payload= {"type":"get", "clientToken":"BKDDAHRGRX-0"} DBG|2019-05-07 21:51:34|light_data_template_sample.c|main(602): cycle report: {"power_switch":0, "color":0, "brightness":0.000000, "name": "dev001"}},



4. Light Demo 程序定时会上报数据到开发平台,数据格式如下:

```
{"version":1, "state":{"reported":
{"power_switch":0,"color":0,"brightness":0.000000,"name":"dev001"}},
"clientToken":"BKDDAHRGRX-2"}
```

5. 继续保持 Light Demo 程序处于运行状态,然后前往控制台查看该设备的数据。

查看设备状态

- 1. 保持 light Demo 程序为运行状态。
- 进入控制台 > 产品开发 > 设备调试,可查看到设备"dev001"的状态为"上线"状态,表示 Demo 程序已成功连接上开 发平台。
- 3. 单击查看,可进入设备详情页。
- 4. 单击设备属性,可查询设备上报到开发平台的最新数据及历史数据。
 - 当前上报数据的最新值:会显示设备上报的最新数据。
 - 当前上报数据的更新时间:显示数据的更新时间。

设备信息	设备属性	设备日志	设备事件	设备行为	设备上下线日志	在线调试	扩展信息	设备调试日志
标识符	功	能名称	历史数据		数据类型	最新值	更新	时间
power_switch	电》	灯开关	查看		布尔型	-	-	



5. 单击查看,可查看某个属性的历史上报数据。

查看设备通信日志

- 1. 单击设备日志,可查询该设备某段时间范围的所有上下行数据。
 - 上行: 上行指设备端上报到开发平台的数据。
 - 下行:下行指从开发平台下发到设备的数据。

上行 下行 30分钟 1小时 今天 昨天 近7天 2020-09-02 00:00 ~ 2020-09-02 23:59 首	
时间 日志类型 通信内容	
当前列表为空	

在线调试

1. 当 Light Demo 成功连接到物联网开发平台后,您可在控制台设备调试列表,单击调试,进入在线调试。

设备属性 设备日	1志 设备事件	设备行为	设备上下线日志	在线调试	扩展信息.	
下发指令					通信日志	自动刷新
功能名称/标识符	期望值				设备上报数据:2019-11-12 10:36:32 {	复制▲
电灯开关 (power_switch)					"method": "report", "clientToken": "3CBMGNNP00-315", "params": {	
颜色(color)	Red	v			"brightness": 68 } }	
亮度(brightness)	_ 6	68 +	%		下发控制指令:2019-11-12 10:36:31 {	
灯位置名称(name)	dev001 支持英文字母。3	数字. 常见半角符	6/64 号组合		<pre>"method": "control", "clientToken": "clientToken-b9e0e0f2-1830-4c9b-9ffa-3f85a0c62851", "params": {</pre>	
					"brightness": 68 } }	
发送清空					设备上报数据:2019-11-12 10:36:28 (•

- 2. 将亮度设置为68,颜色设置为"Red",单击发送。
- 3. 查看 Light Demo 程序,可查看到成功接收到下发的数据。

4. 通信日志会显示如下日志,表示成功下发了指令到设备端。

"method": "control",		
"clientToken": "123",		
"params": {		
"power_switch": 1,		





5. 查看通信日志,即可查看到设备成功接收到下行指令,并上报最新数据到开发平台的详细日志。



使用自定义透传上下行消息

最近更新时间: 2024-09-30 17:54:51

操作场景

MQTT 自定义透传接入平台主要用于设备通过 DTU 或边缘网关接入物联网平台的情况。当设备无法直接使用物联网开发平台的 物模型业务格式时,可以选择使用 MQTT 透传报文的模式。在这种应用场景下,可以利用平台支持的 MQTT 自定义透传报文 能力将设备数据上传到云端。通过本文档可以快速了解如何使用平台将自定义透传协议类的设备上云,以及如何通过云 API 远程 控制使用自定义透传协议的设备。

前提条件

为了快速了解该业务场景,需要提前做好以下准备工作:

- 申请物联网开发平台服务。直接 注册腾讯云账号 即可立即开通物联网开发平台。
- 拥有一台物理或虚拟的 Linux 环境,可以编译、运行 C SDK 中的 raw_data_mqtt_sample 程序。
- raw_data_mqtt_sample 在 Linux 环境下测试和验证,主要基于 Ubuntu 16.04 版本,gcc-5.4(建议至少 gcc-4.7+)。

操作步骤

创建项目

- 1. 登录 物联网开发平台控制台,单击"公共实例"框进入项目列表页面。
- 2. 单击新建项目,在新建项目页面,填写项目基本信息。

新建项目	×
项目名称 *	
	支持中文、英文、数字、下划线的组合,最多不超过20个字符
项目描述	选填
	最多不超过80个字符
	保存取消

- 项目名称: 输入"自定义透传设备上云"或其他名称。
- 项目描述:按照实际需求填写项目描述。
- 3. 项目基本信息填写完成后,单击保存,即可完成新建项目。
- 4. 项目新建成功后,即可新建产品。

新建产品



- 1. 点击创建成功的项目"自定义透传设备上云",单击**产品开发**,进入该项目的产品列表页面,单击**新建产品**。
- 2. 在新建产品页面,填写产品基本信息。

新建产品	
产品名称 *	raw_data
	支持中文、英文、数字、下划线、空格(非首尾字符)、中英文括号、-、@、\、/的组合,最多不超过40个字符
产品品类	标准品类 自定义品类
[智慧生活 / 电工照明 / 灯 💉 🗴
设备类型	设备 网关 子设备
通信方式 *	Wi-Fi 🔹
	请根据业务场景正确选择产品的通信方式,否则会影响后续产品开发
认证方式	密钥认证 证书认证
数据协议	物模型 自定义透传 ①
描述	选填
	早冬天把过200个字位
	ערד דייעאא

- 产品名称:输入"raw_data"或其他产品名称。
- 产品品类:选择标准品类,并选择"智慧生活->电工照明->灯"。
- 设备类型:选择"设备"。
- 认证方式:选择"密钥认证"。
- 通信方式:默认 Wi-Fi。
- 数据协议:选择"自定义透传"
- 描述: 根据需要进行填写。

3. 产品信息填写完成后,单击保存,即可完成新建产品。

4. 产品新建成功后,您可在产品列表页查看到"raw_data"。

设备开发

单击**设备开发 > Topic 列表 > 透传 Topic**,当产品的数据协议为"自定义透传"类型时,设备开发界面如下图所示。



^空 品开发 / raw_data		
✓ 物模型 > 2 设备开发 > (3) 交互开发 〉 (4) 设行	备调试 〉 (5) 批量投产
设备开发 Topic列表 云端解析		
Topic权限	操作权限	备注
\$thing/up/raw/WC 16X/\${deviceName}	发布	透传协议上行,并支持云端解析成JSON
\$thing/down/rawW ⊃X/\${deviceName}	订阅	透传协议下行,并支持云端JSON转换成 透传协议
上 一 步 下 一 步		

() 说明:

- 平台默认创建了 Topic,其中上图红色线框为使用自定义透传协议的系统约定的上行、下行 Topic。
- 平台支持用户使用自定义 Topic 自由定义,以 \$thing 开头的 Topic 支持云端脚本解析能力,若需要通过平台云端 JS 脚本解析自定义透传数据为物模型格式,则必须使用 \$thing/up/raw 开头的 Topic。

创建设备

1. 单击设备调试 > 新建设备,输入设备名称为"dev001",单击保存。

新建设备	×
所属产品 ra	w_data
设备名称 *	dev001
Ŕ	2.持英文、数字、下划线的组合,最多不超过48个字符
	保存取消

2. 单击**设备名称** "dev001" ,进入设备详情页,查看产品 ID、设备名称和设备密钥等信息。此部分信息将会在编译 raw_data_mqtt_sample 时需要使用。



产品开发 / raw	/_data										
✓ 物模型		🖌 设备开发	> (文 交互开发	: >	4 设	全调试		5 批量投产		
← dev001											
() 从2023-	08-01开始,日	日志存储时长将由原	来的7天改成3天	₹.							
设备信息	在线调	试 云端诊断	所日志 ù	设备云端日志	设备本	地日志	扩展信	息			
设备信息											
设备名称	dev001 🛅				产品ID	1	6X 🖬			所属产品	raw_data
设备密钥	-	ryjTO0Q==	ē		设备创建时间	2023-11-	04 18:14:28			最后上线时间	-
激活时间	-				设备状态	未激活				固件版本	-
标签信息											
设备标签	无标签信息										
设备标签	无标签信息										

下载 Demo 程序

下载 raw_data_mqtt_sample 例程

进入 Linux 主机 (建议使用腾讯云 CVM),下载 SDK 的方式有以下两种:

- 从 GitHub 下载 SDK 直接上传至云主机。
- 执行以下 git 命令获取。若 git clone 执行失败则需要设置云主机安全组的入站、出站规则是否正确。

git clone https://github.com/tencentyun/qcloud-iot-explorer-sdk-embedded-c.git

编译

1. 在 qcloud-iot-sdk-embedded-c 目录下执行以下命令进行编译。

./cmake_build.sh

2. 编译成功后,可在 output/release/bin 目录下生成 raw_data_mqtt_sample 执行文件及 device_info.json 文件。 若编译失败,需要检查 gcc 版本是否是8.5以上。

修改配置文件

- 1. 进入 output/release/bin 目录。
- 2. 输入以下命令,修改该目录下的 device_info.json 文件。

vi device_info.json



3. 将红色线框中的数据分别替换为控制台创建的设备 "dev001" 对应的产品 ID、设备名称、设备密钥信息。

{ "auth_mode":"KEY",
<pre>"productId" ['", "productSecret":"YOUR_PRODUCT_SECRET", "deviceName":"",</pre>
<pre>"key_deviceinfo":{ "deviceSecret" },</pre>
<pre>"cert_deviceinfo": "devCertFile":"YOUR_DEVICE_CERT_FILE_NAME", "devPrivateKeyFile":"YOUR_DEVICE_PRIVATE_KEY_FILE_NAME" }</pre>

- 产品 ID: 设备 "dev001" 的产品 ID, 复制到上图 productId。
- 设备名称: 设备 "dev001" 的设备名称,复制到上图 deviceName。
- 设备密钥: 设备 "dev001" 的设备密钥,复制到上图 deviceSecret。
- 4. 保存 device_info.json 文件即可。

运行 Demo 程序

- 1. 进入 output/release/bin 目录。
- 2. 输入 ./raw_data_mqtt_sample 并执行。
- 3. 运行成功后,输出示例如下图红框所示。



4. Demo 程序会上报数据到平台,上报数据为下方16进制数据。

AA 55 01 00 00 00 00 00 01 01 1E 00

查看设备上报数据

- 1. 进入物联网开发平台控制台 > 产品开发 > 设备调试,单击设备名称 "dev001" 进入设备详情页。
- 2. 单击设备云端日志,系统会显示"透传日志",即可在控制台查看该设备上报的数据。



产品开发 / raw_data		
→ 物模型 〉 → 设备开发 〉	交互开发 🧹 设备	调试 > 5 批量投产
← dev001		
() 从2023-08-01开始,日志存储时长将由原来的7天改	7成3天。	
设备信息 在线调试 云端诊断日志	设备云端日志 设备本地日志	扩展信息
透传日志内容日志上下线日志	🗘 🔵 自动刷新	
原始日志 解析日志		
上行 下行 30分钟 1小时	今天 昨天 近3天 2023-11-04	4 17:51 ~ 2023-11-04 18:21 🛅
时间通讯类型	Торіс	数据
2023-11-04 18:20:13.072 上行	\$thing/up/raw3M6X/dev001	qlUBAA0AAAABAmAA
2023-11-04 18:20:07.570 上行	\$thing/up/raw/ M6X/dev001	qIUBAAwAAAABABoA
2023-11-04 18:20:02.070 上行	\$thing/up/raw/\\'3M6X/dev001	qIUBAAsAAAABAVsA
2023-11-04 18:19:56.571 上行	\$thing/up/raw/W 6X/dev001	qlUBAAoAAAABAgsA
2023-11-04 18:19:51.069 上行	\$thing/up/raw/\´``6X/dev001	qIUBAAkAAAABARIA
2023-11-04 18:19:45.569 上行	\$thing/up/raw/W}X/dev001	qIUBAAgAAAABAicA

3. 上图 Topic 列为设备向云端发布消息时的 Topic,数据列红色线框中的数据为平台 base64 编码后的数据。

使用自定义透传模式远程控制设备

1. 当需要从云端控制使用自定义透传协议上云的设备时,可参考 设备透传指令控制 云 API 直接使用 在线调试 工具进行调试。

2. API Explorer 调试工具如下图所示。



API Explorer	物联网开发平台 (IOTE	XPLORER) 🔻	产品体验,您说了算 月
搜索接口,支持中英文搜	史索 Q	PublishMessage iotexplorer 2019-04-23 查看API文档	代码生成 在线调用 签名串生成 参数说明 问题反馈 查看文档 数述
固件升级相关接口	~	输入参数	
设备管理相关接口	^	Region 🚯	注意:通过API发送请求等同于真实操作,请小心进行 占主下面的"发送请求"等同于真实操作,请小心进行
批量禁用启用设备		华南地区 (广州) ap-guangzhou 🔻 🖍	黑山下湖的"发达",赤约东湖,50°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°
批量解绑子设备		参数输入方式	的结果、响应头等怕大信息,供您调试、参考。
发布RRPC消息		表单 JSON 参数推荐	发送请求 请求耗时: 1165ms
设备透传指令控制		ProductId [*] 😔 🛈	
发布广播消息		W: 3M6X	响应结果 响应头 真实请求
获取指定网关设备的子	已设备列表	DeviceName [*] 😄 🤅	{ r
获取设备绑定的用户列	し表	dev001	"Response": {
获取产品的设备列表		Topic [*] 😀 🛈) }
生成单个设备绑定的签	经名	\$thing/down/raw/W 3X/dev001	查看 f23d9c51-6600-4baa-a6df-8127d998f865 的诊断信息 亿
直接绑定设备和家庭		Pavload 🛃 💮 🚯	
查询绑定到家庭的网关	长设备的子设备列表		
获取网关绑定的子设备	钢表	diophysica 💽	
查询设备绑定的网关设	备	Qos (选填) [*) 😧 🚯	
查看设备详情		0	
批量删除设备		PayloadEncoding (选填) [*] ② ①	
删除设备		base64	
展示英文接口 😂) 吐槽	发起调用 调用历史 展示所有参数 ▼	

- **3.** 在发送透传报文指令至设备端前,需运行 ./raw_data_mqtt_sample -1 10 命令连接到平台,保持连接以便接收平台下发的指令报文。
- 4. 当看到 demo 程序接收到云端下发的指令报文时,如下图红色线框中的报文为16进制。

DBG 2023-11-04 18:4	:22 mqtt_client_publish.c qcloud_iot_mqtt_publish(346): publish packetID=0 topicName=\$thing/up/raw/W3VZ6JBM6X/dev001 payload=�
DBG 2023-11-04 18:4	28 raw_data_sample.c _publish_raw_data_msg(223): raw_data published dump:
AA 55 01 00 27 00	0 00 01 02 3C 00
DBG 2023-11-04 18:4	:28 mqtt_client_publish.c qcloud_iot_mqtt_publish(346): publish packetID=0 topicName=\$thing/up/raw/W3VZ6JBM6X/dev001 payload=�
DBG 2023-11-04 18:4	:33 raw_data_sample.c _publish_raw_data_msg(223): raw_data published dump:
AA 55 01 00 28 00	0 00 01 02 40 00
DBG 2023-11-04 18:4	:33 mqtt_client_publish.c qcloud_iot_mqtt_publish(346): publish packetID=0 topicName=\$thing/up/raw/W3VZ6JBM6X/dev001 payload=�
00012020 11 01 101	
INF 2023-11-04 18:4 DBG 2023-11-04 18:4	:33 raw_data_sample.c on_raw_data_message_callback(235): Receive Message With topicName:\$thing/down/raw/W: 6X/dev001, payloadlen :33 raw_data_sample.c on_raw_data_message_callback(238): raw_data reveived dump:
INF 2023-11-04 18:4 DBG 2023-11-04 18:4 AA 55 01 00 08 00	:33 raw_data_sample.c on_raw_data_message_callback(235): Receive Message With topicName:\$thing/down/raw/W: 6X/dev001, payloadlen :33 raw_data_sample.c on_raw_data_message_callback(238): raw_data reveived dump: 0 00 01 02 27 00
INF 2023-11-04 18:4 DBG 2023-11-04 18:4 AA 55 01 00 08 00 DBG 2023-11-04 18:4	:33 raw_data_sample.c on_raw_data_message_callback(235): Receive Message With topicName:\$thing/down/raw/W: 6X/dev001, payloadlen :33 raw_data_sample.c on_raw_data_message_callback(238): raw_data reveived dump: 0 00 01 02 27 00 :39 raw_data_sample.c _publish_raw_data_msg(223): raw_data published dump:
AA 55 01 00 29 00	:33 raw_data_sample.c on_raw_data_message_callback(235): Receive Message With topicName:\$thing/down/raw/W
AA 55 01 00 29 00 DBG 2023-11-04 18:4 AA 55 01 00 08 00 DBG 2023-11-04 18:4 AA 55 01 00 29 00 DBG 2023-11-04 18:4	<pre>:33 raw_data_sample.c on_raw_data_message_callback(235): Receive Message With topicName:\$thing/down/raw/W: 6X/dev001, payloadlen :33 raw_data_sample.c on_raw_data_message_callback(238): raw_data reveived dump: :30 00 01 02 27 00 :39 raw_data_sample.c _publish_raw_data_msg(223): raw_data published dump: :30 00 01 02 0A 00 :39 mqtt_client_publish.c qcloud_iot_mqtt_publish(346): publish packetID=0 topicName=\$thing/up/raw/W3VZ6JBM6X/dev001 payload= </pre>
AA 55 01 00 02 00 DBG 2023-11-04 18:4 AA 55 01 00 08 00 DBG 2023-11-04 18:4 AA 55 01 00 29 00 DBG 2023-11-04 18:4 DBG 2023-11-04 18:4	<pre>:33 raw_data_sample.c on_raw_data_message_callback(235): Receive Message With topicName:\$thing/down/raw/W: 6X/dev001, payloadlen :33 raw_data_sample.c on_raw_data_message_callback(238): raw_data reveived dump: :39 raw_data_sample.c _publish_raw_data_msg(223): raw_data published dump: :39 aw_data_sample.c _publish_raw_data_msg(223): raw_data published dump: :39 mqtt_client_publish.c qcloud_iot_mqtt_publish(346): publish packetID=0 topicName=\$thing/up/raw/W3VZ6JBM6X/dev001 payload= :44 raw_data_sample.c _publish_raw_data_msg(223): raw_data published dump:</pre>
AA 55 01 00 29 00 DBG 2023-11-04 18:4 AA 55 01 00 08 00 DBG 2023-11-04 18:4 AA 55 01 00 29 00 DBG 2023-11-04 18:4 DBG 2023-11-04 18:4 AA 55 01 00 2A 00	<pre>33]raw_data_sample.c on_raw_data_message_callback(235); Receive Message With topicName:\$thing/down/raw/W: 6X/dev001, payloadlen 33]raw_data_sample.c on_raw_data_message_callback(238): raw_data reveived dump: 30 00 01 02 27 00 339 raw_data_sample.c _publish_raw_data_msg(223): raw_data published dump: 30 00 01 02 0A 00 339 mqtt_client_publish.c qcloud_iot_mqtt_publish(346): publish packetID=0 topicName=\$thing/up/raw/W3VZ6JBM6X/dev001 payload= 334 raw_data_sample.c _publish_raw_data_msg(223): raw_data published dump: 330 00 01 02 0A 00 339 mqtt_client_publish.c qcloud_iot_mqtt_publish(346): publish packetID=0 topicName=\$thing/up/raw/W3VZ6JBM6X/dev001 payload= 344 raw_data_sample.c _publish_raw_data_msg(223): raw_data published dump: 30 00 01 02 5C 00</pre>
AA 55 01 00 29 00 DBG 2023-11-04 18:4 AA 55 01 00 08 00 DBG 2023-11-04 18:4 AA 55 01 00 29 00 DBG 2023-11-04 18:4 DBG 2023-11-04 18:4 AA 55 01 00 2A 00 DBG 2023-11-04 18:4	<pre>33]raw_data_sample.c on_raw_data_message_callback(235); Receive Message With topicName:\$thing/down/raw/W: 6X/dev001, payloadlen 33]raw_data_sample.c on_raw_data_message_callback(238): raw_data reveived dump: 30 00 01 02 27 00 339 raw_data_sample.c _publish_raw_data_msg(223): raw_data published dump: 30 00 01 02 0A 00 339 mqtt_client_publish.c qcloud_iot_mqtt_publish(346): publish packetID=0 topicName=\$thing/up/raw/W3VZ6JBM6X/dev001 payload= 44/mqtt_client_publish.c qcloud_iot_mqtt_publish(346): publish packetID=0 topicName=\$thing/up/raw/W3VZ6JBM6X/dev001 payload= 44/mqtt_client_publish.c qcloud_iot_mqtt_publish(346): publish packetID=0 topicName=\$thing/up/raw/W3VZ6JBM6X/dev001 payload= </pre>



设备信息	在线	凋试	云端诊断日志	设备	云端日志	设备本述	地日志	扩展信息	3	
透传日志	内容	日志	上下线日志	φ (自动刷新	斩				
原始日志	解析	日志								
上行	下行	30分银	中 1小时	今天	昨天	近3天	2023-11-04	18:11	~ 2023-11-04 18:41	Ħ
时间			通讯类型	Торіс				数据		
2023-11-04	4 18:40:55.62	0	上行	\$thing/u	ip/raw/W3՝	(/dev0	01	qIUBAC	wAAAABABkA	
2023-11-04	4 18:40:50.11	7	上行	\$thing/u	p/raw/W3	/dev0	01	qlUBAC	sAAAABASsA	
2023-11-04	4 18:40:44.61	7	上行	\$thing/u	p/raw/W3	/dev0	01	qIUBAC	oAAAABAlwA	
2023-11-04	4 18:40:39.11	8	上行	\$thing/u	p/raw/W3	/dev0	01	qlUBAC	kAAAABAgoA	
2023-11-04	4 18:40:33.61	2	上行	\$thing/u	p/raw/W3	/dev0	01	qlUBAC	gAAAABAkAA	
2023-11-04	4 18:40:32.53	4	下行	\$thing/d	lown/raw/W3	/de	ev001	qlUBAA	gAAAABAicA	
2023-11-04	4 18:40:28.11	0	上行	\$thing/u	p/raw/W3	/dev0	01	qIUBAC	cAAAABAjwA	

云端控制设备入门

最近更新时间: 2024-10-03 11:18:11

操作场景

设备接入到物联网开发平台后,一般会通过用户的业务系统发起控制设备的操作。例如,智能公寓场景 App 远程为房间门锁设置 密码,可以通过物联网开发平台提供的云 API 或在线调试功能下发消息至设备侧,以实现用户业务系统通过物联网平台控制、管 理设备。

前提条件

为了快速体验控制设备业务功能,需要提前做好以下准备工作:

- 注册腾讯云账号,申请 物联网开发平台服务。
- 参见 MQTT.fx 快速接入物联网开发平台,能够成功模拟设备连接平台,并能订阅物模型消息用于接收远控 API 下发的消息。

使用 API Explorer 控制设备

- 1. 登录 物联网开发平台 后,访问 设备远程控制 API ,可查看平台提供的远控 API 服务,单击**点击调试**,可进入 API Explorer 在线 API 调试工具。
- 进入设备远程控制 API 调试页面,输入必选参数。然后单击右侧区域的"发送请求"按钮。API Explorer 将返回响应结果。

△ 注意:

在发送请求前,被控设备已成功连接平台,并且成功订阅了物模型下行 Topic。

○ 输入参数 Region: 选择**华南地区 (广州) ap-guangzhou**。

- 输入参数 ProductId: 请输入 MQTT.fx 工具或其他模拟工具或真实物理设备连接到平台的产品 ID。
- 输入参数 DeviceName: 请输入连接到平台的 DeviceName。
- 输入参数 Data: 请输入物模型 JSON,例如{"brightness":20}。



API Explorer 物联网开发平台 (IOTI	EXPLORER) 🔻	产品体验,您说了算 用户之声 🗹
搜索接口,支持中英文搜索 Q	ControlDeviceData iotexplorer 2019-04-23 查看API文档	代码生成 在线调用 签名串生成 参数说明 问题反馈 查看文档 数据模拟
固件升级相关接口 ~	输入参数	
设备管理相关接口		1 注意:通过API反达请求等同于具实操作,请小心进行 点击下面的"发送请求"按钮,系统会以POST的请求方法发送您在左侧填写的参数到对应的接口,该
批量禁用启用设备	Kegion ()	操作等同于真实操作,建议您仔细阅读产品计费文档了解费用详情,同时系统会给您展示请求之后 的结果、响应头等相关信息,供您调试、参考。
批量解绑子设备		
发布RRPC消息	参数输入方式	发送请求 请求耗时: 965ms
设备透传指令控制	表单 JSON 参数推荐	
发布广播消息	ProductId [*] 😔 🛈	响应结果 响应头 真实请求
获取指定网关设备的子设备列表	BO3L64H	(6 +
获取设备绑定的用户列表	DeviceName [*] 😧 🚯	response : 1 "Data": "", "15-2-700 0415 47-5 0504 720200557445"
获取产品的设备列表	devt	<pre>"Result": "{\"Sent\":1, \"pushResult\":0)"</pre>
生成单个设备绑定的签名	Data [*] 😄 🚯	}
直接绑定设备和家庭	{"brightness":20}	查看 1fc3a702-0d15-47ce-9b8d-736382b57d4a 的诊断信息 亿
查询绑定到家庭的网关设备的子设备列表	Method (洗垣) [#] 😧 🚯	9
获取网关绑定的子设备列表	string	7
查询设备绑定的网关设备		5
查看设备详情	DeviceId (远琪) [*] 😧 🕃	Ē
批量删除设备	string	=
删除设备	DataTimestamp (选填) [*] 😔 🛈	
创建设备	integer	

3. 若输入参数输入正确,而且远控的设备已连接物联网开发平台,并成功订阅物模型属性下行 Topic,MQTT.fx 将会收到平 台下发的物模型消息。

WQTT.fx - 5.3.0 - Standard Edition						-	
MQTT.fx Extras Help							
Test	Connect	Disconnect		(МОТТ 3) 🗂	- •	MQ'	T.fx
Publish Subscribe Sparkplug Explorer	Sparkplug Editor (beta)) Scripts Broker Stat	us Log				
\$thing/down/property/BO3L64HQ7C/dev001	Subscribe			Qo5 0	QoS 1 QoS 2	Autoscro	
\$thing/down/property/BO3L64HQ7C/dev001	2 Topic Filter						List Table
Dump Messages (JSON) Mute	Unsubscribe	10:44:42.38682821	\$thing/down/property/BO3L64HQ7C/de	v001			
	2	10:52:18.39138357	\$thing/down/property/BO3L64HQ7C/de	v001			
Topics Collector (0) Scan	Stop 😋 \$thing/dov	wn/property/BO3L64HQ7	C/dev001				
	{"method f2b-64f6	d":"control","clientTok 5-4a2c-b7c3-fd874d9c590	<pre>xen":"v2530232175Cyptk::89d97)4"."params":{"brightness":20</pre>	2			
	0}}		. , , ,	01-09-2023 10:52:18.3	39138357		
				Name	Value		
					No content in	table	
	Content Type	e text/plain	· D · I				
	Payload deco	oded with Plain Te	xt Decoder				


平台转发消息至用户 HTTP 服务

最近更新时间: 2024-11-05 15:02:52

操作场景

物联网解决方案通常需要实时获取设备上报到物联网平台的数据、状态等信息,再结合各自场景的业务数据来完成整套物联网解 决方案的闭环业务流程。本文档主要介绍如何使用平台的规则引擎消息转发至用户自建 HTTP 服务的能力,将设备数据、状态转 发到用户自建的 HTTP 服务,用户可根据此文档中的代码示例快速了解并构建自己的 HTTP 服务。

准备工作

为了快速体验该业务功能,需要提前做好以下准备工作:

- 注册腾讯云账号,申请 物联网开发平台服务。
- 参见 MQTT.fx 快速接入物联网开发平台,能够成功模拟设备连接平台,并能发布物模型消息至平台。
- 用户需提前注册开通 腾讯云 SCF 产品,通过 SCF 产品托管 HTTP 服务,将对应的 HTTP 服务接收地址准备好配置到规则引擎转发服务中。

操作步骤

使用 SCF 部署 HTTP 服务

本文档使用腾讯云云函数(Serverless Cloud Function,SCF)产品快速搭建 Web 服务,来接收物联网平台规则引擎处理 后的数据。本示例只展示如何接收物联网平台的数据,进一步使用数据需要用户根据实际业务场景去处理。

在 SCF 平台基于 Express 快速搭建 Node 服务示例

1. 登录 SCF 控制台,选择 Serveless 应用,单击新建应用。

Serverless	Serverless 应用					
☷ 概览 ② 函数服务	 Web 建站全新体验 无改進 【联合特惠】全景录制,所 	部署,函数直接处理 HTTP 请求,体 ;见即所得的录制模式,高度还原互动	\$验产品写问卷,有机会获得精美礼品! <u>产品文档>>></u>)效果,免后期合成,稳定支持高并发业务需求,更1	<mark>ビ <u>问巻入口>></u> ビ</mark> 有实时音视频、云函数资源包,低至 1 元, <u>立即</u> :	<u> 领取>></u> 【2	
 Serverless 应用 高级能力 	新建应用			请选择您要进行过滤	恩的标签	Q Ø
◇ 层 □ 函数套餐包	应用名称 ^{\$} iot-test	状态 ✓ 正常	标签 http	上次修改时间 \$ 2023-08-23 15:44:26	操作	
拓展能力	共 1 条				10 ▼ 条/页	/1页 ▶ ₩
W'ED争计总线 C						0
						2
						E
□ 给产品打个分 🧿						

2. 选择 Web 应用下的 Express 框架模板,单击下一步。



Serverless	← 新建应用						
₽ 概览							
② 函数服务	创建方式	应用市场	Web	应用			
鎫 Serverless 应用		快速创建开箱即用的 Serverless 应用	通过模	版或导入已有项目,快速部署 Web 应用			
高级能力							
\$ 层	框架选择	请输入名称查询					Q
🛾 函数套餐包							
拓展能力		Next.js 框架 社区模版 查希	详情	Nuxt.js 框架 社区模版	查看详情	Express 框架 社区模版	查看详情
☑ ASW工作流 ☑ ঔ EB事件总线 ☑		NEXT.s		NUXT		expres	s
		基于云函数和 API 网关,快速迁移您的 Next.js 应	用	基于云函数和 API 网关,快速迁移您	的 Nuxt.js 应用	基于云函数和 API 网关,快速迁移您	的 Express 应用
		Koa 框架 社区模版 查希	详情	Egg 框架 社区模版	查看详情	Flask 框架 社区模版	查看详情
		I			۰	► E11	I
三 给产品打个分 ③	下一步取消	ii					

3. 按下图所示输入应用名,选择环境、地域,上传方式。配置结束后单击**完成**。

Serverless	← 新建应用		
書 概览			
② 函数服务	基础配置		
诊 Serverless 应用	应用名	iot	
高级能力		最短2个字符,最长63个字符,只能包含小写字母、数字及分隔符"-"、且必须以小写字母开头,数字或小写字母结尾。	
	环境	开发环境 - dev ▼	
I 函数套餐包		为您的项目选择不同部署环境,实现开发、测试和生产环境的隔离。	
拓展能力	框架	Express 应用 ▼	
CI ASW工作流 II	地域	▶ ▶ ▶	
⊗ EB事件总线 ☑	1. Observed		
	上传万式		9
			<i>(</i> †
	高级配置		, D
	自定义域名 🛈	自用	P
	函数可要		E
	121 27 BC 86		
三 给产品打个分 ③	取消	5.6t	

4. 等待部署完成后进入应用,在 API 网关中可以看到 URL。此 URL 就是您要在规则引擎中填写的 API 地址,假设此 URL 地址为: https://iot-api/ , 那么在本示例中您的 API 地址为 URL 加上接口路由,即: https://iot-api/test 。单击函数名称,进入函数服务。



Serverless	← iot dev *	访问应用 注销应用
日 概览	资源列表 开发部署 部署日志	
② 函数服务	基础信息	
😵 Serverless 应用	应用名称 iot	
	实例名称 http-e> ix	
高级能力	地域 ap-guangzhou (广州)	
\$ 层	API网关	
🗵 函数套缀句	服务ID service-d 0 IZ	
15 四奴去食巴	域名 service-c 0.gz.apigw.tencentcs.com	
拓展能力	环境 release	
	URL III IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	
	云函数	
🖗 EB事件总线 🖸	函数名称	
	命名空间 default	
	运行环境 Nodejs12.16	
	使用层 iot-layer(版本1)	
	内存 512MB	74
	超时时间 3秒	
		P.9
	环境受重 SERVERLESS = 1	4
	401 XX	E
三 给产品打个分 ③		

5. 在函数服务中,可以看到如下图所示的对应的代码模板。单击 app.js,复制示例代码到 app.js 文件中,注意无需全覆盖。

Serverless	← http-ex	1x II%	函数服务帮助文档 IZ
₩ 概览 Ø 函数服务	函数管理	函数管理	版本: \$LATEST v 操作 v
② Serverless 应用 高级能力	別名管理	函数代码 层管理 监控信息 日志查询 提交方法①・在线编辑 ・ 运行环境 Nodejs 12.16	Node.js 12.16 开发教程 IZ 下载 ▼
◇ 层 ○ 函数套餐包	函数 URL bbbcce自	ジ 編輯 选择 查看 转到 ··· ← → ○ http-ex ix ⑦ 资源管理器 ··· JS app.js ×	■ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
拓展能力 プロ ASW工作流 ビ 砂 EB事件总线 ビ	日志查询 并发配额	YHTTP-E: Xx src> Js app.js > > .vscode 1 const express = require('express') > src 2 const path = require('path') > layer 3 const app = express()	Register Register Version Version Version Version Version Version
	部署日志	> node_modules 5 // Routes JS app.js 6 app.get(2^), (req, res) ⇒ { () package-lock.json 7 res.sendFile(path.join(_dirname, 'index.html')) () package.json 10 S sef.bootstran 11	
		<pre>/ serverless.yml 12 app.get('/user', (req, res) => { src.map 13 res.send([yarn.lock 15 title: 'serverless framework', 16 link: '<u>https://serverless.com'</u> 17 } </pre>	
三 给产品打个分 🕥		19 }) 20 2) app.get('/user/tid', (reg. res) ⇒ {	

需要将下述代码复制到 app.js 的代码,请覆盖原 app.js 中的 const 开头的3行代码,原 app.js 中的 app.get 方法无需 覆盖。

```
const express = require('express')
const path = require('path')
const bodyParser = require('body-parser');
const sha1 = require('js-sha1');
const app = express()
app.use(bodyParser.json());
// 鉴权信息校验的代码在上文已经解释过,这里使用Middleware来简化代码
function checkSignatureMiddleware(req, res, next) {
```



```
const token = 'test';
const { signature, timestamp, nonce } = reg.headers;
let tmpStr = [token, timestamp, nonce].sort().join('');
tmpStr = shal(tmpStr);
if (tmpStr === signature) {
    next(); // 身份校验通过,继续执行后续的中间件和路由处理器
  } else {
    res.status(403).send('Forbidden'); // 身份校验失败,返回403 Forbidden 状态码
  }
}
app.get('/test', checkSignatureMiddleware, (req, res) => {
    const { echostr } = req.headers;
    res.set('Content-Type', 'text/plain; charset=utf-8');
    res.send(echostr);
});
// 注意接收规则引擎转发的数据接口为 POST 类型
app.post('/test', checkSignatureMiddleware, (req, res) => {
    const params = req.body.payload.params;
    console.log(params); // { body_temperature; 36 }
    res.end();
});
```

 6. 上述步骤完成后,需要引入两个依赖包,分别是 body-parser 与 js-sha1。在代码编辑器的上方选择终端 > 新建终端,然

 后在底部终端中输入命令:
 cd src 进入 src 目录,接着分别输入命令 npm install body-parser

 npm install js-sha1
 安装两个依赖包。

Serverless	← http-e	x 正常								函数服务帮助文档 🗹
器 概 览	承数管理	函数配置	函数代码 层管理	理 监控信息	111日)	志查询				
⊘ 函数服务	版本管理	提交方法 ⑦・	在线编辑	▼ ji	記行环境 No	dejs 12.16			Node.js 12.16 开发教程 🕻	下载▼
诊 Serverless 应用	别名管理	5 编辑	选择 查看 转到	←	\rightarrow	, <i></i> ⊘ http-ex		x		
高級能力	触发管理	^ء ل	资源管理器	运行		新建物谱	^^`		ī	2部署 🛄 …
◇ 伝 図 図 図 数 套 餐 包	函数 URL		HTTP-EX C	帮助		拆分终端	#/			New States
拓展能力	监控信息 日志香询		∕src >layer			运行任务… 运行生成任务	ሳ ዘ ይ			
🕼 ASW工作流 🖻	并发配额		> node_modules Js app.js		const app.i	显示正在运行的任务				
心 EB事件总线 IZ	部署日志		 ordex.html package-lock.json package.json scf bootstrap 	, 8 9 10 11	// Ro app.g res_ })	重启正在运行的任务 终止任务		index.html'))		•
			! serverless.yml ≌ src.map & yarn.lock		funct cons let tmpS cons if (ne: } el	<pre>配置[[[]]</pre>	tamp, nor lestamp, r ature);) { 继续执行后	, res, next) { ice } = req.headers; ionce].sort().join(''); 续的中间件和路由处理器		
三 给产品打个分 🧿					re: }	.status(403).send('	Forbidder			



← http-express-I x 正常		函数服务帮助文
函数管理 > type-is > unpipe > utils-merge	18 } 19 } 20	
版本管理 > vary {},package-lock.json	<pre>21 app.get('/test', checkSignatureMiddleware, (req, res) => { 22 const { echostr } = req.headers;</pre>	
別名管理 J5 app.js	23 res_set('Content-Tune' 'text/nlain' charset=utf-R')· 问题 输出 调试控制台 终端	+ ~ … ^ ×
触发管理 O package-lock.json	● → http-express-bWIc-iLtkx cd src	∎ D zsh
函数 URL り package Json \$ scf_bootstrap	○ → src npm install body-parser []	I zsh src I zsh src
监控信息 ! serverless.yml		
日志查询		
并发配额 () package.json		
部署日志		
Cloud Studio 2001 目初安装依赖关闭 目	日初部署 天闭 行 1, 列 1 (已选择189) 空格:2 UTF-8	LF {} JavaScript 💭
部署	切换到旧	坂编辑器 使用遇到问题 ②

7. 等待依赖安装完成之后,单击下图红色线框中的部署,即可成功部署 Web 服务,SCF 会自动启动该 Web 服务,约5秒 –
 10秒后系统会提示"部署成功"。



控制台操作

创建规则引擎



- 1. 登录物联网开发平台控制台,选择平台默认开通的公共实例或用户购买的企业实例。
- 2. 进入某个项目,单击左侧菜单基础服务 > 规则引擎,再次单击规则引擎列表页的新建规则。
 - 规则名称: 输入 "HTTP_Forward" 或其他内容。
 - 规则描述: 输入转发某类产品下设备数据的备注信息。

岘则名称★	HTTP_Forward	
	支持英文、数字、下划线的组合,最多不超过32个字符	
观则描述	将智能灯设备数据转发到第三方HTTP服务	
	最多不超过256个字符	

3. 规则基本信息填写完成后,单击确定,即可完成规则的创建。

配置规则引擎

筛选数据

- 1. 在规则引擎列表选择刚创建成功的规则 HTTP_Forward,单击规则名称或操作列的管理,进入规则引擎配置页面。
- 2. 单击下图筛选数据右侧的编辑,选择要转发的设备数据源。

规则引擎 / HTTP_Forward	腾讯云loT技术交流群 🗹	使用指南 🖸
基本信息		编辑
规则名称 HTTP_Forward		
规则状态 已集用		
规则描述 将智能灯设备数据转发到第三方HTTP服务		
		_
筛选数据 ⑦	编辑	SQL调试
字段		
Topic \$(productid)/\$(devicename)/event		
条件 当前SQL SELECT FROM '\$(productid)\\$(devicename)/event'		
行为操作		9
添加行为操作		4
		E
转发错误行为操作		E
添加行为操作		

- 3. 在编辑规则页面,输入或选择如下信息。
 - 字段:输入"*",表示将设备上报的所有 JSON 数据进行转发。
 - Topic:首先选择对应的产品。建议选择"MQTT.fx 快速接入平台"中所创建的产品智能灯;第二步选择设备,请选择
 全部设备,表示规格引擎将转发该产品下所有的设备;最后一步选择转发的数据,请选择物模型属性上报。



段*	*
	仅支持**'、 ','、','、'('、')'、'_'、单引号、 空格、字母和数字,不为空,最多不 超过300个字符
pic *	智能灯
	全部设备 ▼
	物模型属性上报 ▼
	名称命名支持字母、数字、下划线、"("、")"、"\$"、"{"、"}"、","组合;不同层级之间用/分层。+表示一级、使用/+/命名、不能/+aaa/:长度限制为1-64位。
<i>.</i> /+	法值

4. 单击确定按钮,将返回配置规则页面,系统自动生成如下图所示对应的 TOPIC 及筛选数据 SQL。

规则引擎 /	HTTP_Forward	腾讯云IoT技术交流群 🖸	使用指南 🛽
		❷ 编辑成功	×
基本信息			编辑
规则名称	HTTP_Forward		
规则状态	已禁用		
规则描述	将智能灯设备数据转发到第三方HTTP服务		
筛选数据	\odot	编辑	SQL调试
字段			
Topic	\$thing/up/		
条件			
当前SQL	SELECT * FROM '\$thing/up/property/		

配置规则引擎

行为操作配置

1. 在规则引擎列表选择刚创建成功的规则 HTTP_Forward,单击规则名称或操作列的管理链接,进入规则引擎配置页面。



规则引擎 / HTTP_Forward	腾讯云loT技术交流群 🖸	使用指南 🖸
基本信息		编辑
规则名称 HTTP_Forward		
规则状态 已禁用		
规则描述 将智能灯设备数据转发到第三方HTTP服务		
筛选数据 ⑦	编辑	SQL调试
字段 •		
Topic Sthing/up/proper		
条件		
当前SQL SELECT * FROM '\$thing/up/prop		
		6
行为操作		
添加行为操作		4
		FA

2. 下图行为类型选择 "数据转发到第三方服务(Forward)"、选择 API 地址类型选择 "使用已有 HTTP 服务地址"。 HTTP 地址则为 此步骤 中复制 SCF 自动生成的 URL 地址,并在最后加上 /test 。类似

https://service.****.apigw.tencenttcs.com/release/test •

编辑规则	×
 将筛选后的数据转发到第三方服务中。您可使用物联使能部署您的服务,提供服务端模板支持低代码、免备案获取服务地址,转发更快速稳定、资源消耗更低,点击查看文档 	
行为类型	
数据转发到第三方服务 (Forward) ▼	
API地址:★	
https://service-28bbld6q-1259319100.gz.apigw.tencentcs.com/release/tes	
✔ 增加鉴权token	
Token: (i) *	
test	
保存取消	

3. 请勾选 "增加鉴权 Token",且输入 test,该 Token 必须与 SCF 函数服务中 app.js 里定义的 Token 值一致。



4. 单击保存,完成规则行为配置。

5. 返回规则引擎列表,将该规则的状态设置为**启用**。

模拟验证

- 1. 参见 MQTT.fx 快速接入物联网开发平台,模拟设备上报物模型消息。注意上报的数据相关产品需要与规则引擎删选的目标 产品、设备保持一致。若不一致 SCF 日志查询将无法打印输出数据。
- 2. 进入 SCF 的函数服务,单击日志查询,可查看到 SCF 打印的日志,可完整打印出设备上报的物模型属性。

← http-express- (正常) 函数服务帮助文档 2			
函数管理	日志查询		
版本管理 别名管理	① 日志查询功能由購訊云日志服务CLS提供, <u>脑讯云日志服务免费额度</u> Ⅰ 于2022年9月5日0点起调整。如果您暂时不需要日志投递功能,可以在「函数配置」中选择关闭,并前往 CLSF <u>制台</u> Ⅰ 删除不必要的存量函数日志,避免产生费用。	空	
触发管理			
函数 URL	调用日志 高级检索		
监控信息	版本: \$LATEST v 全部日志 v		
日志查询	近15分钟 ▼ 2023-08-31 16:18:35 ~ 2023-08-31 16:33:35 団 刷新	Q	
并发配额			
部署日志	2023-08-31 16:33:32 调用成功 请求ld: 9f740d493d200570a08a451100d621f1 常见错误说明及解决方	滨 	
	2023-08-31 16:33:29 调用成功 运行时间: 2023-08-31 16:33:32 运行时间:1ms 运行内存: 10.806434631347656MB		
	2023-08-31 16:33:25 调用成功 日志: START RequestId: 9f740d493d200570a08a451100d621f1	6	
	2023-08-31 16:33:22 调用成为 《Color: 1, power_switch: 0 》 END RequestId: 977404493d200570a08a451100d621f1 END RequestId: 977404493d200570a08a451100d621f1 Enort Recurstick: 0 * Color: 1, power_switch: 0	C.	
	2023-08-31 16:33:19 调用成为 调用成为	L.	
	2023-08-31 16:33:16 调用成功	T E	
	2023-08-31 16:33:12 调用成功		