

物联网开发平台

应用端开发指南

产品文档



腾讯云

【 版权声明 】

© 2013–2022 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 】



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。

文档目录

应用端开发指南

小程序开发

腾讯连连 H5 自定义开发

概述

快速入门

开发指南

H5 自定义开发 SDK

自主品牌小程序

概述

快速入门

开发指南

小程序 SDK

连连企业接入指南

App 开发

通用版 App

基于 App 开源版

概述

接入指南

部署流程

第三方服务接入指南

App 发布前确认事项

基于 App SDK

概述

接入指南

创建引导

应用端 API

应用端 API 简介与签名介绍

用户管理

微信号注册登录

手机号注册用户

邮箱账号注册用户

随机发送邮箱验证码

随机发送手机短信

手机号、邮箱账号登录

使用手机号重置密码

使用邮箱重置密码

用户注销

修改用户信息

获取用户信息设置

修改用户信息设置

配网管理

生成 Wi-Fi 设备配网 Token

查询配网 Token 状态

用户绑定 Wi-Fi 设备

设备管理

获取产品信息

用户删除设备

获取用户绑定设备列表

获取设备当前状态

修改设备名称

获取设备详情

设备更换房间

添加子设备到用户绑定列表

获取指定网关设备的子设备列表

获取家庭下网关的子设备列表

蓝牙上报数据

蓝牙设备发送设备行为的回复消息

蓝牙设备上报事件

配网绑定设备

设备控制

用户控制设备

同步调用设备行为

异步调用设备行为

设备分享

App 端发送设备分享

获取设备分享 Token

获取设备分享 Token 信息

绑定用户分享的设备

查询用户分享设备列表

删除用户分享的设备

查询设备的用户列表

删除设备分享的用户

家庭管理

创建家庭

删除家庭

修改家庭

获取家庭列表

获取家庭详情

新建房间

修改房间

删除房间

获取房间列表

邀请家庭成员

成员加入家庭

删除家庭成员

成员退出家庭

获取家庭成员列表

App 端邀请家庭成员

设备定时

新建定时任务

修改定时任务

删除定时任务

修改定时任务状态

获取定时任务列表

固件升级

查询设备固件是否升级

确认固件升级任务

查询设备固件升级状态

消息管理

获取消息列表

删除消息

发布 RRPC 消息

手动智能

创建手动智能联动

获取手动智能联动列表

删除手动智能联动

修改手动智能联动

执行手动智能联动

自动智能

创建自动智能联动

获取自动智能联动列表

获取自动智能联动详情

删除自动智能联动

修改自动智能联动

修改自动智能联动状态

数据查询

获取设备物模型数据

获取设备物模型历史数据

获取设备的历史事件

设备聚合数据查询

位置服务

获取设备当前位置

获取设备历史位置

创建设备地理围栏

获取设备围栏告警事件列表

获取设备地理围栏列表

删除设备地理围栏

修改设备地理围栏

修改设备地理围栏启用状态

长连接通信

注册监听

心跳

设备状态推送

文件管理

查询设备指定文件信息

获取下载文件

获取文件上传地址

上传后创建文件

下发文件到设备

音乐服务

歌单歌曲

获取歌曲播放链接

每日推荐

批量查询歌曲信息

推荐歌单

新歌首发

用户登出

用户信息

数据结构

小程序插件使用指南

应用端开发指南

小程序开发

腾讯连连 H5 自定义开发

概述

最近更新时间：2021-05-20 10:10:37

腾讯连连 H5 自定义开发是指设备制造商、方案商，将智能化产品通过“腾讯连连”小程序统一管理控制，只需根据腾讯连连 H5 自定义开发规范进行厂商个性化设备操控面板的交互开发，即可直接使用“腾讯连连”提供的所有能力，从而减少用户开发完整独立应用的开发成本。

微信小程序以免下载、即扫即用降低消费者使用智能化产品的门槛，腾讯云物联网提供“腾讯连连”官方小程序，让消费者实现不同品牌、不同品类、不同通信方式设备的互联互通，满足厂商个性化和共享腾讯连连生态的需求。

快速入门

最近更新时间：2021-04-08 14:09:39

腾讯云物联网开发平台为开发者提供一个 H5 自定义面板 Demo，开发者可以按本文档的指引流程部署并体验 H5 自定义面板 Demo。关于 Demo 的更多信息，请参见 [H5 面板 Demo](#)。

前提条件

注册[腾讯云](#) 账号，并完成 [实名认证](#)。

步骤1：创建项目、产品及设备

本步骤指导您通过物联网开发平台的快速入门功能，创建一个智能灯产品及设备，并运行平台生成的模拟设备 Demo。如需自行创建项目、产品和设备，请参见 [产品开发](#)。

1. 登录腾讯云 [物联网开发平台控制台](#)，选择【公共实例】或您购买的【标准企业实例】进入项目管理页面。
2. 单击页面左上角的【快速入门】，进入快速连接设备页面。
3. 单击【开始使用】进入设备新建界面，按照实际情况填写以下信息：
 - **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。
 - **产品名称**：支持中文、英文、数字、下划线的组合，最多不超过20个字符。
 - **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。

快速连接设备

- 1 新建设备 > 2 下载设备Demo > 3 连接和测试

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

产品名称 *

支持中文、英文、数字、下划线的组合，最多不超过20个字符

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

4. 根据您的电脑的操作系统，单击页面下方 Windows、Linux 或 OSX 操作系统的【设备 Demo 下载】。

设备Demo下载



5. 然后单击【确定下载】，下载设备 Demo 程序到本地。

下载提醒

请确认您的电脑为Win 10 64位操作系统，否则设备Demo可能会运行失败。



6. 单击【下一步】，根据页面指引运行设备 Demo 程序。运行设备 Demo 程序后，页面显示设备已激活，并显示设备上报的数据。单击【完成】。

快速连接设备

- ✓ 新建设备 > ✓ 下载设备Demo > 3 连接和测试

运行设备Demo，设备连接到云端后自动上报数据

Windows系统

在电脑解压设备Demo压缩包后，直接运行quick_start_sample.exe程序。

Linux/OSX系统

第一步：在电脑解压设备Demo压缩包，可以执行下面的命令行：

```
tar -vxvf tencentcloud_iot_device_demo-*.tar.gz
```

[复制](#)

第二步：进入文件，运行设备Demo程序，可以执行下面的命令行：

```
sh ./run.sh
```

[复制](#)

第三步：连接成功后，下面将显示物模型的数据（注：脚本会每隔5秒上报数据，数据为："switch":"0","color":"0","brightness":"1"）

设备通信日志 设备未激活

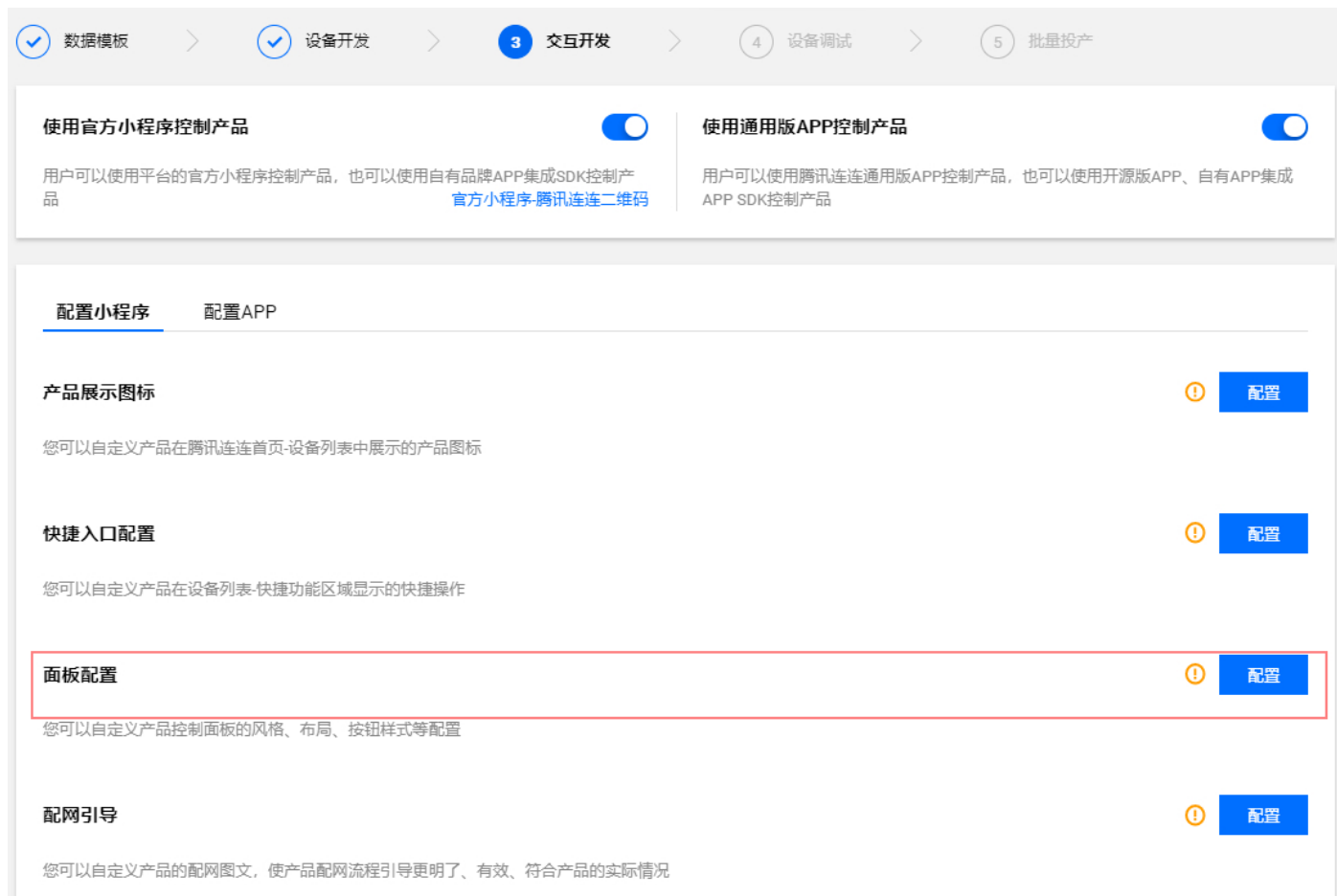
[上一步](#)[完成](#)

7. 页面提示已成功将设备接入腾讯云物联网开发平台，单击【完成】。

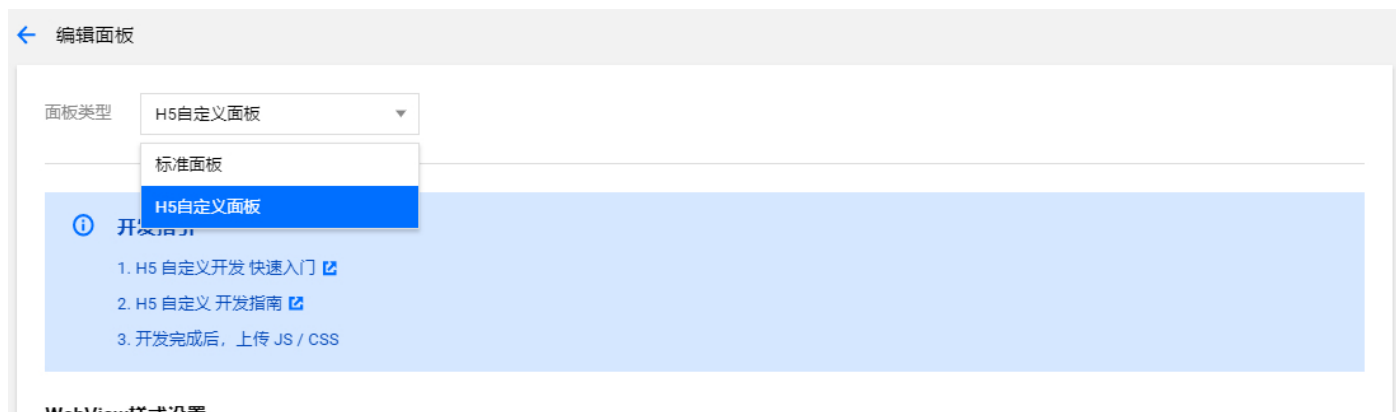
步骤2：配置自定义 H5 面板

1. 下载 H5 面板的 [JS 文件](#) 和 [CSS 文件](#)。
2. 登录腾讯云 [物联网开发平台控制台](#)，选择 [步骤1](#) 中创建的项目，进入项目详情页面。
3. 从产品列表中选择 [步骤1](#) 中创建的产品，进入产品详情页面。

4. 选择【交互开发】>【面板配置】，单击【配置】，进入面板配置页面。



5. 在面板类型中，下拉选择【H5 自定义面板】。



6. 在【上传 JS】表单中，单击【上传】，选择 H5 面板的 JS 文件。上传完成后，页面显示 JS 文件的名称、大小和 MD5 信息。

7. 在【上传 CSS】表单中，单击【上传】，选择 H5 面板的 CSS 文件。上传完成后，页面显示 CSS 文件的名称、大小和 MD5 信息。

上传文件

上传JS (2MB以内)

https://ic

00

上传

上传CSS (2MB以内)

https://iotm:

/100

上传

步骤3：添加连连用户到 H5 面板访问白名单

开发阶段，需要设置访问白名单才可使用腾讯连连调试 H5 自定义面板。设置访问白名单的操作步骤如下：

1. 打开腾讯连连小程序。
2. 单击【我的】，然后单击页面顶部的个人信息，进入个人信息页面。

3. 长按【账号】以复制腾讯连连用户 ID，复制成功后页面提示内容已复制。



4. 登录 [物联网开发平台控制台](#)，选择对应的项目中及产品。

5. 在单击产品中的【交互开发】>【配置小程序】>【面板配置】>【配置】进入面板配置页。

数据模板

设备开发

3 交互开发

4 设备调试

5 批量投产

使用官方小程序控制产品

使用通用版APP控制产品

用户可以使用平台的官方小程序控制产品，也可以使用自有品牌APP集成SDK控制产品

用户可以使用腾讯连连通用版APP控制产品，也可以使用开源版APP、自有APP集成APP SDK控制产品

官方小程序-腾讯连连二维码

产品

配置小程序

配置APP

产品展示图标

配置

您可以自定义产品在腾讯连连首页-设备列表中展示的产品图标

快捷入口配置

配置

您可以自定义产品在设备列表-快捷功能区域显示的快捷操作

面板配置

配置

您可以自定义产品控制面板的风格、布局、按钮样式等配置

配网引导

配置

您可以自定义产品的配网图文，使产品配网流程引导更明了、有效、符合产品的实际情况

扫一扫产品介绍

配置

您可以自定义用户在使用微信扫一扫添加设备时的产品介绍页

智能联动配置

配置

您可以自定义用户在添加智能时，该产品可作为条件或任务的功能属性

6. 将“面板类型”选择为【H5自定义面板】，单击【设置白名单】，进入设置白名单页面。



7. 在白名单列表中，输入上述步骤复制的腾讯连连用户 ID。

设置白名单



只有将腾讯连连账号配置在下方白名单列表才允许在开发阶段通过腾讯连连小程序访问 H5自定义面板

白名单列表

请复制腾讯连连小程序“我的->个人信息”下的账号至白名单列表中，输入回车键分隔每个账号，最多支持配置10个

确定

取消

8. 单击【确定】，页面提示保存成功。

9. 关闭“设置白名单”的弹窗后，单击配置页面的【保存】即可。

步骤4：在小程序绑定设备

虚拟设备

1. 在产品详情页面，选择【设备调试】>【虚拟设备调试】，进入虚拟设备调试页面，页面将展示虚拟设备调试的二维码。



2. 使用微信扫描该二维码，微信即跳转到腾讯连连小程序。授权完成后，腾讯连连小程序自动绑定该设备，页面提示添加设备成功。

真实设备

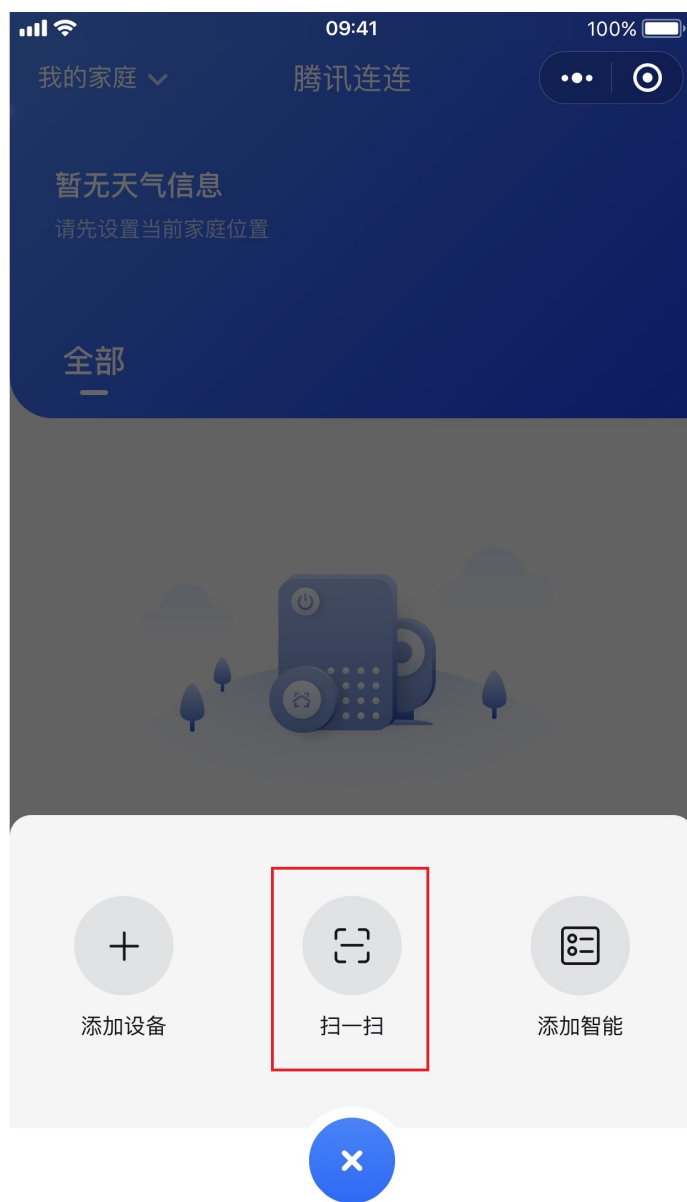
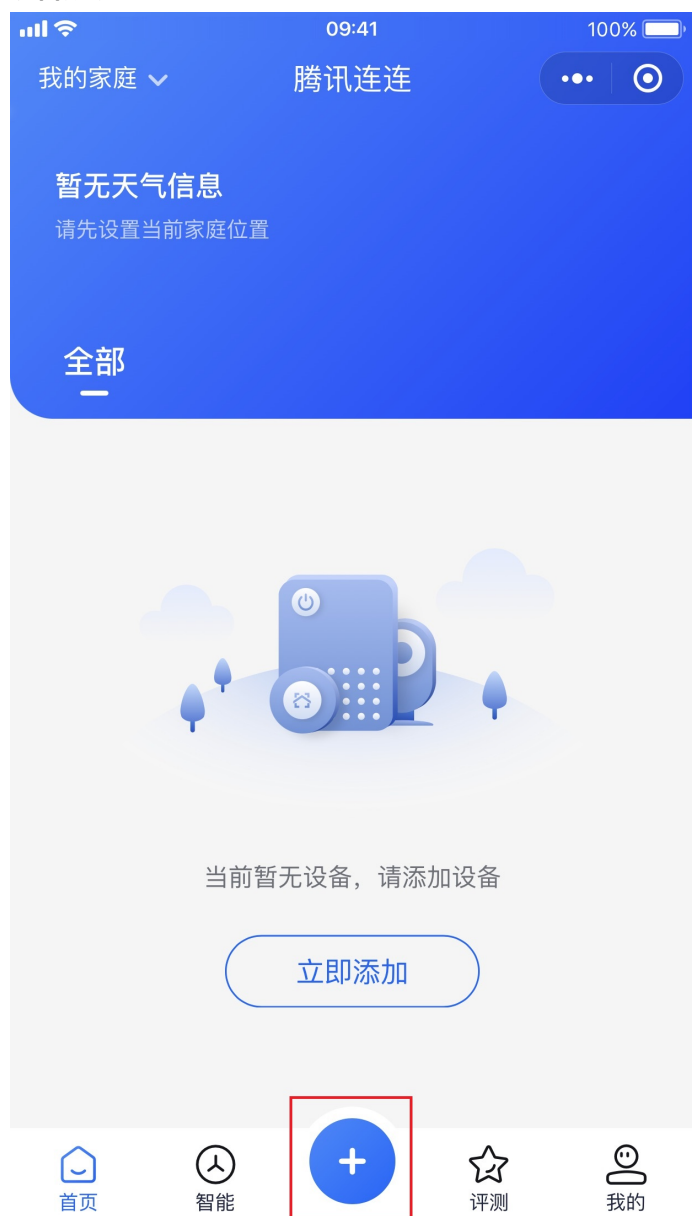
1. 在产品详情页面，单击【设备调试】，进入设备调试页面，页面展示设备列表。
2. 在设备列表中，选择 步骤1 中创建的设备，单击【二维码】，页面展示设备绑定二维码。



3. 使用微信扫描下方的小程序码，打开腾讯连连小程序。



4. 选择【+】>【扫一扫】，扫描控制台页面展示的二维码。扫描成功后，腾讯连连小程序自动绑定该设备，页面提示添加设备成功。



步骤4：体验自定义 H5 面板

在腾讯连连小程序中，选择 [步骤3](#) 中绑定的设备，进入设备控制页面。页面展示自定义 H5 面板，可在该面板进行设备控制等操作。



开发指南

最近更新时间：2021-09-10 14:05:11

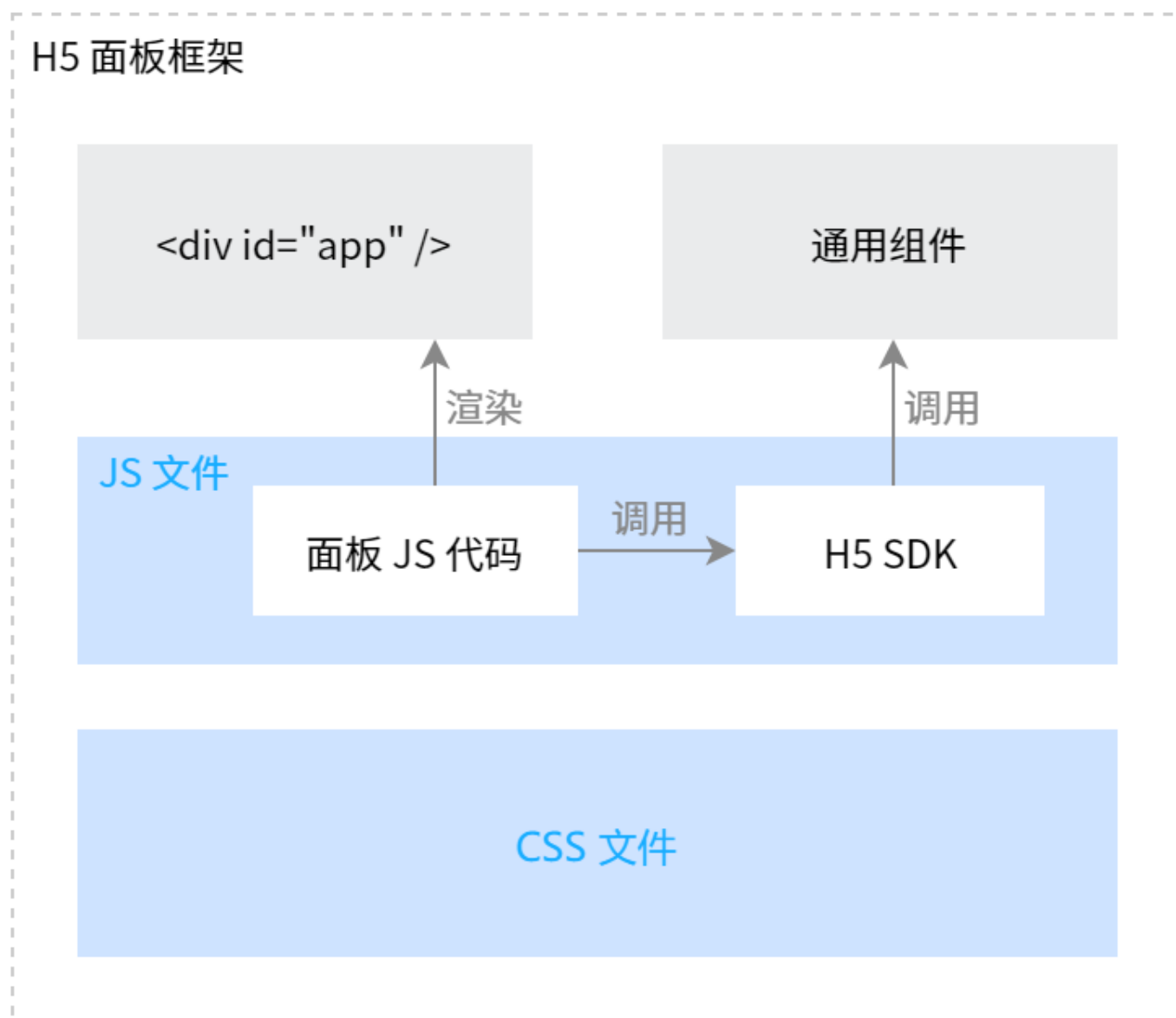
本文介绍开发者如何基于物联网开发平台 H5 SDK（下称 H5 SDK）构建 H5 自定义面板（下称 H5 面板），通过 H5 SDK 使用平台提供的能力。

前提条件

1. 进行 H5 面板开发的运行环境需包含 Node.js 和 npm。
2. 已创建项目、产品及设备，详情请参见 [产品定义](#)。

H5 面板原理

H5 面板框架组成如下：



- <div id="app"> 节点，H5 面板需要渲染到该 DOM 节点下。
- H5 面板的通用组件（例如设备详情页面、模态提示框、顶部提示等），可通过 H5 SDK 调用。
- JS 文件与 CSS 文件由开发者上传。

开发者需要在 JS 文件中实现将 H5 面板渲染到 <div id="app"> 节点的逻辑，可以自行选择使用任何技术方案进行前端渲染，最终仅需输出一个 JS 文件和一个 CSS 文件（CSS 文件可选），提供给 H5 面板加载即可。

通过 H5 SDK 可获取 H5 面板所需的参数、调用 H5 面板的标准组件以及平台提供的能力。

H5 面板开发

H5 面板 Demo

物联网开发平台提供一个 H5 面板 Demo 供开发者参考，H5 面板 Demo 以开源的方式向开发者开放，便于开发者在开发 H5 面板时进行参考。H5 面板 Demo 项目，详情请参见 [iotexplorer-h5-panel-demo](#)。

启动调试

H5 面板开发调试时，需要执行以下命令。

1. 安装依赖（下载 Demo 后执行一次即可）。

```
npm install
```

2. 实时编译 H5 面板（开发过程中需要保持运行）。

```
npm run dev
```

该命令执行时，H5 面板的 JS 文件可通过 <https://127.0.0.1:9000/index.js> 访问，CSS 文件可通过 <https://127.0.0.1:9000/index.css> 访问。当 H5 面板源代码文件更改后，将自动重新编译 JS 文件和 CSS 文件，刷新 H5 面板页面即可加载到更新后的 JS 文件和 CSS 文件。

编译构建

在 H5 面板开发完成后，需要执行以下命令构建 H5 面板的 JS 文件和 CSS 文件。

```
npm run release
```

构建完成后，JS 文件和 CSS 文件将会分别输出到 `dist/index.js` 文件和 `dist/index.css` 文件。开发者需要在 [物联网开发平台控制台](#) 将这些文件上传到产品的交互开发配置项中。上传操作请参见 [上传 H5 面板](#)。

H5 面板调试

使用浏览器或微信开发者工具的 [公众号网页调试](#) 功能访问指定的 URL，可进入 H5 面板的开发模式，对 H5 面板进行调试。

开发模式下，H5 面板框架将分别从以下两个 URL 加载 JS 文件和 CSS 文件。您需要配置代理，使得浏览器或微信开发者工具可通过这两个 URL 加载 H5 面板的 JS 文件和 CSS 文件，从而进行 H5 面板的调试。

- JS 文件 URL：<https://developing.script/developing.js>
- CSS 文件 URL：<https://developing.style/developing.css>

② 说明：

开发模式需要用到您的腾讯云登录态，必须使用对该产品有操作权限的腾讯云账号登录，才可以进入相应产品的 H5 面板开发模式。

本地调试

本地调试将会注册一个与您腾讯云账号关联的腾讯连连用户作为调试用户，并创建一个家庭和一个用于调试的设备，让您无需关注创建家庭、设备绑定等逻辑，只需专注于 H5 面板的开发。

访问以下 URL 可进行 H5 面板的本地调试，请将 ABCDEFG 替换为实际的产品 ID。

```
https://iot.cloud.tencent.com/h5panel/developing?productId=ABCDEFG
```

下面以微信开发者工具进行 H5 面板 Demo 的本地调试为例，描述具体的操作步骤。

步骤1：下载 Demo 项目

1. 拉取 Demo 项目源代码

```
git clone https://github.com/tencentyun/iotexplorer-h5-panel-demo.git
```

2. 安装依赖

```
cd iotexplorer-h5-panel-demo  
npm install
```

步骤2：安装并配置 whistle

1. 安装 whistle，执行如下命令：

```
npm install -g whistle
```

2. 启动 whistle，执行如下命令：

```
w2 start
```

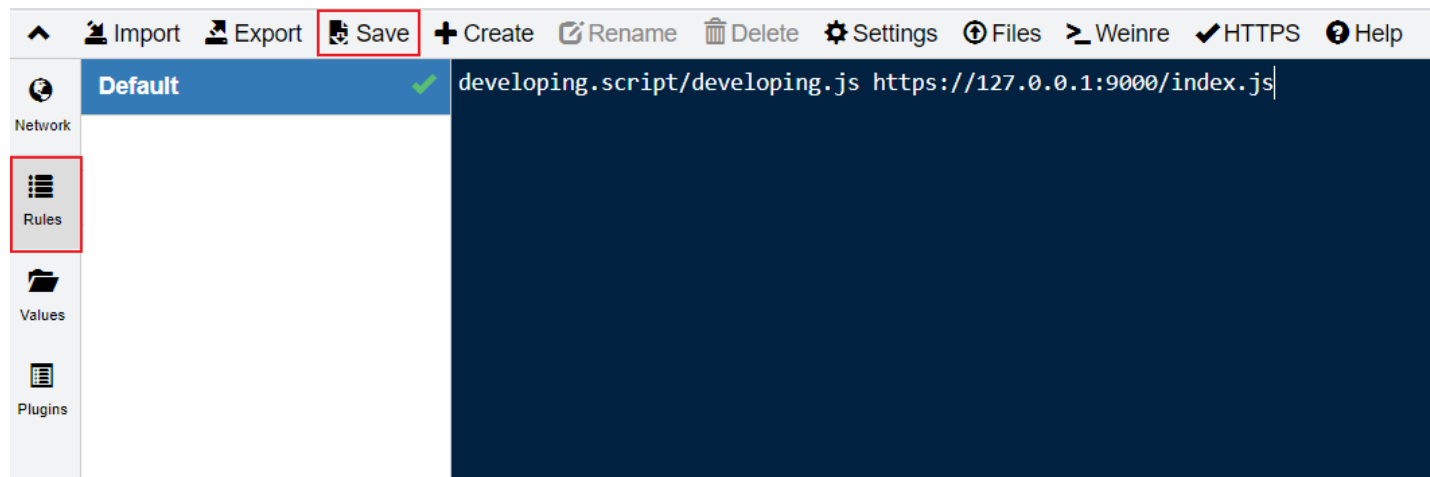
whistle 启动成功后，命令行有如下输出：

```
[i] whistle@2.5.7 started  
[i] 1. use your device to visit the following URL list, gets the IP of the URL you can access:  
http://127.0.0.1:8899/  
... (其后输出省略)
```

3. 通过浏览器打开 `http://127.0.0.1:8899/`，进入 whistle 的用户界面。
4. 选择 **Rules > Default**，进入规则配置页面。在输入框中填入以下代理规则：

```
developing.script/developing.js https://127.0.0.1:9000/index.js  
developing.style/developing.css https://127.0.0.1:9000/index.css
```

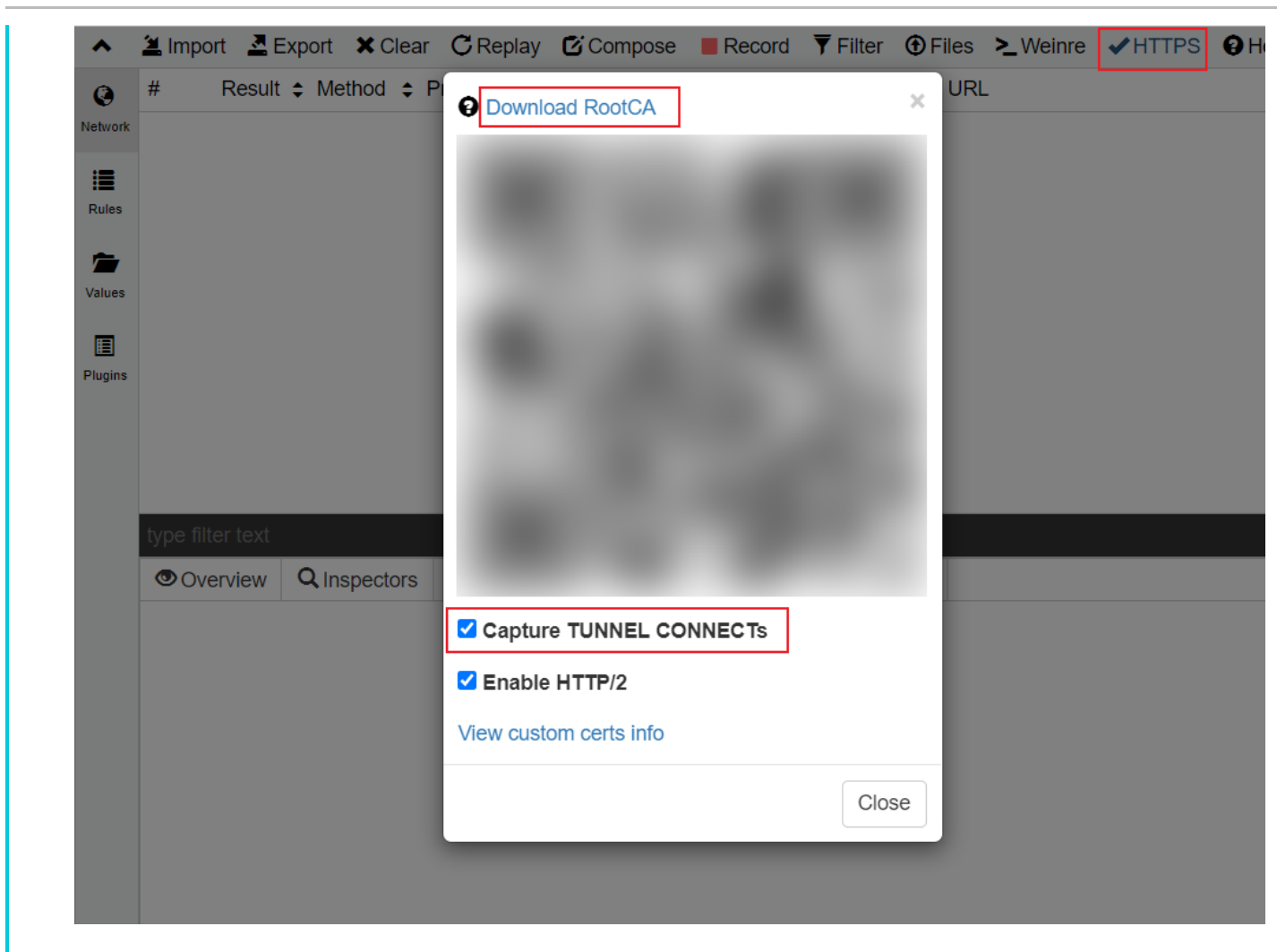
单击**Save**，保存更改内容。



5. 单击菜单栏的**HTTPS**，进入 HTTPS 配置页面。勾选**Capture TUNNEL CONNECTs**。
6. 单击**Download RootCA**，下载根证书到本地。

注意：

此步骤无需关注二维码信息，单击**Download RootCA**并勾选**Capture TUNNEL CONNECTs**即可。



7. 安装根证书，具体步骤请参见 [安装根证书](#)。

步骤3：启动 Demo 项目的实时编译

在 Demo 项目目录下，执行如下命令：

```
npm run dev
```

编译成功后命令行有如下输出：

```
i [wds]: Project is running at https://localhost:9000/
... (中间输出省略)
i [wdm]: Compiled successfully.
```

步骤4：配置微信开发者工具

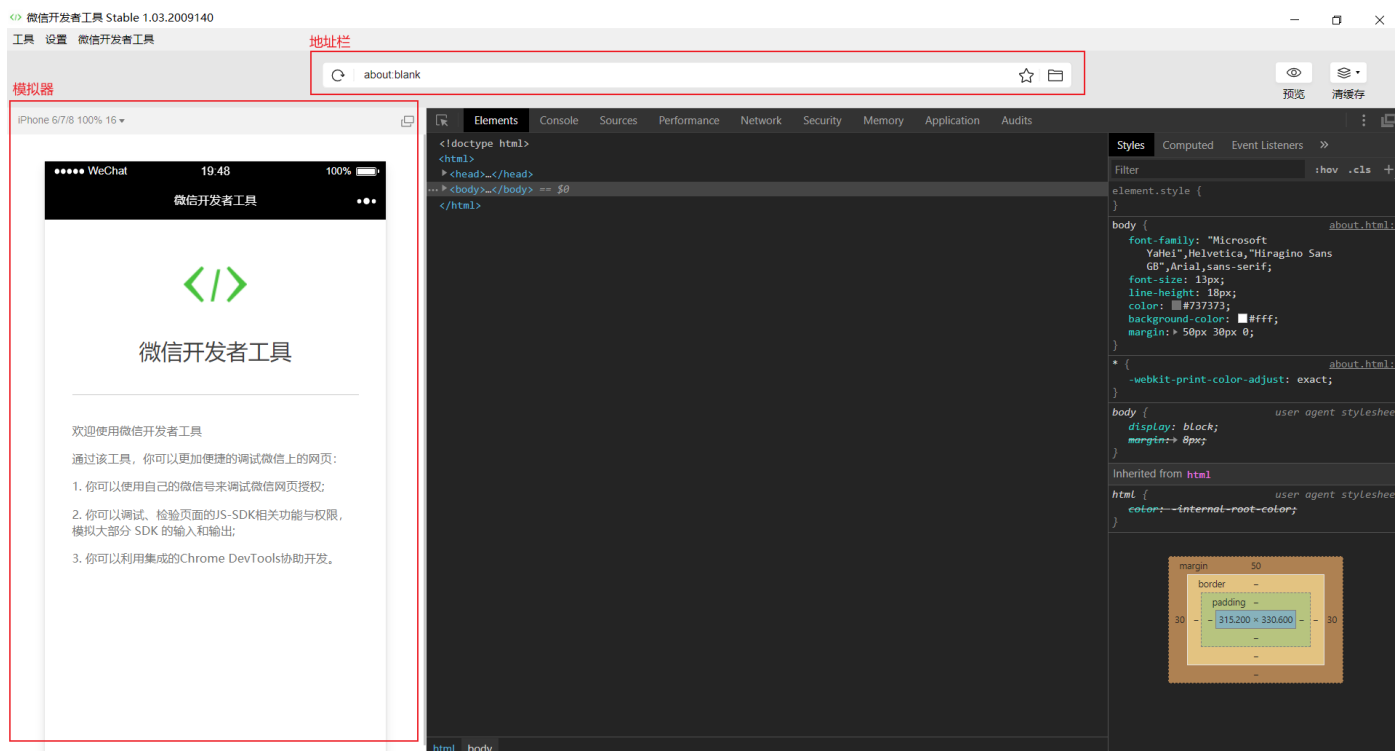
1. 打开微信开发者工具，选择菜单栏的设置 > 代理 > 代理设置。

2. 选择手动设置代理，代理地址填写 127.0.0.1，端口填写 8899。



步骤5：微信开发者工具打开 H5 面板

1. 打开微信开发者工具，在新建项目页面，点击公众号网页，进入公众号调试页面。



2. 验证 H5 面板 JS 文件是否可以访问。

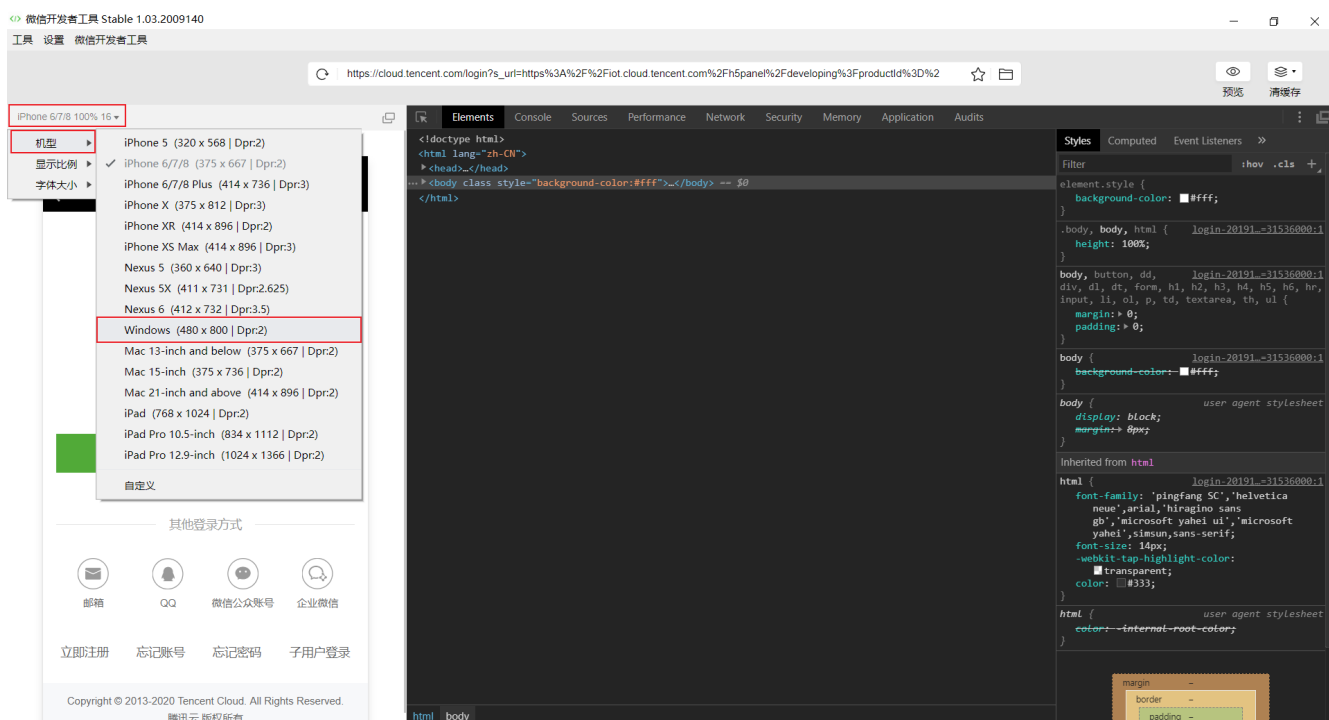
在地址栏输入 <https://developing.script/developing.js>，按回车键。若模拟器中可加载并显示 JS 文件代码，则表明 H5 面板 JS 文件可以访问。

3. 在地址栏输入 本地调试 URL，按回车键，模拟器将跳转到腾讯云控制台的登录页面。

4. 在模拟器菜单中选择机型 > *Windows (480×800 | Dpr:2) *。

说明：

若模拟器模拟的是手机环境，则不能正常进行微信或 QQ 登录，此处先更改为 Windows 机型，登录完成后可改回原来的机型设定。



5. 在模拟器页面中登录腾讯云控制台，登录成功后，模拟器将跳转至 H5 面板，可进行 H5 面板调试。

真实设备调试

通过真实设备调试，您可以对指定的设备进行 H5 面板调试。请将 ABCDEFG 替换为实际的产品 ID，abcdefg 替换为实际的设备名称。如需调试物联网开发平台提供的虚拟设备，请将 abcdefg 替换为 ~virtualDev。

```
https://iot.cloud.tencent.com/h5panel/developing?productId=ABCDEFG&deviceName=abcdefg
```

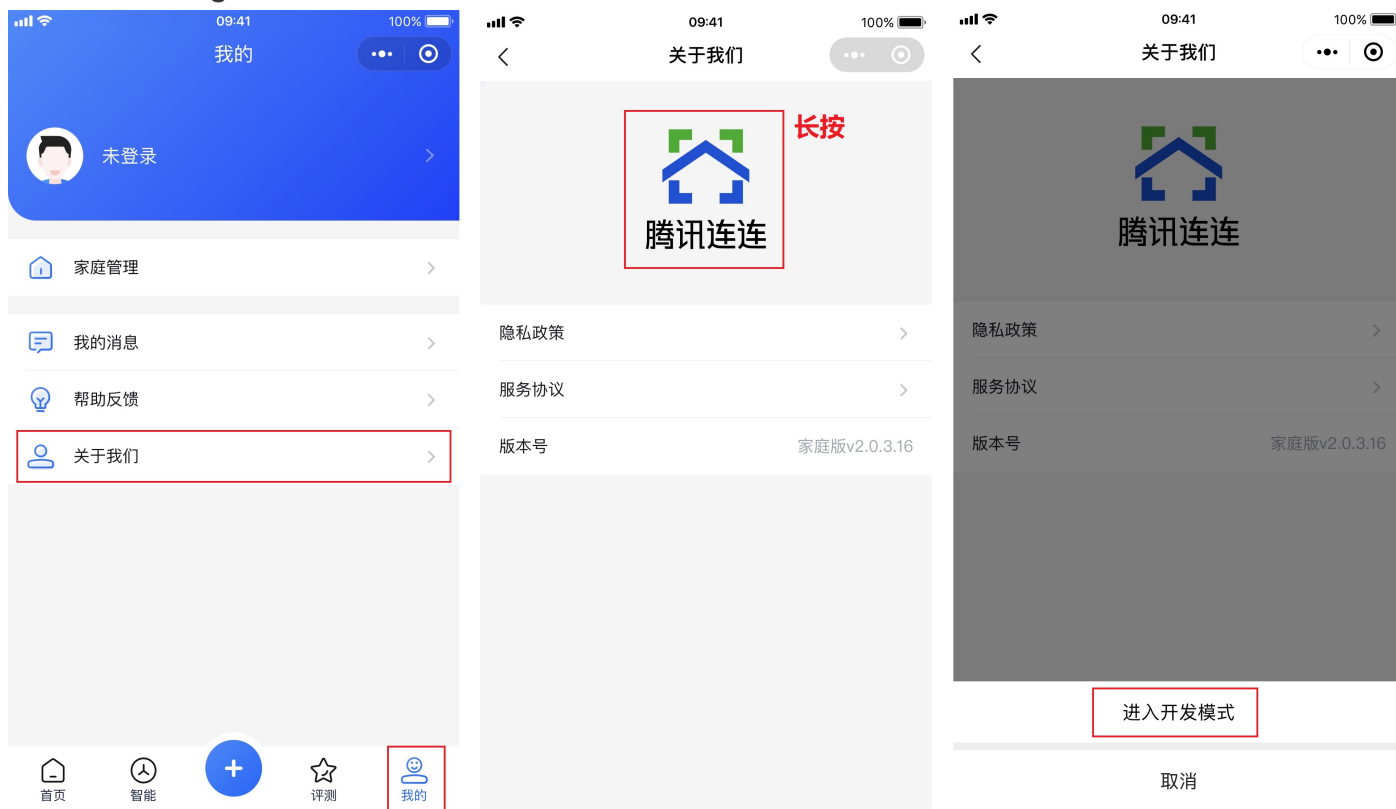
注意：

- 对指定设备进行调试时，会自动将该设备绑定到您的调试用户的家庭下，同时会解除该设备原来的绑定关系。
- “已发布”状态的产品不支持真实设备调试，因为存在将真实用户正在使用的设备解除绑定关系的风险。

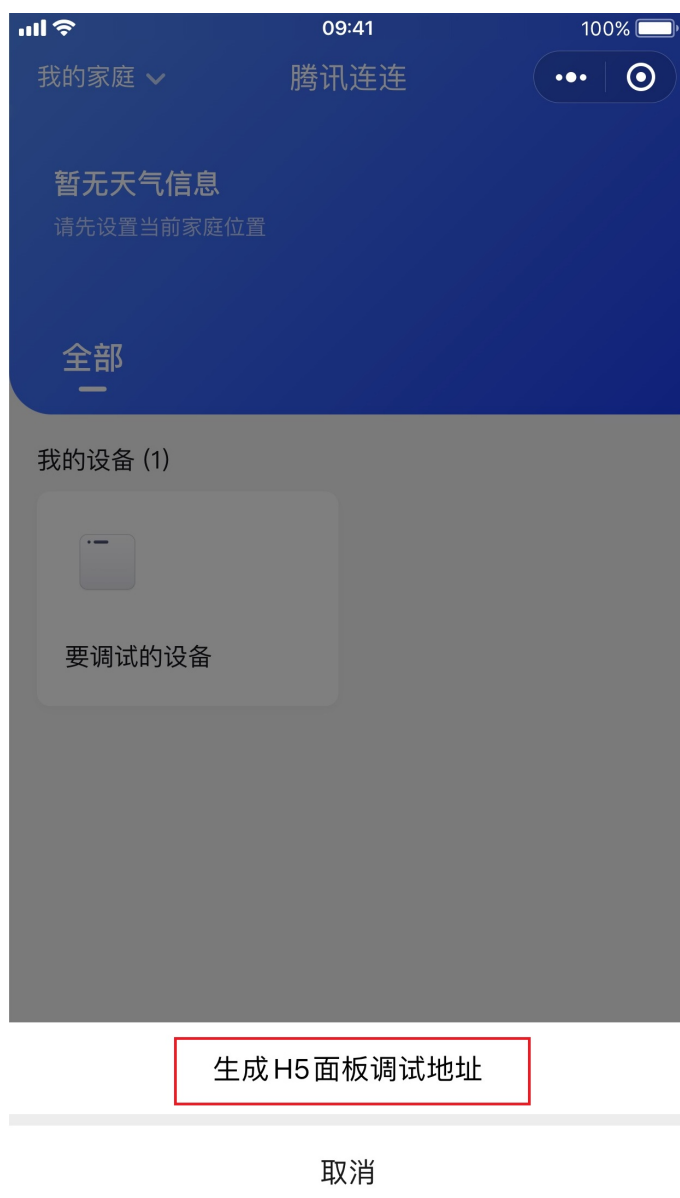
真机调试

对于已发布的产品，不开放真实设备调试，且在小程序中调试 H5 面板又非常困难，为了解决这一问题，平台提供真机调试方案。真机调试可以在保持设备原有绑定关系的情况下，对 H5 面板进行调试。在开始之前，需要先使用腾讯连连小程序，将待调试的设备进行绑定。具体操作步骤如下。

1. 打开腾讯连连小程序，选择**我的** > **关于我们**，进入关于我们页面。
2. 长按腾讯连连 Logo，在弹出菜单中选择**进入开发模式**。页面提示进入开发模式成功。



3. 返回腾讯连连小程序首页，在设备列表中，长按需要调试的设备，在弹出菜单中选择**生成H5面板调试地址**，之后该调试地址将被复制到剪贴板。



4. 通过浏览器访问该调试地址，可以进行 H5 面板真机调试。

H5 面板访问白名单

开发阶段，需要设置访问白名单才可使用腾讯连连调试 H5 自定义面板。设置访问白名单的操作步骤，详情可参见 [快速入门](#)。

调用 H5 SDK 能力

引入 H5 SDK

直接通过 Window 访问。

```
window.h5PanelSdk;
```


或 webpack 配置 externals。

```
// webpack.config.js
module.exports = {
  externals: {
    'qcloud-iotexplorer-h5-panel-sdk': 'h5PanelSdk',
  },
};
```

获取基本参数

H5 SDK 提供产品信息、设备数据、用户信息与家庭信息等基本参数供 H5 面板使用，可通过 `sdk.属性名` 取得。具体的参数列表，请参见 [基本参数](#)。

```
console.log(sdk.deviceId);
```

调用应用端 API

关于应用端 API 的信息，请参见 [应用端 API 简介](#)。

H5 SDK 对应用端 API 的调用过程已进行封装，发送请求时将会自动带上公共参数 `AccessToken` 与 `RequestId`。以下示例代码以调用 [获取设备当前状态](#) 应用端 API 为例。

```
const action = 'AppGetDeviceStatuses'; // 请求应用端 API 的 Action 名
const params = { // 请求应用端 API 的参数
  DeviceIds: [sdk.deviceId]
};
sdk.requestTokenApi(action, params).then(data => {
  // 请求成功
  console.log(data);
}).catch(err => {
  // 请求失败
  console.error(err);
});
```

控制设备

调用 H5 SDK 的 [控制设备属性](#) 接口可对指定设备发起控制操作。

```
// 指定要控制的设备的属性数据
const deviceData = {
```

```
light_switch: 0
};

// 控制 H5 面板当前设备
sdk.controlDeviceData(deviceData).then(data => {
// 请求成功
console.log(data);
}).catch(err => {
// 请求失败
console.error(err);
});

// 控制指定设备
const deviceId = '要控制的设备 ID';
sdk.controlDeviceData(deviceData, deviceId).then(data => {
// 请求成功
console.log(data);
}).catch(err => {
// 请求失败
console.error(err);
});
```

固件升级

SDK 封装了检查固件更新的接口，以及一个默认样式的固件升级提示。开发者通过调用 H5 SDK 提供的接口，可以检查设备当前是否有可升级固件，以及进行固件升级操作。

检查固件升级

调用 H5 SDK 的 [检查设备是否有可升级固件](#) 接口，可以检查指定设备是否有可升级固件。开发者可以直接使用 H5 SDK 提供的默认样式固件升级提示，也可以利用该接口的返回值来实现自定义的固件升级提示，详情如下。

- 方式一：使用 H5 SDK 提供的默认样式固件升级提示

执行以下命令调用 `sdk.checkFirmwareUpgrade` 。

```
sdk.checkFirmwareUpgrade();
```

若当前设备有可升级固件，且设备在线，则弹出默认样式的固件升级提示。

发现新固件

发现设备有新版本(3.3.0)，是否升级？

取消

立即升级

• 方式二：使用自定义的固件升级提示

调用 `sdk.checkFirmwareUpgrade` 时传入 `silent` 参数值为 `true`，则不会弹出上述的默认样式固件升级提示。开发者可根据接口返回的固件升级信息，展示自定义的固件升级提示。其中 `CurrentVersion` 为设备当前固件版本，`DstVersion` 为固件可升级版本。

```
sdk.checkFirmwareUpgrade({
  silent: true // silent 参数为 true 则不弹出 H5 SDK 的默认固件升级提示
}).then(({ CurrentVersion, DstVersion }) => {
  if (DstVersion && CurrentVersion !== DstVersion) {
    // 开发者需要在此处实现自定义的固件升级提示的展示
  }
});
```

进行固件升级

调用 H5 SDK 的 [进行固件升级](#) 接口，可对指定设备进行固件升级。调用该接口后，会跳转至小程序的固件升级页面，进行固件升级操作。

```
// 对当前设备进行固件升级
sdk.goFirmwareUpgradePage();

// 对指定设备进行固件升级
sdk.goFirmwareUpgradePage({
  deviceId: '要进行固件升级的设备 ID'
});
```

长连接通信

H5 SDK 初始化时默认自动连接 WebSocket 服务端，并订阅当前设备的上报数据以及状态信息。

```
// 监听设备上报数据推送
sdk.on('wsReport', ({ deviceId, deviceData }) => {
```

```
console.log('websocket device report', deviceId, deviceData);
});

// 监听设备在线状态变更推送
sdk.on('wsStatusChange', ({ deviceId, deviceStatus }) => {
  console.log('websocket device status change', deviceId, deviceStatus);
});

// 监听设备控制推送
sdk.on('wsControl', ({ deviceId, deviceData }) => {
  console.log('websocket device control', deviceId, deviceData);
});
```

设备详情

H5 SDK 为开发者提供两种设备详情页面，开发者可以根据需要选择使用。

- 标准设备详情：跳转到小程序标准面板的设备详情页面。
- H5 自定义设备详情：在 H5 面板内展示设备详情页面，可在标准设备详情的基础上增加自定义的菜单和按钮。

调用标准设备详情

执行以下代码后，将跳转到小程序标准面板的设备详情页面。

```
sdk.goDeviceDetailPage();
```

调用 H5 自定义设备详情

执行以下代码后，将在 H5 面板内展示设备详情页面。关于 H5 自定义设备详情的接口参数描述，详见 [展示 H5 自定义设备详情视图](#)。

```
sdk.showDeviceDetail({
  labelWidth: 120, // 设备详情的label宽度
  marginTop: 8, // 设备详情的上间距
  shareParams: { // 设备分享时设置自定义分享参数
    a: 'a',
    b: 'b',
  },
  extendItems: [
    {
      labelIcon: 'https://main.qcloudimg.com/raw/be1d876d55ec2479d384e17c94****e69.svg',
      label: '自定义菜单',
    }
  ]
});
```

```
content: '自定义菜单内容（可选）',
onClick: () => console.log('点击自定义菜单'),
},
],
extendButtons: [
{
text: '自定义按钮',
type: 'warning',
onClick: () => console.log('点击自定义按钮'),
},
{
text: '自定义按钮2',
type: 'primary',
onClick: () => console.log('点击自定义按钮2'),
},
{
text: '获取自定义分享参数',
onClick: async () => {
const shareParams = await sdk.getShareParams();
console.log('自定义分享参数: ', shareParams);
}
},
{
text: '关闭设备详情',
type: '',
onClick: () => sdk.hideDeviceDetail(),
},
],
});
```

子设备使用网关面板

平台支持设定网关的面板配置对所有子产品生效，开启后，网关下的子产品将使用网关的面板配置。配置步骤如下：

1. 登录腾讯云 [物联网开发平台控制台](#)，选择项目与网关产品，进入网关产品的产品详情页面。

2. 选择交互开发 > 面板类型，单击配置，进入面板配置页面。

产品开发 / 智能灯

1 数据模板 > 2 设备开发 > 3 交互开发 > 4 设备调试 > 5 批量投产

使用官方小程序控制产品

使用通用版APP控制产品

用户可以使用平台的官方小程序控制产品，也可以使用自有品牌APP集成SDK控制产品

官方小程序-腾讯连连二维码

用户可以使用腾讯连连通用版APP控制产品，也可以使用开源版APP、自有APP集成APP SDK控制产品

配置小程序

配置APP

产品展示图标

您可以自定义产品在腾讯连连首页-设备列表中展示的产品图标

配置

快速入口配置

您可以自定义产品在设备列表-快捷功能区域显示的快捷操作

配置

面板配置

您可以自定义产品控制面板的风格、布局、按钮样式等配置

配置

配网引导

您可以自定义产品的配网图文，使产品配网流程引导更明了、有效、符合产品的实际情况

配置

扫一扫产品介绍

您可以自定义用户在使用微信扫一扫添加设备时的产品介绍页

配置

3. 在面板类型中选择自定义 H5 面板。

产品开发 / 智能灯

1 数据模板 > 2 设备开发 > 3 交互开发 > 4 设备调试 > 5 批量投产

编辑面板

面板类型

H5自定义面板

标准面板

H5自定义面板

开发指引

1. H5 自定义开发 快速入门

2. H5 自定义 开发指南

3. 开发完成后，上传 JS / CSS

WebView样式设置

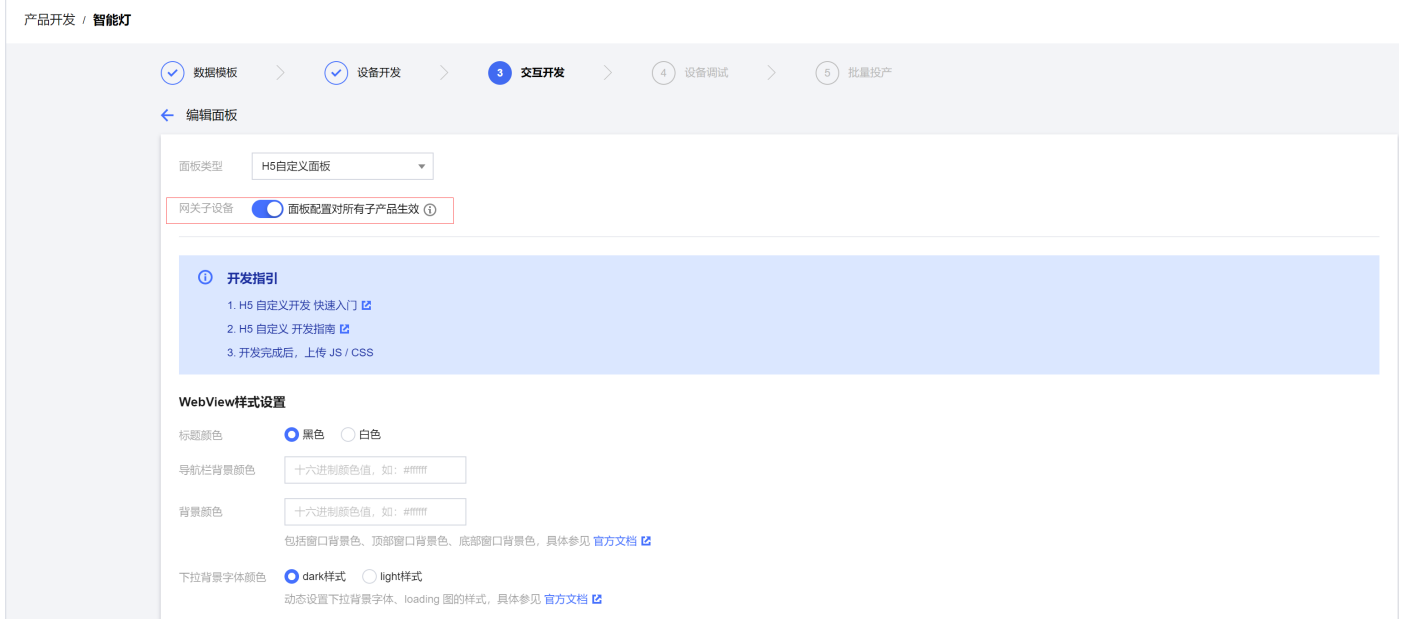
标题颜色

导航栏背景颜色

背景颜色

下拉背景字体颜色

4. 开启网关子设备中面板配置对所有子产品生效的开关。



监听页面显示、隐藏事件

H5 SDK 提供页面显示、隐藏事件回调。当 H5 页面或小程序切换到前台或后台时，会触发相应的事件。开发者可以通过监听这些事件，在对应事件触发时执行自己的业务逻辑。

```
// 监听 H5 页面切前台
sdk.on('pageShow', () => {
  console.log('on pageShow');
});

// 监听 H5 页面切后台
sdk.on('pageHide', () => {
  console.log('on pageHide');
});

// 监听小程序切前台
sdk.on('appShow', () => {
  console.log('on appShow');
});

// 监听小程序切后台
sdk.on('appHide', () => {
  console.log('on appHide');
});

// H5 页面切前台时，刷新页面标题的设备名称
```

```
sdk.on('pageShow', async () => {  
  const deviceInfo = await sdk.getDeviceInfo();  
  
  // 设备展示名称  
  const deviceDisplayName = deviceInfo.AliasName || sdk.productInfo.Name;  
  
  // 更新页面标题  
  window.document.title = deviceDisplayName;  
});
```

蓝牙设备通信

H5 面板内无法直接调用小程序的蓝牙接口，H5 SDK 中已封装 H5 面板与小程序间的蓝牙协议通信机制。要在 H5 面板使用蓝牙通信能力，必须使用 H5 SDK 提供的蓝牙模块。关于蓝牙设备的 H5 面板开发，请参见 [蓝牙设备文档](#)。

上传 H5 面板

在 H5 面板开发完成以后，需要在控制台上传 H5 面板的 JS 文件和 CSS 文件，具体操作如下：

1. 登录腾讯云 [物联网开发平台控制台](#)，选择项目与产品，进入产品详情页面。
2. 选择**交互开发** > **面板类型**，单击**配置**，进入面板配置页面。

数据模板

设备开发

交互开发

设备调试

批量投产

使用官方小程序控制产品

使用通用版APP控制产品

用户可以使用平台的官方小程序控制产品，也可以使用自有品牌APP集成SDK控制产品

官方小程序-腾讯连连二维码

用户可以使用腾讯连连通用版APP控制产品，也可以使用开源版APP、自有APP集成APP SDK控制产品

配置小程序

配置APP

产品展示图标

①

配置

您可以自定义产品在腾讯连连首页-设备列表中展示的产品图标

快捷入口配置

①

配置

您可以自定义产品在设备列表-快捷功能区域显示的快捷操作

面板配置

①

配置

您可以自定义产品控制面板的风格、布局、按钮样式等配置

配网引导

①

配置

您可以自定义产品的配网图文，使产品配网流程引导更明了、有效、符合产品的实际情况

扫一扫产品介绍

①

配置

您可以自定义用户在使用微信扫一扫添加设备时的产品介绍页

智能联动配置

①

配置

您可以自定义用户在添加智能时，该产品可作为条件或任务的功能属性

3. 在面板类型中选择自定义 H5 面板。



4. 在上传 JS 表单中，单击上传，选择 H5 面板的 JS 文件。上传完成后，页面显示 JS 文件的名称、大小和 MD5 信息。

上传文件

上传JS (2MB以内)

上传

文件名: index.js 文件大小: 1.74MB md5: b583bd241193aac025cc23d02cd34c5f

上传CSS (2MB以内)

上传

5. 如果需要上传 CSS 文件，在上传 CSS 表单中，单击上传，选择 H5 面板的 CSS 文件。上传完成后，页面显示 CSS 文件的名称、大小和 MD5 信息。

6. 单击保存，页面提示保存成功，上传的 JS 文件与 CSS 文件保存生效。

蓝牙搜索 H5 页面

由于自定义蓝牙协议设备的蓝牙搜索以及协议解析是在 H5 容器内进行运行，所以在小程序的搜索区块上无法搜索到对应的蓝牙设备，需要进入对应产品的蓝牙搜索 H5 页面进行搜索。



前提条件

通信方式为“BLE”，并且设备开发方式为“基于自定义蓝牙协议开发”的产品，可以使用蓝牙搜索 H5 页面。具体配置步骤如下：

1. 登录 [物联网开发平台控制台](#)，创建“通信方式”为BLE的产品，详情请参见 [创建项目及产品](#)。

⚠ 注意：

已创建的产品不能更改“通信方式”如现有产品选择的“通信方式”不是BLE，请重新创建产品。

2. 打开创建成功的BLE产品，单击**设备开发**，将“设备开发方式”选择为**基于自定义蓝牙协议开发** > **下一步**后，控制台右上角将提示“保存成功”即可完成配置。

产品开发 / BLE产品

1 数据模板

2 设备开发

3 交互开发

4 设备调试

5 批量投产

请选择设备开发方式

基于标准蓝牙协议开发（默认）
使用官方标准蓝牙协议（LLSync 协议）接入设备，请查看文档了解如何基于标准蓝牙协议开发 [LLSync SDK 下载](#) [开发指南](#) ▼

基于自定义蓝牙协议开发
使用自定义蓝牙协议接入设备，请查看文档了解如何基于用户自定义蓝牙协议开发 [自定义 H5 面板 Demo 下载](#) [开发指南](#) ▼

上一步

下一步

3. 按照 [上传 H5 面板](#) 中的步骤，上传 H5 页面的 JS 文件与 CSS 文件。

页面入口

蓝牙搜索 H5 页面入口分为“产品未发布”和“产品已发布”两种情况，具体说明如下。

产品未发布

方式一

1. 登录 [控制台](#) 选择对应的项目，单击**产品开发**进入产品列表页面。
2. 选择产品状态为“开发中”的某一产品进入产品详情页，单击**交互开发** > “扫一扫产品介绍”右侧的**配置**进入配置页面。

3. 上传“产品实体图”、“产品备注”等相关信息，单击**保存**后，您可扫描页面右下角提供的二维码进行体验。

产品名称 ⓘ
产品名称，20字符以内

产品实体图  [选择图片](#)
请上传大小500KB以内，JPG 或 PNG 格式的图片，图片尺寸建议为300px*300px

产品备注 请输入产品备注，15字符以内
产品备注，15字符以内

保存



点击保存后可使用微信扫码体验



方式二

1. 如 [真机调试](#) 阐述，进入腾讯连连开发模式。

2. 单击+ > **添加设备**，长按页面上方的“扫描附近的蓝牙设备”模块，在弹出框内“输入前往 H5 蓝牙搜索页的产品

ID”。



产品已发布

方式一

1. 登录 [控制台](#) 选择对应的项目，单击**产品开发**进入产品列表页面。

2. 选择产品状态为“已发布”的某一产品进入产品详情页，单击**批量投产**，您即可扫描页面下方提供的二维码进行体验。

1. 产品确认

2. 申请发布

3. 量产管理

产品名称



状态

已发布

产品品类

智能城市-公共事业-路灯照明

通信方式

其它

数据协议

数据模板

认证方式

密钥认证

创建时间

2020-08-28 09:21:12

扫一扫二维码

**⚠ 注意：**

如果扫码打开的不是自定义蓝牙搜索页，而是添加设备的页面，请检查是否已经按照 [前提条件](#) 对产品进行配置。

方式二

您可 [提交申请](#) 并通过腾讯连连测试认证流程，成为推荐产品后；产品将会在添加设备 > [产品分类列表](#) 中展示供您选择。

电工照明

推荐



遥控大师空调伴侣



华来小方智能彩灯带STP01

安防报警

家用电器

网络设备

厨房电器

运动健康

影音办公



窗帘



五路排插



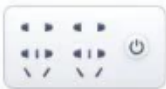
灯



插座



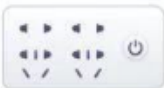
开关面板



插排



五路灯



排插

极速模式

H5 加载时，需要在服务端拉取面板所需要的所有数据，导致加载速度较慢。为了解决这个问题，我们推出了极速模式，其原理是加载时服务端不去拉取数据，而是由 H5 打开后由小程序客户端将本地的数据通过我们特有的链路推送到 H5，可以大大加速加载时间，但是由于从原本的直出时同步准备好所有数据改成了页面渲染后异步加载数据，所以渲染时依赖直出数据的面板需要做一定的适配才能够正常使用。

适配方法

仅需要渲染页面之前先等待一个 sdkReady 的 Promise 即可，具体实现可参考以下 React 实现的示例：

```
function Page() {  
  const [sdkReady, setSdkReady] = useState(false);  
  useEffect(() => {  
    h5PanelSdk.sdkReady().then(() => setSdkReady(true));  
  }, []);  
  
  return !sdkReady ? <div> loading...</div> : (  
    <div>  
      /* 正常渲染您的页面 */  
    </div>  
  );  
}
```

开启说明

由于极速模式是针对产品级生效，存量产品请务必全量发布适配版本后，再开启极速模式，否则可能会影响到老版本用户的正常使用。

H5 自定义开发 SDK

最近更新时间：2021-12-09 16:06:29

基本参数

H5 SDK 提供产品信息、设备数据、用户信息与家庭信息等基本参数供 H5 面板使用，可通过 `sdk.属性名` 取得。

属性名	类型	描述
deviceId	string	设备 ID，由 {productId}/{deviceName} 组成
productId	string	产品 ID
deviceName	string	设备名称
deviceInfo	DeviceList ShareDevices	设备信息，数据结构同 DeviceList ；如果是分享设备，则数据结构同 ShareDevices
roomList	RoomList []	当前家庭的房间列表
roomName	string	当前设备的房间名称
dataTemplate	string	设备所在产品的物模型，与【数据模板】>【查看JSON】中展示的物模型定义一致
deviceStatus	number	设备在线状态，在线：1，非在线：0
deviceDisplayName	string	设备展示名称，会依次取：AliasName > productInfo; name > deviceName 来展示
isShareDevice	boolean	是否是分享设备
familyId	string	设备所在家庭 ID，如果是分享设备则无此值
roomId	string	设备所在房间 ID，如果是分享设备则无此值
familyInfo	FamilyList	设备所在家庭详情，如果是分享设备则无此值
isFamilyOwner	boolean	用户是否是当前家庭的所有者
userInfo	object	用户信息 - Avatar: string; 头像 URL - CountryCode: string; 国家代码 - Email: string; 邮箱地址 - NickName: string; 昵称 - PhoneNumber: string; 手机号码 - UserID: string; 用户 ID

设备管理

获取产品信息

• 接口定义

```
sdk.getProductInfo({ productId?: string }) => Promise<{  
  ProductId: string,  
  Name: string,  
  Description: string,  
  DataTemplate: string,  
  NetType: string,  
  CategoryId: number,  
  ProductType: number,  
  UpdateTime: number,  
}>
```

• 参数说明

参数名	参数描述	类型	必填
productId	可选，不传则使用当前设备的产品 ID	string	否

• 返回值

返回一个 Promise，输出参数如下。

参数名	参数描述	类型
ProductId	产品 ID	string
Name	产品名称	string
Description	产品描述	string
DataTemplate	产品数据模板	string
NetType	通信方式（子设备为接入网关协议）	string
CategoryId	产品分类 ID	number
ProductType	产品类型（0：普通产品；5：网关产品）	number
UpdateTime	最后更新的 Unix 时间戳（秒级）	number

获取设备信息

• 接口定义

```
sdk.getDeviceInfo({ deviceId?: string }) => Promise<{  
  ProductId: string,  
  DeviceName: string,  
  DeviceId: string,  
  IconUrl: string,  
  AliasName: string,  
  UserId: string,  
  RoomId: string,  
  CreateTime: number,  
  UpdateTime: number  
}|{  
  ProductId: string,  
  DeviceName: string,  
  DeviceId: string,  
  IconUrl: string,  
  AliasName: string,  
  CreateTime: string  
}>
```

• 参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

• 返回值

返回一个 Promise，输出参数请参见 [DeviceList](#)（非分享设备）和 [ShareDevices](#)（分享设备）。

控制设备属性

• 接口定义

```
sdk.controlDeviceData(data, deviceId?: string) => Promise
```

• 参数说明

参数名	参数描述	类型	必填
data	要控制的设备属性数据	object	是
deviceId	可选，不传则使用当前设备的设备 ID	string	否

- 返回值

返回一个 Promise，输出参数请参见 [用户控制设备](#)。

调用设备行为(action)

- 接口定义

```
sdk.callDeviceAction(actionPayload: ActionPayload, actionId: string, deviceId?: string) => Promise
```

- 参数说明

参数名	参数描述	类型	必填
actionPayload	物模型中定义的行为调用入参	object	是
actionId	在物模型中定义的该行为的标志符	string	是
deviceId	可选，不传则使用当前设备的设备 ID	string	否

- 返回值

返回一个 Promise，输出参数请参见 [同步调用设备行为](#)。

获取设备物模型数据

- 接口定义

```
sdk.getDeviceData({ deviceId?: string }) => Promise<object>
```

- 参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

- 返回值

返回一个 Promise，输出参数为设备的物模型数据。

获取设备物模型历史数据

- 接口定义

```
sdk.getDeviceDataHistory({  
  fieldName: string,  
  maxTime: number,
```

```
MinTime: number,  
Context?: string,  
Limit: number  
) => Promise<{  
  RequestId: string,  
  Context: string,  
  FieldName: string,  
  Listover: boolean,  
  Results: DataHistoryItem[]  

```

• 参数说明

参数名	参数描述	类型	必填
FieldName	查询的属性名称	string	是
MaxTime	结束时间，毫秒时间戳	number	是
MinTime	开始时间，毫秒时间戳	number	是
Context	翻页游标，首次查询时可不传	string	否
Limit	单页数据量	number	是

• 返回值

返回一个 Promise，输出参数请参见 [获取设备物模型历史数据](#)。

获取设备当前状态

• 接口定义

```
sdk.getDeviceStatus({ deviceId?: string }) => Promise<0 | 1>
```

• 参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

• 返回值

返回一个 Promise，输出参数取值如下：

- 0：设备离线
- 1：设备在线

删除设备

接口定义

```
sdk.deleteDevice({ deviceId?: string }) => Promise<void>
```

参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise。

获取自定义分享参数

若该设备是分享设备，且分享方设置了自定义分享参数，则被分享方在接受分享后可通过该接口获取自定义分享参数。

接口定义

```
sdk.getShareParams({ deviceId?: string }) => Promise<any>
```

参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise，输出参数为自定义分享参数。

检查设备是否有可升级固件

检查指定设备是否有可升级固件，若有可升级固件，且设备在线，则弹出固件升级提示。

接口定义

```
sdk.checkFirmwareUpgrade({  
  deviceId?: string,  
  silent?: boolean  
}) => Promise<{  
  CurrentVersion: string,  
  DstVersion: string,  
}>
```

• 参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否
silent	可选，默认为 false。 ◦ true: 只检查固件升级，不弹出固件升级提示 ◦ false: 检查固件升级，若有可升级固件，且设备在线，则弹固件升级提示	boolean	否

• 返回值

返回一个 Promise，输出参数请参见 [查询设备固件是否升级](#)。

进行固件升级

跳转到小程序的固件升级页面，进行固件升级。

• 接口定义

```
sdk.goFirmwareUpgradePage({ deviceId?: string }) => Promise
```

• 参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

用户管理

获取用户信息

• 接口定义

```
sdk.getUserInfo() => Promise<{  
  Avatar: string,  
  CountryCode: string,  
  Email: string,  
  NickName: string,  
  PhoneNumber: string,  
  UserID: string  

```

• 返回值

返回一个 Promise，输出参数如下。

参数名	参数描述	类型
Avatar	头像 URL	string
CountryCode	国家代码	string
Email	邮箱地址	string
NickName	昵称	string
PhoneNumber	手机号码	string
UserID	用户 ID	string

界面组件

tips 组件

tips 组件，样式和风格与连连小程序一致。

展示 tips

• 接口定义

```
sdk.tips.show(message, options) => Promise
```

• 参数说明

参数名	参数描述	类型	必填
message	提示文本	string	是
options.type	tips 类型	'info' 'danger' 'loading' 'success'	否
options.waitForHide	若为 true，则 show 方法返回一个 Promise，并且当关闭后才会触发 resolve	boolean	否
options.duration	展示提示的时间，单位毫秒，默认1500	number	否
options.delayDuration	默认为0，单位毫秒，提示会在该延时后展示	number	否
options.canClickClose	默认为 true，点击 mask 是否能够关闭提示	boolean	否
options.canBeReplace	默认为 false，为 false 时上一个提示未关闭前，再次调用 tips.show 会被忽略	boolean	否

关闭 tips

接口定义


```
sdk.tips.hide() => Promise
```

展示 loading tips

封装后的 `tips.show` 方法，等价于：

```
sdk.tips.show(message, {  
  type: 'loading',  
  canBeReplace: true,  
  duration: 0,  
  delayDuration: 200,  
  canClickClose: false,  
  ...options,  
});
```

• 接口定义

```
sdk.tips.showLoading(message, options) => Promise
```

• 参数说明

请参见 [展示 tips 参数说明](#)。

关闭 loading tips

[展示 loading tips](#) 后必须主动调用本接口，否则 tips 将会一直保留。

接口定义

```
sdk.tips.hideLoading() => Promise
```

展示成功 tips

封装后的 `tips.show` 方法，等价于：

```
sdk.tips.show(message, { type: 'success', ...options });
```

• 接口定义

```
sdk.tips.showSuccess(message, options) => Promise
```

- 参数说明

请参见 [展示 tips 参数说明](#)。

展示提示 tips

封装后的 `tips.show` 方法，等价于：

```
sdk.tips.show(message, { type: 'info', ...options });
```

- 接口定义

```
sdk.tips.showInfo(message, options) => Promise
```

- 参数说明

请参见 [展示 tips 参数说明](#)。

展示错误 tips

先标准化处理错误展示信息，然后展示错误 tips。等价于：

```
sdk.tips.show(getErrorMsg(error), { type: 'error', ...options });
```

- 接口定义

```
sdk.tips.showError(error, options) => Promise
```

- 参数说明

参数名	参数描述	类型	必填
error	错误对象或错误信息	Error { code, msg } string	是
options	请参见 展示 tips 参数说明	object	否

展示模态对话框

展示一个弹窗，参数、功能、样式同小程序原生 `showModal` 基本一致。

- 接口定义

```
sdk.tips.showModal({  
  title?: string,  
  content?: string,  
  showCancel?: boolean,
```

```
cancelText?: string,  
cancelColor?: string,  
confirmText?: string,  
confirmColor?: string,  
}) => Promise<boolean>
```

• 参数说明

参数名	参数描述	类型	必填
title	弹窗标题	string	否
content	弹窗内容	string	否
showCancel	是否展示取消按钮，默认为 true	boolean	否
cancelText	取消按钮文案，默认为“取消”	string	否
cancelColor	取消按钮颜色，默认为“#6C7078”	string	否
confirmText	确认按钮文案，默认为“确定”	string	否
confirmColor	确认按钮颜色，默认为“#0066FF”	string	否

• 返回值

返回一个 Promise<boolean>：

- true：代表用户单击确认。
- false：代表用户单击取消。

展示确认模态对话框

基于 sdk.tips.showModal 封装，用于向用户进行二次确认操作时使用。

• 接口定义

```
sdk.tips.confirm(title, content, options) => Promise<boolean>
```

• 参数说明

参数名	参数描述	类型	必填
title	弹窗标题	string	否
content	弹窗内容	string	否
options	请参见 展示模态对话框	object	否

• 示例代码

用户确认示例代码如下：

```
const isConfirm = await sdk.tips.confirm('确认删除该设备吗? ');
if (isConfirm) {
  // 用户确认，执行后续流程
}
```

展示提示模态对话框

基于 `sdk.tips.showModal` 封装，用于向用户进行消息提示操作时使用。

• 接口定义

```
sdk.tips.alert(content, options) => Promise<boolean>
```

• 参数说明

参数名	参数描述	类型	必填
content	弹窗内容	string	否
options	请参见 展示模态对话框	object	否

• 示例代码

向用户进行消息提示的示例代码如下：

```
await sdk.tips.alert('该功能暂时无法使用，请稍后再试');
```

设备离线提示组件

设备离线提示组件，样式和风格与连连小程序一致。

展示设备离线提示

接口定义

```
sdk.offlineTip.show() => void
// 或
sdk.showOfflineTip() => void
```

关闭设备离线提示

接口定义

```
sdk.offlineTip.hide() => void  
// 或  
sdk.hideOfflineTip() => void
```

H5 自定义设备详情视图

展示 H5 自定义设备详情视图

在当前 H5 展示一个铺满全屏的设备详情视图，支持增加自定义菜单项及按钮。

• 接口定义

```
sdk.showDeviceDetail({  
  deviceInfo?: object,  
  labelWidth?: number,  
  marginTop?: number,  
  shareParams?: object,  
  extendItems?: ExtendItemConfig[],  
  extendButtons?: ExtendButtonConfig[],  
  containerClassName?: string  
}) => void
```

• 参数说明

参数名	参数描述	类型	必填
deviceInfo	展示详情的设备信息，不传则使用当前设备信息	object	否
labelWidth	设备详情的 label 宽度，默认 110，单位 px	number	否
marginTop	设备详情的上间距，默认 10，单位 px	number	否
shareParams	自定义分享参数	object string	否
extendItems	自定义菜单配置	ExtendItemConfig[]	否
extendItems[].labelIcon	展示在 label 前的 icon 地址	string	否
extendItems[].label	自定义菜单项的标题	string	是
extendItems[].content	自定义菜单项的内容	string	否
extendItems[].className	自定义菜单项的样式类名	string	否

参数名	参数描述	类型	必填
extendItems[].onClick	点击自定义菜单项后触发的回调	() => any	否
extendButtons	自定义按钮配置	ExtendButtonConfig[]	否
extendButtons[].text	自定义按钮文案	string	是
extendButtons[].className	自定义按钮的样式类名	string	否
extendButtons[].type	自定义按钮的风格	" 'danger' 'primary' 'warning'	否
extendButtons[].onClick	自定义按钮点击后触发的回调	() => any	是
containerClassName	容器的样式类名	string	否

关闭 H5 自定义设备详情视图

接口定义

```
sdk.hideDeviceDetail() => void
```

调用小程序能力

跳转小程序的标准设备详情页面

• 接口定义

```
sdk.goDeviceDetailPage({
  reload?: boolean,
  deviceId?: string,
  isShareDevice?: string,
  shareParams?: object | string,
}) => Promise
```

• 参数说明

参数名	参数描述	类型	必填
reload	- 如果为 true，则进入详情页后会重新拉取一次该设备的数据 - 如果为 false，则进入详情页后会使用缓存的设备数据	boolean	否
deviceId	可选，不传则使用当前设备的设备 ID	string	否

参数名	参数描述	类型	必填
isShareDevice	可选，设备是否为分享设备，不传则使用当前的 sdk.isShareDevice	boolean	否
shareParams	可选，设备自定义分享参数	object string	否

跳转小程序的反馈页面

接口定义

```
sdk.goFeedBackPage() => Promise
```

跳转小程序的设备信息页面

接口定义

```
sdk.goDeviceInfoPage({ deviceId?: string }) => Promise
```

参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

跳转小程序的修改设备名称页面

接口定义

```
sdk.goEditDeviceNamePage: ({ deviceId?: string, name?: string }) => Promise
```

参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否
name	可选，不传则使用当前设备的 aliasName	string	否

跳转小程序的房间设置页面

接口定义

```
sdk.goRoomSettingPage({ deviceId?: string }) => Promise
```

- 参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

跳转小程序的设备分享页面

- 接口定义

```
sdk.goShareDevicePage({ deviceId?: string }) => Promise
```

- 参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

小程序刷新数据

要求小程序在当前 H5 面板关闭后进行一次数据刷新。

接口定义

```
sdk.reloadAfterUnmount() => Promise
```

返回小程序的上一级页面

可用于主动关闭 H5 面板。

接口定义

```
sdk.navBack() => Promise
```

设置当前页面的分享内容

设置当前页面的分享内容，通过 [wx.miniProgram.postMessage](#) 向小程序推送分享信息，具体参考 [小程序页面分享文档](#)。

- 接口定义

```
sdk.setShareConfig({ title: string, imgUrl: string? }) => Promise
```

- 参数说明

参数名	参数描述	类型	必填

参数名	参数描述	类型	必填
title	分享的标题	string	是
imgUrl	分享图片的地址，默认会取当前页面截图	string	否

应用端 API

H5 SDK 对应用端 API 的调用过程进行了封装，发送请求时会自动带上公共参数 `AccessToken` 与 `RequestId`。

调用应用端 API

接口定义

```
sdk.requestTokenApi(action, data, options) => Promise
```

参数说明

参数名	参数描述	类型	必填
action	具体应用端 API Action 名称，如：AppGetDeviceStatuses	string	是
data	接口调用参数	object	否
options	请参见 wx.request	object	否

返回值

- 请求成功（code=0）：返回一个 resolved 的 Promise，其值为应用端 API 响应中的 Response 部分数据。
- 请求失败：返回一个 rejected 的 Promise，其值的数据结构为：{ code, msg, ...detail }。

事件监听

监听事件

接口定义

```
sdk.on(type: string, listener: (...args) => void) => void
```

参数说明

参数名	参数描述	类型	必填
type	要监听的事件	string	是
listener	事件触发时的回调函数	(...args) => void	是

取消监听事件

• 接口定义

```
sdk.off(type: string, listener: (...args) => void) => void
```

• 参数说明

参数名	参数描述	类型	必填
type	要取消监听的事件	string	是
listener	要取消监听的事件的回调函数，不传则清除该事件的所有回调函数	(...args) => void	否

WebSocket 事件

• wsClose 事件: WebSocket 的 close 事件。

参数名	参数描述	类型
code	服务器发送的关闭码	number
reason	服务器关闭连接的原因	string

• wsError 事件: WebSocket 的错误事件。

• wsControl 事件: 当 WebSocket 收到 control 指令后触发。

参数名	参数描述	类型
deviceId	设备 ID	string
deviceData	设备数据	object

• wsReport 事件: 当 WebSocket 收到 report 指令后触发。

参数名	参数描述	类型
deviceId	设备 ID	string
deviceData	设备数据	object

• wsStatusChange 事件: 当 WebSocket 收到 wsStatusChange 指令后触发。

参数名	参数描述	类型
deviceId	设备 ID	string
deviceStatus	设备在线状态	0 1

- **wsEventReport 事件**: 当 WebSocket 收到 wsEventReport 指令后触发。

参数名	参数描述	类型
deviceId	设备 ID	string
Payload	设备回复详情	object

- **wsActionPush 事件**: 当 WebSocket 收到 wsActionPush 指令后触发。

参数名	参数描述	类型
deviceId	设备 ID	string
Payload	下发的 Action 指定详情	object

- **wsActionReport 事件**: 当 WebSocket 收到 wsActionReport 指令后触发。

参数名	参数描述	类型
deviceId	设备 ID	string
Payload	设备回复的 Action 响应详情	object

前后台切换事件

- **appShow 事件**: 当小程序 [App.onShow](#) 执行后触发。
- **appHide 事件**: 当小程序 [App.onHide](#) 执行后触发。
- **pageShow 事件**: 当小程序 [Page.onShow](#) 执行后触发。
- **pageHide 事件**: 当小程序 [Page.onHide](#) 执行后触发。

蓝牙模块

名词解释

名词	解释
serviceld	服务 id, 蓝牙服务的 uuid, 搜索设备时主要通过 serviceld 来过滤我们需要的设备。
deviceId	小程序 API 搜索出来的设备的标识, 连接设备时主要通过 deviceId 来标识需要连接的设备。
explorerDeviceId	物联网开发平台侧定义的设备 ID, 查询设备数据和上报设备数据时以设备 ID 作为设备标识。
BlueToothAdapter 蓝牙适配器	全局单例, 实例上声明了蓝牙搜索、连接等方法。
DeviceAdapter 设	真正用来连接设备以及跟设备进行通信的模块, 每一个设备连接对应一个设备适配器实例,

备适配器

设备适配器会在连接设备后实例化，并在设备断开连接后销毁。根据不同的 `serviceld` 来区别不同类型设备的适配器构造函数。

蓝牙适配器

添加设备适配器

添加一个设备适配器。使用蓝牙模块时需要根据具体设备情况创建一个设备适配器，并调用本接口将其构造函数添加到蓝牙适配器中。H5 SDK 默认添加了一个支持 LLSync 蓝牙协议的设备适配器。

• 接口定义

```
sdk.bluetoothAdapter.addAdapter(deviceAdapter: DeviceAdapterConstructor) => void
```

• 参数说明

参数名	参数描述	类型	必填
deviceAdapter	要添加的设备适配器的构造函数	DeviceAdapterConstructor	是

• 示例代码

```
class DemoDeviceAdapter extends DeviceAdapter {
  static serviceld = '0000FFF0-0000-1000-8000-00805F9B34CC';
  static deviceFilter(deviceInfo) {
    if (deviceInfo.advertisServiceUUIDs) {
      const matchedServiceld = deviceInfo.advertisServiceUUIDs.find(id => id ===
        DemoDeviceAdapter.serviceld);
      if (matchedServiceld && deviceInfo.advertisData) {
        try {
          const macArr = deviceInfo.advertisData.slice(2);
          const mac = macArr.join(':');
          return {
            ...deviceInfo,
            deviceName: mac,
            serviceld: matchedServiceld,
          };
        } catch (err) {
          console.error('parse mac error', err);
        }
      }
    }
  }
}
```

```
handleBLEMessage(hex) {  
  return {  
    type: 'unknown',  
    data: hex,  
  };  
}  
  
sdk.bluetoothAdapter.addAdapter(DemoDeviceAdapter);
```

初始化蓝牙模块

初始化蓝牙模块，包括初始化蓝牙模块、打通小程序间蓝牙通信以及注册全局回调等。本接口可重复调用，可在每次使用蓝牙模块前调用。

• 接口定义

```
sdk.bluetoothAdapter.init() => Promise<void>
```

• 返回值

返回一个带缓存的 Promise，初始化成功后 resolve。若初始化未完成或已初始化成功，则多次调用后返回同一个 Promise。若初始化失败，则该缓存的 Promise 在 reject 之后会被释放，再次调用则将重新初始化。

• 示例代码

```
sdk.bluetoothAdapter.init().then(() => {  
  // 调用蓝牙模块能力  
});
```

开始搜索蓝牙设备

开始搜索蓝牙设备。（将会调用 [wx.startBluetoothDevicesDiscovery](#)，比较耗费系统资源，务必在不需要搜索后调用 [停止搜索蓝牙设备](#)，如离开页面后）

• 接口定义

```
sdk.bluetoothAdapter.startSearch({  
  serviceId?: string,  
  serviceIds?: string[],  
  ignoreDeviceIds?: string[],  
  onSearch: (DeviceInfo[]) => void,  
  onError: (Error) => void,  
  timeout: number  
}) => Promise<void>
```

• 参数说明

参数名	参数描述	类型	必填
serviceld	指定需要搜索的 serviceld，不传的话会使用当前支持的所有 DeviceAdapter 的 serviceld 来匹配	string	否
servicelds	参数描述同 serviceld	string[]	否
ignoreDeviceIds	可选，需要过滤掉的 deviceId 列表（例如刚添加完的设备），搜索结果中将不会出现这些设备	string[]	否
onSearch	当搜索结果更新后调用，返回搜索到的设备列表	(DeviceInfo[]) => void	是
onError	当搜索过程中发生错误后调用，触发后设备搜索将会中止	(Error) => void	是
timeout	可选，默认20000，单位毫秒，超过指定时长没有搜索到设备后将会触发超时错误	number	是

• 返回值

返回一个 Promise。

停止搜索蓝牙设备

接口定义

```
sdk.bluetoothAdapter.stopSearch() => void
```

搜索单个蓝牙设备

开始搜索蓝牙设备，并在搜索到第一个满足条件的设备后停止搜索。

• 接口定义

```
sdk.bluetoothAdapter.searchDevice({  
  deviceName: string,  
  serviceld?: string,  
  servicelds?: string[],  
  ignoreDeviceIds?: string[]  
}) => Promise<DeviceInfo>
```

• 参数说明

参数名	参数描述	类型	必填

参数名	参数描述	类型	必填
deviceName	指定需要搜索的设备 deviceName	string	是
serviceId	指定需要搜索的 serviceId，不传的话会使用当前支持的所有 DeviceAdapter 的 serviceId 来匹配	string	否
serviceIds	参数描述同 serviceId	string[]	否
ignoreDeviceIds	可选，需要过滤掉的 deviceId 列表（例如刚添加完的设备），搜索结果中将不会出现这些设备	string[]	否

• 返回值

返回一个 Promise，在找到第一个满足条件的设备后 resolve。

连接蓝牙设备

连接指定蓝牙设备。

• 接口定义

```
blueToothAdapter.connectDevice(deviceInfo: DeviceInfo, options?: { autoNotify?: boolean }) => Promise<DeviceAdapter>
```

• 参数说明

参数名	参数描述	类型	必填
deviceInfo	传入 开始搜索蓝牙设备 或 搜索单个蓝牙设备 接口搜索出来的 DeviceInfo	DeviceInfo	是
options.autoNotify	可选，默认为 true。指定为 true 时，在连接设备后，会自动去拉取服务列表，以及主服务下的特征值列表，并会自动订阅第一个 notifyId 或 indicateId 特征值的 notify。若设备含有多个服务或多个 notify 特征值，请传 false，并自行通过 getBLEDeviceServices、getBLEDeviceCharacteristics、notifyBLECharacteristicValueChange 等方法获取及订阅特征值。	boolean	否

• 返回值

返回一个 Promise，连接成功后返回设备适配器。

获取设备适配器实例

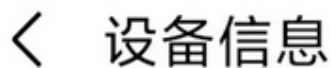
根据 deviceId 查询对应的设备适配器实例。

```
sdk.bluetoothAdapter.getDeviceAdapter({explorerDeviceId: string}) => DeviceAdapter
```

参数名	参数描述	类型	必填
explorerDeviceId	设备适配器实例的 explorerDeviceId,可以通过 sdk.deviceId 获得	string	是

返回与传入 `explorerDeviceId` 相匹配的设备适配器实例，如果找不到，则返回 `undefined`。

蓝牙设备不能通过mqtt直接上报设备的mac地址等信息，所以需要H5端进行上报，对应的是设备详情里面的设备信息



⚠ 注意:

图片里面厂家名称和产品型号是在设备量产时在控制台填写的，mac 地址，固件版本等由 H5 端进行上报

• 接口定义

```
sdk.bluetoothAdapter.reportDeviceInfo({ productId: string, deviceName: string, deviceInfo: any }) => Promise;
```

• 参数说明

参数名	参数描述	类型
productId	产品 ID	string
deviceName	设备名称	string
deviceInfo	设备信息	any, 详情见下面示例

```
deviceInfo: {  
  "module_hardware": "模组具体硬件型号 N10",  
  "module_software": "模组软件版本",  
  "fw_ver": "mcu 固件版本",  
  "imei": "设备 imei 号, 可选上报",  
  "mac": "设备 mac 地址, 可选上报",  
  "device_label": {  
    "append_info": "设备商自定义的产品附加信息"  
  }  
}
```

监听事件

监听蓝牙适配器事件。

• 接口定义

```
sdk.bluetoothAdapter.on(type: string, listener: (...args) => void) => void
```

• 参数说明

参数名	参数描述	类型	必填

参数名	参数描述	类型	必填
type	要监听的事件	string	是
listener	事件触发时的回调函数	(...args) => void	是

取消监听事件

取消监听蓝牙适配器事件。

• 接口定义

```
sdk.bluetoothAdapter.off(type: string, listener: (...args) => void) => void
```

• 参数说明

参数名	参数描述	类型	必填
type	要取消监听的事件	string	是
listener	要取消监听的事件的回调函数，不传则清除该事件的所有回调函数	(...args) => void	否

蓝牙适配器事件

adapterStateChange 事件：当适配器状态变化时触发。

参数名	参数描述	类型
available	蓝牙适配器是否可用	boolean
discovering	蓝牙适配器是否处于搜索状态	boolean

设备适配器

自定义设备适配器

自定义设备适配器类需要继承 DeviceAdapter，并补充以下实现。

- **serviceld：**自定义设备适配器类需要设置该属性，代表该设备的主服务 ID。
- **deviceFilter：**自定义设备适配器类需要实现该静态方法，在搜索蓝牙设备时会每个搜出的设备信息传入该方法，如果判断是本产品的设备，则需在除入参 deviceInfo 之外返回设备唯一标识 deviceName 及 serviceld，否则返回空。

```
DeviceAdapter.deviceFilter: (deviceInfo: DeviceInfo) => {  
  deviceName: string,  
  serviceld: string,  
}
```

```
...deviceInfo
}
```

- **handleBLEMessage**: 自定义设备适配器类需要实现该方法，用于处理收到 onBLECharacteristicValueChange 回调后的协议解析。

```
DeviceAdapter.handleBLEMessage: (hexString, { serviceId, characteristicId }) => {
  reportData?: any,
  ...any
}
```

返回值中如果返回 reportData，则会将该部分数据上报到云端（注意需与产品定义物模型匹配），其他字段则会透传到 message 事件的 payload 中。

设备适配器属性

属性名	属性描述	类型
explorerDeviceId	只读，设备的 explorerDeviceId	string
isConnected	只读，当前是否已连接设备	boolean
deviceId	只读，设备的 deviceId	string
serviceId	只读，设备的主服务 ID，与构造函数上的静态属性 DeviceAdapter.serviceId 一致	string
originName	只读，设备的原始名称，即小程序接口搜索出来时的 name 字段	string

断开设备连接

接口定义

```
deviceAdapter.disconnectDevice() => void
```

获取设备服务列表

接口定义

```
deviceAdapter.getBLEDeviceServices() => Promise<ServiceList>
```

返回值

返回一个 Promise，ServiceList 数据结构请参见 [wx.getBLEDeviceServices](#)。

获取设备指定服务的特征值列表

获取设备指定服务中的特征值列表，并将特征值存放在 `deviceAdapter` 实例上。

接口定义

```
deviceAdapter.getBLEDeviceCharacteristics({ servid: string }) => Promise<CharacteristicsList>
```

参数说明

参数名	参数描述	类型	必填
<code>servid</code>	可选，指定要获取特征值列表的服务 ID，默认为主服务 ID	string	否

返回值

返回一个 `Promise`，`CharacteristicsList` 数据结构请参见 [wx.getBLEDeviceCharacteristics](#)。

说明

将获取到的特征值按照如下数据结构存放在 `deviceAdapter` 实例上。

```
deviceAdapter.characteristicsMap[servid] = {  
  notifyIds: string[],  
  indicateIds: string[],  
  writeIds: string[],  
  readIds: string[]  
}
```

协商设置蓝牙最大传输单元

本接口仅支持在安卓系统下调用，iOS 因系统限制不支持。

接口定义

```
deviceAdapter.setBLEMTU({  
  mtu: number  
}) => Promise
```

参数说明

参数名	参数描述	类型	必填
<code>mtu</code>	最大传输单元。设置范围为 (22,512) 区间内，单位为 bytes	number	是

读取指定特征值的二进制数据

接口定义

```
deviceAdapter.readBLECharacteristicValue({
  serviceId?: string,
  characteristicId: string
}) => Promise<void>
```

• 参数说明

参数名	参数描述	类型	必填
serviceId	可选，默认为主服务 ID	string	否
characteristicId	需要读取的特征值 ID，默认会取主服务下的第一个 read 特征值	string	是

• 返回值

返回一个 Promise。接口读取到的信息需要在 onBLECharacteristicValueChange 方法注册的回调中获取，具体参见 [wx.readBLECharacteristicValue](#)。

获取蓝牙设备信号强度

• 接口定义

```
deviceAdapter.getBLEDeviceRSSI() => Promise
```

• 返回值

返回一个 Promise，具体数据结构请参见 [wx.getBLEDeviceRSSI](#)。

启用特征值变化时的 notify 功能

• 接口定义

```
deviceAdapter.notifyBLECharacteristicValueChange({
  characteristicId?: string,
  serviceId?: string,
  state?: boolean
}) => Promise<void>
```

• 参数说明

参数名	参数描述	类型	必填
serviceId	需要订阅的服务 ID，默认会取主服务 ID	string	否

参数名	参数描述	类型	必填
characteristicId	需要订阅的特征值 ID，默认会取主服务下的第一个 notify 或 indicate 特征值	string	否
state	是否启用 notify，默认为 true	boolean	否

- 返回值

返回一个 Promise。

写入二进制数据到指定特征值中

- 接口定义

```
deviceAdapter.write(hexString: string, options?: {  
  writeId?: string,  
  serviceId?: string  
}) => Promise
```

- 参数说明

监听

参数名	参数描述	类型	必填
hexString	需要写给蓝牙设备的十六进制字符串	string	是
options.writeId	可选，需要写入的特征值 ID，默认会取主服务下的第一个 writeId	string	否
options.serviceId	可选，需要写入的服务 ID，默认会取主服务 ID	string	否

监听事件

可以使用 deviceAdapter.on 来 [监听设备适配器事件](#)。

- 接口定义

```
deviceAdapter.on(type: string, listener: (...args) => void) => void
```

- 参数说明

参数名	参数描述	类型	必填
type	要监听的事件	string	是
listener	事件触发时的回调函数	(...args) => void	是

取消监听事件

取消监听设备适配器事件。

• 接口定义

```
deviceAdapter.off(type: string, listener: (...args) => void) => void
```

• 参数说明

参数名	参数描述	类型	必填
type	要取消监听的事件	string	是
listener	要取消监听的事件的回调函数，不传则清除该事件的所有回调函数	(...args) => void	否

设备适配器事件

- **connect 事件**：设备连接后触发。
- **disconnect 事件**：设备断开后触发。
- **message 事件**：当收到 onBLECharacteristicValueChange 回调，并经过 handleBLEMessage 处理后触发。

参数名	参数描述	类型
timestamp	收到设备消息的时间戳，单位毫秒	number
dataReported	收到设备的消息是否已上报云端	boolean
（其他）	handleBLEMessage 函数返回的其他参数将会透传到 message 事件中	any

- **BLEConnectionStateChange 事件**：当 onBleConnectionStateChange 触发时触发，若 connected 为 true，则接下来会触发 connect 事件，否则会触发 disconnect 事件。

参数名	参数描述	类型
connected	设备是否连接	boolean

标准蓝牙协议

LLSync 标准蓝牙协议设备的绑定流程在小程序中进行，开发者只需在自定义 H5 面板中关注设备的连接、控制操作与事件监听。自定义 H5 面板已默认添加 LLSync 标准蓝牙协议的设备适配器（StandardDeviceAdapter），支持通过 LLSync 标准蓝牙协议与设备通信。H5 面板 Demo 中提供了 [标准蓝牙协议 demo](#) 供开发者参考。

搜索并连接设备

与标准蓝牙协议设备建立连接的过程分为三个步骤：搜索设备、连接设备、连接鉴权。示例代码如下：

```
import { StandardDeviceAdapter } from 'qcloud-iotexplorer-h5-panel-sdk';
```

```
// 初始化蓝牙适配器
await sdk.bluetoothAdapter.init();

// 搜索设备
const deviceInfo = await sdk.bluetoothAdapter.searchDevice({
  deviceName: sdk.deviceName,
  serviceId: StandardDeviceAdapter.serviceId,
  productId: sdk.productId,
  disableCache: true,
});

if (!deviceInfo) {
  throw new Error('未搜索到设备');
}

// 连接设备
const deviceAdapter = await sdk.bluetoothAdapter.connectDevice({
  ...device,
  productId: sdk.productId,
});

// 连接鉴权
if (!deviceAdapter.authorized) {
  await deviceAdapter.authenticateConnection({
    deviceName: sdk.deviceName,
  });
}
```

设备通信

通过上述流程与设备建立连接后，H5 面板即可控制设备，接收设备的属性、事件上报。

- 控制设备

接口定义请参见 [控制设备属性](#)，示例代码如下：

```
sdk.controlDeviceData({
  // 要控制的设备属性
  power_switch: 1
});
```


• 监听设备事件

接口定义请参见 [事件监听](#)，示例代码如下：

```
// 监听设备属性上报
sdk.on('wsReport', ({ deviceId, deviceData }) => {
  console.log('device', deviceId, 'report_property', deviceData);
});

// 监听设备事件上报
sdk.on('wsEventReport', ({ deviceId, Payload }) => {
  console.log('device', deviceId, 'report_event', Payload);
});
```

设备适配器属性

属性名	属性描述	类型
explorerDeviceId	只读，设备在物联网开发平台的 DeviceId	string
isConnected	只读，当前是否已连接设备	boolean
deviceId	只读，小程序蓝牙接口提供的 deviceId	string
serviceId	只读，设备的主服务 ID，与构造函数上的静态属性 DeviceAdapter.serviceId 一致	string
originName	只读，设备的原始名称，即小程序接口搜索出来时的 name 字段	string
bleVersion	只读，LLSync 协议版本号	number
mtu	只读，设备要求设置的最大传输单元 (MTU) 大小	number
otaVersion	只读，设备固件版本号	string
extendInfo.macStr	只读，设备的 mac 地址	string
authorized	只读，是否与设备完成鉴权	boolean

BLE + WIFI 双路通信

针对 Wi-Fi + BLE Combo 模组，提供设备端 SDK 和 H5 SDK，支持设备 Wi-Fi 离线状态下，小程序通过 LLSync 标准蓝牙协议与设备通信，为用户提供 Wi-Fi 断网下的更佳体验。设备端 SDK 请参考 [开发指引](#)。

对于自定义 H5 面板，配网流程无需开发者关注，开发者需要在面板中关注：

- 1. 监听设备在线状态，wifi连接是否正常；
- 2. 当设备离线时启用蓝牙进行通信。下面分开说明。

监听设备在线状态

我们可以通过监听 [WebSocket 事件](#) 中的 `wsStatusChange` 事件来感知设备的在线离线状态。

```
sdk.on('wsStatusChange', ({deviceId, deviceStatus}) => {  
  if (deviceStatus === 0) {  
    // 设备已离线，开始启用蓝牙连接进行通信，见第二步  
  }  
})
```

启用蓝牙通信

我们提供了 [双路通信 demo](#) 供参考。

STEP1: 添加 BleComboDualModeDeviceAdapter4H5

BleComboDualModeDeviceAdapter4H5 是一个设备适配器实例，如果您还不了解设备适配器，请阅读 [设备适配器部分](#)。

```
import { BleComboDualModeDeviceAdapter4H5 } from 'qcloud-iotexplorer-appdev-plugin-wificonf-ble-combo/lib/protocols/BleComboDualMode';  
  
const sdk = window.h5PanelSdk;  
BleComboDualModeDeviceAdapter4H5.injectOptions({ appDevSdk: sdk.appDevSdk });  
  
sdk.blueToothAdapter.addAdapter(BleComboDualModeDeviceAdapter4H5, true);
```

STEP2: 搜索设备

`searchDevice` 的参数详见 [蓝牙适配器](#) 搜索单个蓝牙设备的参数说明部分。

```
await blueToothAdapter.init();  
console.log('开始搜索设备', sdk.deviceName);  
const deviceInfo = await blueToothAdapter.searchDevice({  
  deviceName: sdk.deviceName,  
  serviceId: BleComboDualModeDeviceAdapter4H5.serviceId,  
  productId: sdk.productId,  
  disableCache: true,  
});
```

STEP3: 连接设备

使用上一步获取到的 `deviceInfo`，我们可以调用 `connectDevice` 连接设备以获得 `deviceAdapter`，传入参数定义详见 [连接蓝牙设备](#) 部分。

```
const deviceAdapter = await blueToothAdapter.connectDevice({
  ...device,
  productId: sdk.productId,
});
console.log('deviceAdapter:', deviceAdapter);
// authorized之后，才能向设备发送控制数据
if (!deviceAdapter.authorized) {
  await deviceAdapter.authenticateConnection({
    deviceName: sdk.deviceName,
  });
}
```

当小程序和设备间的蓝牙连接成功后，面板就可以对设备进行控制了，`sdk`会将控制数据通过蓝牙发送到设备。比如进行开关的控制：

```
sdk.controlDeviceData({ power_switch: 1 });
```

ASR 语音识别

语音识别

目前支持两种场景，分别为“录音文件”与“一句话识别”，两种场景下入参会有所不同。

由于识别过程是异步，因此接口“`voiceRecognition`”并不会立即返回识别结果，可以理解为该接口是创建了一个“异步任务”，当这个成功被创建的“异步任务”执行完后，会通过 `websocket` 将结果推送到所有订阅该 `deviceId` 的终端上，下文为您介绍详细介绍 ASR 语音识别。

voiceRecognition

接口定义

```
sdk.voiceRecognition({...}) => Promise<{...}>
```

参数说明

- 关于“录音文件”场景支持的音频类型、大小限制以及相关字段的详细介绍，详情请参见 [录音文件识别](#)。
- 关于“一句话识别”场景支持的音频类型、大小限制以及相关字段的详细介绍，详情请参见 [一句话识别](#)。

公共参数

参数名	参数描述	类型	必填
DeviceId	默认使用当前设备的设备 ID	string	否
AudioType	识别场景。 “录音文件” 取值 “file” “一句话识别” 取值 “sentence”	string	是
Data	音频文件	Blob File	是
ResourceName	文件名称，如果 Data 类型是 File，则取其 “name” 作为默认值	string	否
EngineType	引擎模型类型，默认值为 “16k_zh”	string	否
FilterDirty	是否过滤脏词	number	否
FilterModal	是否过滤语气词	number	否
FilterPunc	是否过滤标点符号	number	否
ConvertNumMode	是否进行阿拉伯数字智能转换	number	否

录音文件额外参数

参数名	参数描述	类型	必填
ChannelNum	语音声道数，默认为1	number	否
SpeakerDiarization	是否开启话者分离	number	否
SpeakerNumber	话者分离人数	number	否

返回值说明

参数名	参数描述	类型
ResourceToken	某个设备下，音频文件的唯一标示	string

监听识别结果 – asrResponse

对于“录音文件”场景，如果音频文件过大，后台可能会对音频文件进行分片识别，每个分片识别完成后，都将推送一条 websocket 消息，但推送的消息不保证顺序（例如有可能分片2的结果先到达）。

对于“asrResponse”事件，实际是基于“wsControl”事件进行二次封装；当然，您也可以通过监听“wsControl”事件获取识别结果。

接口定义

```
sdk.on('asrResponse', ({ deviceId, data }) => void)
```

返回值说明

参数名	参数描述	类型
deviceId	设备 ID	string
data	识别结果数据	object
data.resource_token	某个设备下，音频文件的唯一标示	string
data.result_code	状态码，0代表成功	number
data.total_num	分片总数	number
data.seq	当前分片序号	number
data.res_text	当前分片识别结果，对于“录音文件”场景，识别结果会包含分段时间戳	string

获取语音文件下载链接（仅限“录音文件”场景）

接口定义

```
sdk.getAsrDownloadUrl({...}) => Promise<{...}>
```

参数说明

参数名	参数描述	类型	必填
DeviceId	设备 ID	string	是
ResourceToken	调用 voiceRecognition 返回的 ResourceToken	string	是

返回值说明

参数名	参数描述	类型
ResourceURL	cos 访问链接	string

TRTC音视频

腾讯云物联网开发平台实时音视频服务，调用该接口从 h5 跳转至腾讯连连小程序 TRTC 页面。

接口定义

```
sdk.TRTCManager.goTRTCPage()
```

参数说明

参数名	参数描述	类型	必填
callType	呼叫类型：video audio	string	是

音乐服务

公共说明

获取实例

```
const tmeSdk = await sdk.getTMESdk();
```

接口通用返回

接口统一返回值

TMESDK接口调用的返回值统一为 Promise<TMEResponse> 类型

```
interface TMEResponse {  
  error_code: number,  
  error_msg: string,  
  data?: any;  
}
```

- 调用成功：返回一个resolved的Promise，其值为TMEResponse类型，error_code=0，data为返回结果
- 调用失败：返回一个rejected的Promise，包含错误码（error_code）及提示信息（error_msg）

属性名	描述	类型
error_code	错误码	number
error_msg	错误信息	string
data	响应数据	object

错误码列表

错误码	说明
200001	参数错误
200002	系统繁忙,如幂等接口并发调用等, 通常由于用户并发操作造成
200003	认证信息过期或错误,请重新登录
200004	设备未激活
200005	当前sp暂未支持此接口
200006	系统错误,如内部调用超时等, 由于服务内部异常导致
200200	可直充剩余次数为0
400000	登录授权失败
400001	设备端超时无响应
400002	调用SDK参数错误

登录授权部分

用户设备登录授权

跳转酷狗音乐小程序授权,当再次返回h5时,Promise状态改变。

接口定义

```
tmeSdk.login(deviceId?: string) => Promise
```

参数说明

参数名	参数描述	类型	必填
deviceId	可选, 不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>

用户设备登出

原token将登出

接口定义

```
tmeSdk.logout(deviceId?: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>

校验设备授权

接口定义

```
tmeSdk.checkDeviceAuth(deviceId?: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>

获取用户信息

接口定义

```
tmeSdk.getUserInfo(deviceId?: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>，其中data如下：

属性名	描述	类型

属性名	描述	类型
userid	用户id	string
nick_name	用户昵称	string
img	用户头像	string
is_vip	是否vip 0:否 1:是	enum: 0 1
vip_end_time	vip有效期终止时间	string
car_vip_end_time	车机会员有效期终止时间	string
svip_end_time	豪V有效期终止时间	string

播控部分

接口描述

调用播控SDK，会下发物模型属性+control_seq，需要设备上报相同的control_seq

- 若在超时范围内收到上报，视为下发播控成功，返回resolved状态的Promise<TMEResponse>
- 若超时未收到上报，返回rejected状态的Promise<TMEResponse>

超时设置可以通过 tmeSdk.config.timeout 来配置，默认值为10000，单位：毫秒(ms)

通用播控接口

接口定义

```
tmeSdk.controlKugouDeviceData(deviceData, deviceId?: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
deviceData	设备物模型数据	object	是
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>

播放

接口定义

```
tmeSdk.play(deviceId?: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>

暂停

接口定义

```
tmeSdk.pause(deviceId?: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>

上一首

接口定义

```
tmeSdk.preSong(deviceId?: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>

[下一首](#)

接口定义

```
tmeSdk.nextSong(deviceId?: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>

设置播放模式

接口定义

```
tmeSdk.setPlayMode(playMode: number, deviceId?: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
playMode	播放模式：0:顺序播放 1:单曲循环 2:随机播放	enum: 0 1 2	是
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>

设置音量

接口定义

```
tmeSdk.setVolume(volume: number, deviceId?: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
volume	音量：0-100之间	number	是

参数名	参数描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>

设置播放进度

接口定义

```
tmeSdk.setPlayPosition(playPosition: number, deviceId?: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
playPosition	播放进度：单位:秒(s)	number	是
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>

设置播放质量

接口定义

```
tmeSdk.setPlayQuality(recommendQuality: number, deviceId?: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
recommendQuality	播放质量：0:标准 1:高清 2:无损	enum: 0 1 2	是
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>

设置当前播放歌曲

接口定义

```
tmeSdk.playSong(songId: string, songIndex: string, newQueueType: string, newQueueId: string | number, deviceId?: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
songId	歌曲ID	string	是
songIndex	歌曲所在播放列表的位置，从0开始	number	是
newQueueType	播放列表的类型： playlist newSongs recommendDaily	string	是
newQueueId	播放列表ID（当类型为"每日推荐"时，不存在id，传 undefined）	string number	是
deviceId	可选，不传则使用当前设备的设备 ID	string	否

播放列表目前支持三种类型：歌单(playlist)、新歌首发(newSongs)、每日推荐(recommendDaily)

返回值

返回一个 Promise<TMEResponse>

内容部分

拉取内容通用接口

请求酷狗API拉取内容通用接口

接口定义

```
tmeSdk.requestTMEApi(action: string, params, deviceId?: string) => Promise<TMEResponse>
```

参数说明

参数名	描述	类型	必填
action	接口action	string	是
params	请求参数	object	否
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>

注：action、params 及 返回值 data 参考 [音乐服务](#)

获取设备当前播放歌曲

接口定义

```
tmeSdk.getCurrentPlaySong(deviceId?: string) => Promise<TMEResponse>
```

参数说明

参数名	描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>，data为歌曲信息

获取设备当前播放列表

根据目前支持的播放类型(playType)，拉取对应的歌单列表，并查出歌曲的详细信息

接口定义

```
tmeSdk.getCurrentPlayQueue(deviceId?: string) => Promise<TMEResponse>
```

参数说明

参数名	描述	类型	必填
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>

TMEResponse中data如下

属性名	描述	类型
playType	播放列表类型	enum: playlist newSongs recommendDaily

属性名	描述	类型
queueId	当前播放列表id，根据playType对应playlist_id、album_id、top_id	string number
total	列表中歌曲总数	number
songs	歌曲数组，具体歌曲属性参考TME文档中Song属性	Array[]

获取歌曲详细信息

接口定义

```
tmeSdk.getSongDetail(songId: string, deviceId?: string) => Promise<TMEResponse>
```

通过调用requestTMEApi，请求歌曲播放链接与歌曲信息，返回歌曲的详细信息

参数说明

参数名	描述	类型	必填
songId	歌曲ID	string	是
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>，data为歌曲信息

获取歌单详细信息

接口定义

```
tmeSdk.getPlaylistDetail(action: string, params, deviceId?: string) => Promise<TMEResponse>
```

通过调用requestTMEApi，请求歌单列表与歌曲信息，丰富列表中的歌曲信息，返回歌单列表

参数说明

参数名	描述	类型	必填
action	新歌首发(awesome_newsong)、每日推荐(awesome_everyday)、歌单歌曲(playlist_song)	string	是
params	参考应用端API=>音乐服务中对应API的KugouParams	object	是
deviceId	可选，不传则使用当前设备的设备 ID	string	否

返回值

返回一个 Promise<TMEResponse>，data为歌单列表

蓝牙设备调用微信背景音乐

说明：蓝牙设备可在自定义H5面板中，调用微信背景音乐接口。

应用场景：开发者可通过BT+BLE的蓝牙设备对接到连连小程序，在H5页面中，BLE作为播控的通道，通过小程序到手机端的BT链路进行音乐播放。

获取实例

```
const bam = await h5PanelSdk.wxapi.getBackgroundAudioManager();
```

获取属性

获取BackgroundAudioManager实例的属性

接口定义

```
bam.getBackgroundAudioAttribute(keys: Array<string>) => Promise<any>
```

参数说明

参数名	描述	类型	必填
keys	属性名数组	Array<string>	是

设置属性

设置BackgroundAudioManager实例的属性

接口定义

```
bam.setBackgroundAudioAttribute(parmas: Object) => Promise<any>
```

参数说明

参数名	描述	类型	必填
params	key为属性名称，value为属性值	Object	是

其他方法

所有api使用方法与微信官方文档相同

微信背景音乐接口(BackgroundAudioManager)

使用示例

```
// 播放音乐
bam.setBackgroundAudioAttribute({
  title: '歌曲名称',
  singer: '歌手',
  epname: '专辑名',
  src: 'https://**.mp3',
  coverImgUrl: 'https://**.png',
});
// 事件监听
bam.onTimeUpdate((currentTime) => {
  // 更新播放进度
  setCurrentTime(currentTime);
});
```

底层 SDK 能力

应用开发 SDK

H5 SDK 底层依赖应用开发小程序端 SDK。通过以下代码可以获取应用开发 SDK 的实例，更多调用能力请参见 [应用开发小程序端 SDK](#) 文档。

接口定义

```
sdk.appDevSdk
```

微信 JS SDK

通过以下代码可以获取微信 JSSDK 实例，具体用法请参见 [小程序 web-view 文档](#)。使用前需要先调用 [初始化 JSSDK](#)。

接口定义

```
sdk.wx
```

初始化 JS SDK

• 接口定义

```
sdk.wxSdkReady() => Promise
```

- **返回值**

返回一个带缓存的 Promise，初始化成功后 resolve。若初始化未完成或已初始化成功，则多次调用后返回同一个 Promise。若初始化失败，则该缓存的 Promise 在 reject 之后会被释放，再次调用则将重新初始化。

- **示例代码**

```
sdk.wxSdkReady().then(() => wx.miniProgram.navigateBack());
```

自主品牌小程序概述

最近更新时间：2021-04-26 15:12:03

当前消费物联网领域更多的是通过 App 去管理、操控设备，主要存在以下两大问题。

- 对消费者来说，存在着 App 下载门槛高，打开率低的问题。
- 对设备厂商来说，通过产品的智能化升级以便更好的连接消费者，为消费者提供更便捷的服务，但实际上 App 在实际应用时打开率并不是特别高，导致厂商无法获取更多的消费者信息反馈。

针对以上两个问题，微信小程序可有效解决 App 带来的问题，消费者无需下载安装 App，扫码使用即可。为了更好的提升用户体验，腾讯云物联网开发平台秉持开放策略，向开发者开放小程序 SDK，以便需要自主品牌小程序的开发者，能快速基于小程序 SDK 构建自主品牌小程序，降低研发成本，并提升消费者用户体验。

平台功能

小程序 SDK 将通用能力进行封装，开发者无需投入资源研发，小程序 SDK 包含以下功能：

- 设备配网
- 设备管理
- 长连接通信
- 应用端 API 封装

快速入门

最近更新时间：2021-05-11 09:06:46

腾讯云物联网开发平台为开发者提供一个 Demo 小程序参考，开发者可以按本文档的指引流程部署并体验一个属于自己的 Demo 小程序。Demo 小程序以开源的方式向开发者开放，便于开发者在开发自主品牌小程序时进行参考。在开始之前需完成以下操作：

- 已注册 [腾讯云账号](#) 并完成 [实名认证](#)。
- 已安装 [微信开发者工具](#)。
- 已安装 [Node.js](#)。

步骤1：创建项目、产品及设备

本步骤指导您通过物联网开发平台的快速入门功能，创建一个智能灯产品及设备，并运行平台生成的模拟设备 Demo。如需自行创建项目、产品和设备，请参见 [产品开发](#)。

1. 登录腾讯云 [物联网开发平台控制台](#)，单击【公共实例】或您创建的【标准企业实例】进入项目管理页面。
2. 单击页面左上角的【快速入门】，进入快速连接设备页面。
3. 单击【开始使用】进入设备新建界面，按照实际情况填写以下信息：

快速连接设备

1 新建设备 > 2 下载设备Demo > 3 连接和测试

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

产品名称 *

支持中文、英文、数字、下划线的组合，最多不超过20个字符

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

- **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。
- **产品名称：**支持中文、英文、数字、下划线的组合，最多不超过20个字符。
- **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。

4. 根据您的电脑的操作系统，单击页面下方 Windows、Linux 或 OSX 操作系统的【设备 Demo 下载】。

设备Demo下载



5. 然后单击【确定下载】，下载设备 Demo 程序到本地。

下载提醒

请确认您的电脑为Win 10 64位操作系统，否则设备Demo可能会运行失败。



6. 单击【下一步】，根据页面指引运行设备 Demo 程序。运行设备 Demo 程序后，页面显示设备已激活，并显示设备上报的数据。单击【完成】。

快速连接设备

✓ 新建设备 > ✓ 下载设备Demo > 3 连接和测试

运行设备Demo，设备连接到云端后自动上报数据

Windows系统

在电脑解压设备Demo压缩包后，直接运行quick_start_sample.exe程序。

Linux/OSX系统

第一步：在电脑解压设备Demo压缩包，可以执行下面的命令行：

```
tar -vxf tencentcloud_iot_device_demo-*.tar.gz
```

[复制](#)

第二步：进入文件，运行设备Demo程序，可以执行下面的命令行：

```
sh ./run.sh
```

[复制](#)

第三步：连接成功后，下面将显示物模型的数据（注：脚本会每隔5秒上报数据，数据为："switch":"0","color":"0","brightness":"1"）

设备通信日志 设备未激活

[上一步](#)[完成](#)

7. 页面提示已成功将设备接入腾讯云物联网开发平台，单击【完成】。

步骤2：获取小程序 AppKey 与 AppSecret

1. 登录腾讯云 [物联网开发平台控制台](#)，选择 [步骤1](#) 中创建的项目，进入项目管理页面。
2. 在左侧菜单中选择【应用开发】，单击【新建应用】，进入新建应用页面。

3. 填写应用信息。

新建应用



应用名称 *

支持中文、英文、数字、下划线的组合，最多不超过50个字符

备注

选填

最多不超过1024个字符

保存

取消

- **应用名称**：必填，根据实际业务填写便于识别的应用名称。
 - **备注**：非必填，填写应用的备注信息（选填）。
4. 单击【保存】，页面提示保存成功，回到应用开发页面。
5. 单击应用列表中应用名称，进入应用详情页面。
6. 记录小程序应用下显示的【APP Key】和【APP Secret】。

小程序应用 ①

APP Key

APP Secret

7. 在页面下方的关联产品列表中，单击【关联】列的开关，使小程序与 [步骤1](#) 中创建的产品关联（开关为开启状态）。

⚠ 注意：

小程序只能对已关联产品下的设备进行绑定、控制等操作。

关联产品

① 应用与产品关联后，可以在应用中对产品下的设备进行绑定、控制等操作。

产品名称

状态

关联

小程序关联的产品

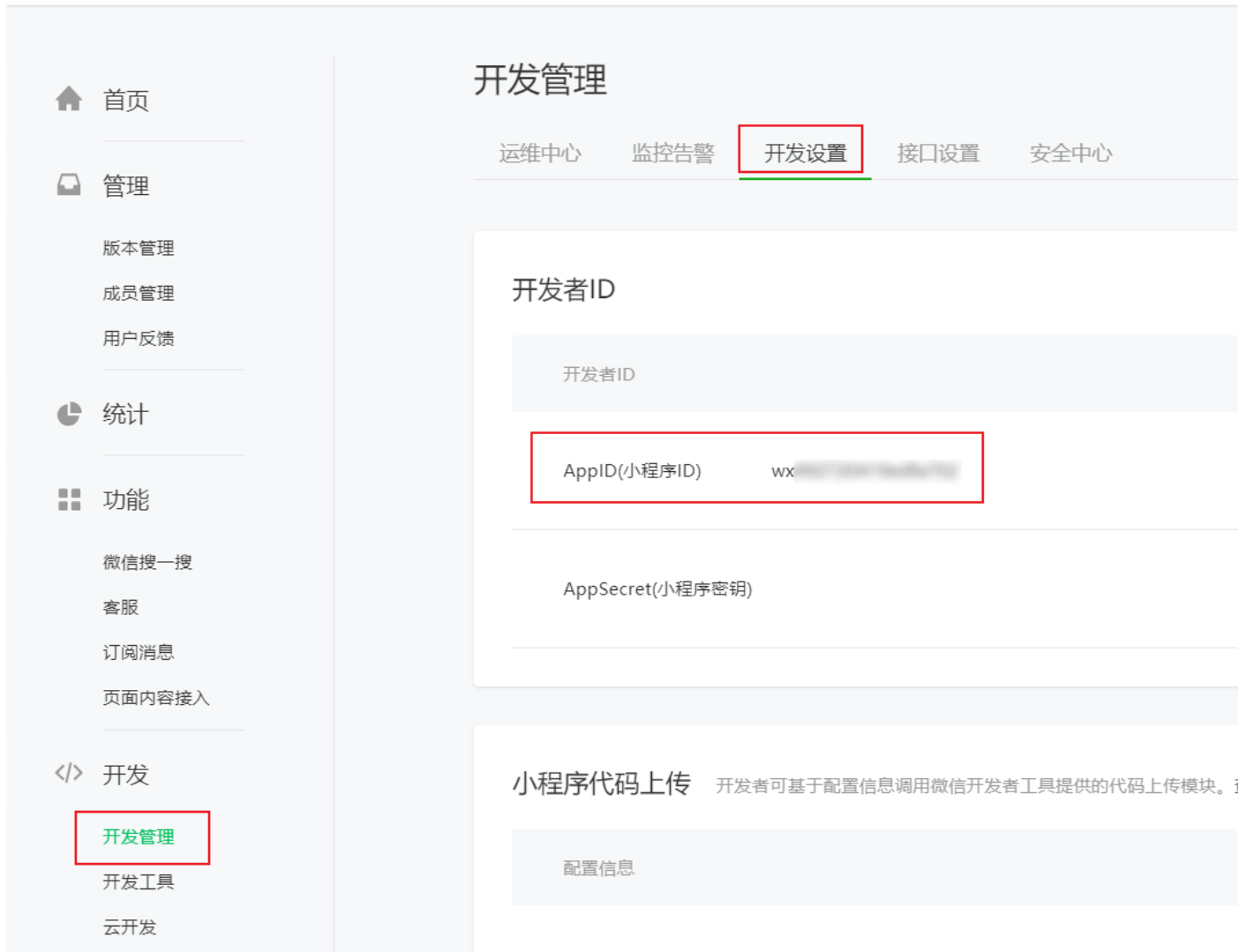
开发中



步骤3：注册、配置小程序

1. 前往 [微信公众平台](#) 注册小程序。
2. 登录微信公众平台的小程序后台，选择【开发管理】>【开发设置】。
3. 记录开发者 ID 中显示的【AppID（小程序 ID）】。

小程序



4. 在服务器域名中单击【开始配置】（若曾经配置过则单击【修改】），根据页面指引完成身份确认。

服务器域名

尚未配置服务器信息，查看 [小程序域名介绍](#)
使用官方推出的 [小程序云开发](#)，无需服务器及域名配置即可上线小程序，立即开通 [云开发](#)
如需购买服务器资源及域名，可 [前往腾讯云购买](#)。

开始配置

5. 填写服务器信息。

配置服务器信息



① 身份认证 — ② 配置服务器信息

如需购买服务器资源及域名，可前往[腾讯云](#)购买。或开通[云开发](#)，无需服务器及域名配置即可上线小程序。

request合法域名	https://iot.cloud.tencent.com;
socket合法域名	wss://iot.cloud.tencent.com;

- **request 合法域名**：https://iot.cloud.tencent.com
- **socket 合法域名**：wss://iot.cloud.tencent.com

6. 单击【保存并提交】。

⚠ 注意：

请务必进行配置！小程序只能与服务器域名列表中指定的域名进行网络通信。若未配置小程序的服务器域名，则小程序不能正常连接到物联网开发平台。在真机预览时若遇到接口出错，请确认此步骤域名是否配置正确。

步骤4：下载、配置 Demo小程序

1. 前往 [qcloud-iotexplorer-appdev-miniprogram-sdk-demo](#)，选择【Code】>【Download ZIP】，下载 Demo小程序代码到本地并解压。
2. 在 demo/miniprogram/app.js 文件中填写 [步骤2](#) 获得的 AppKey。

```
const APP_KEY = 'YOUR_APP_KEY_HERE'; // 填写 AppKey
```

3. 在 demo/cloudfunctions/login/index.js 文件中填写 [步骤2](#) 获得的 AppKey 与 AppSecret。

```
const APP_KEY = 'YOUR_APP_KEY_HERE'; // 填写 AppKey
const APP_SECRET = 'YOUR_APP_SECRET_HERE'; // 填写 AppSecret
```

4. 打开命令行，切换到 Demo 代码中的 demo/miniprogram 目录。
5. 执行以下命令安装依赖。

```
npm install
```

6. 运行微信开发者工具，在项目列表中单击【+】，进入创建小程序页面，填写项目信息。



- **项目名称：**可自行填写。
- **目录：**选择 Demo 代码中的 demo 目录。
- **AppID：**填写 [步骤3](#) 获取的 AppID。

7. 单击界面右下角的【新建】，项目创建完成后，进入微信开发者工具的主界面。选择菜单栏的【工具】>【构建 npm】，构建成功后界面提示完成构建。



8. 选择菜单栏的【项目】>【重新打开此项目】，以加载上一步构建的 npm 依赖。

步骤5：开通云开发并部署云函数

本步骤指导您使用小程序云开发部署登录接口，以支持小程序登录物联网开发平台。关于小程序登录物联网开发平台的更多信息，详情请参见 [接入微信登录](#)。

说明：

小程序登录物联网开发平台，需要在小程序中获取用户基本信息，并由开发者的后台服务器调用应用端 API 获取 AccessToken。关于应用端 API，请参见 [应用端 API 简介](#) 及 [微信号注册登录](#)。关于通过小程序 SDK 登录物联网开发平台，请参见小程序 SDK > [SDK 初始化](#)。

1. 单击微信开发者工具主界面上方的【云开发】，进入开通小程序云开发页面。
2. 单击【开通】，进入创建环境页面。

3. 填写云开发环境名称及环境 ID，单击【确定】。

创建环境

环境名称

一个具体的环境名称有助于区分和记忆。

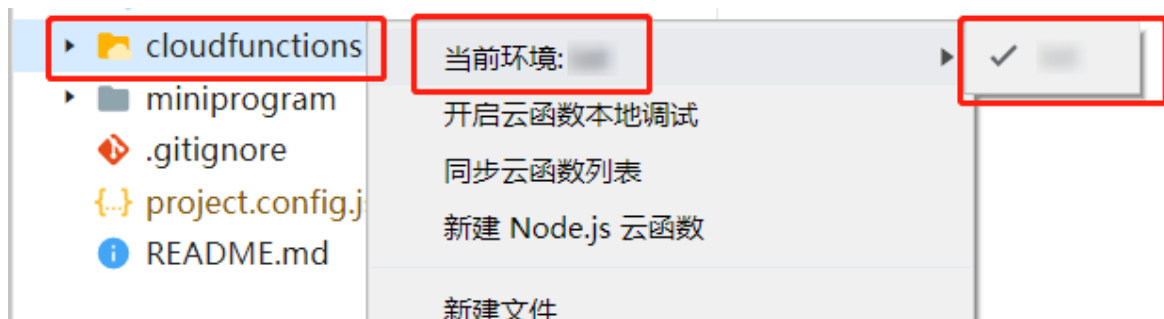
环境 ID

环境 ID 是在使用云服务时需要用到的全局唯一标识符，一经创建便不可修改。

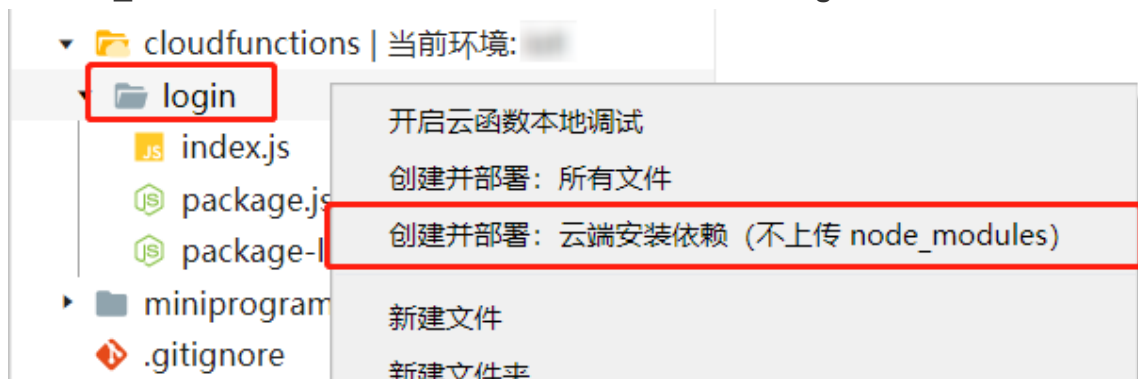
4. 云开发环境创建完成后，自动进入云开发控制台页面。关闭云开发控制台窗口，回到微信开发者工具的主界面。

5. 单击微信开发者工具主界面上方的【编辑器】，打开文件编辑器。

6. 右键单击文件列表中的【cloudfunctions】，选择【当前环境】，然后选择上述步骤中创建的云开发环境。



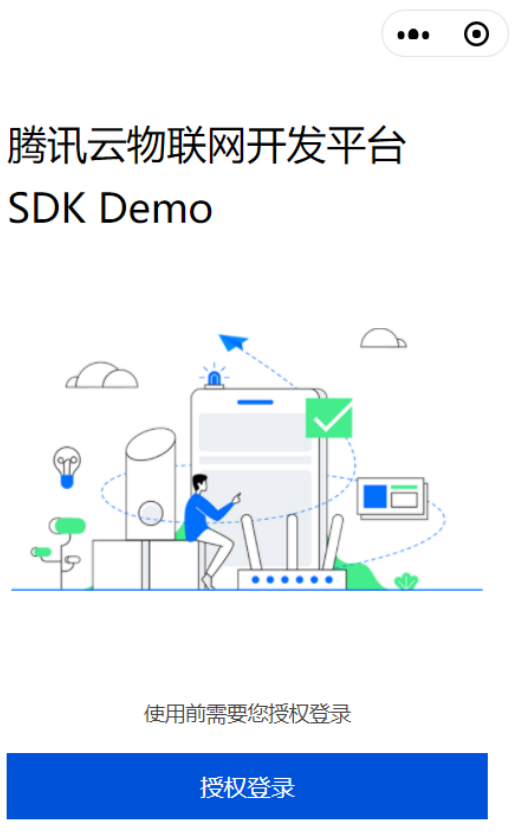
7. 展开文件列表中的【cloudfunctions】>【login】，右键单击【login】，选择【创建并部署：云端安装依赖（不上传 node_modules）】，上传完成后界面右上角提示上传云函数 login 成功。



步骤6：编译、运行小程序

1. 在微信开发者工具的主界面，单击界面上方的【编译】，编译完成后小程序在模拟器中运行（或单击界面上方的【预览】，编译完成后小程序在真机中运行）。

2. 小程序启动后，进入登录页面。单击【授权登录】>【允许】，登录完成后进入设备列表页面。

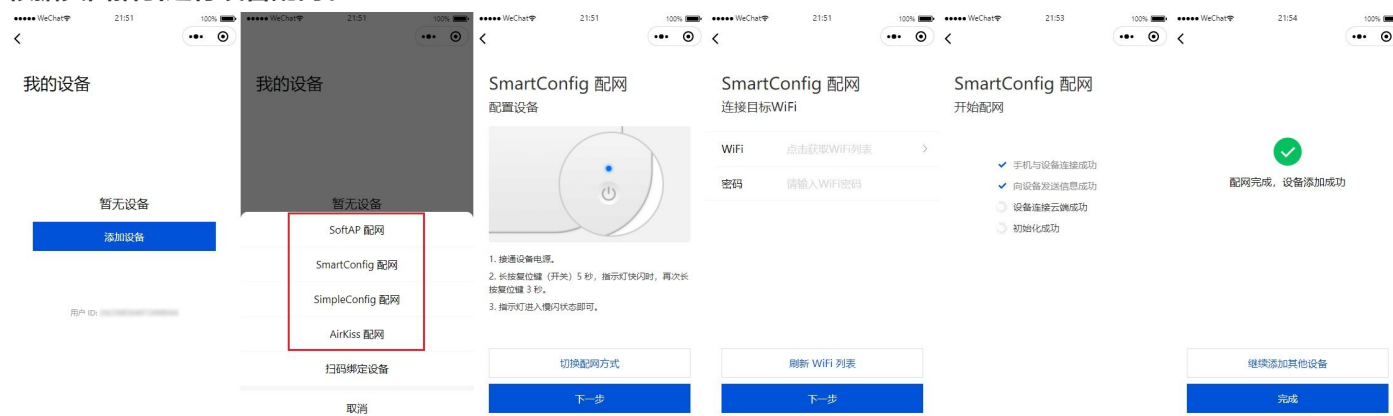


步骤7：通过小程序绑定设备

Wi-Fi 配网绑定设备

Wi-Fi 配网绑定设备需要配合真实设备（如 ESP8266 模组）进行操作。关于配网协议与配网流程，详情请参见 [配网开发概述](#)。在 Demo 小程序中进行 Wi-Fi 配网的步骤如下：

1. 在小程序的设备列表页面，选择【添加设备】>【SmartConfig 配网】或【SoftAP 配网】。
2. 根据页面指引进行设备配网。



扫描二维码绑定设备

🔍 说明：

物联网开发平台提供的设备二维码可以用于快速绑定真实设备，帮助开发者降低开发难度。量产后为了安全性，将会关闭二维码入口。

1. 登录腾讯云 [物联网开发平台控制台](#)，选择 [步骤1](#) 中创建的项目，进入项目管理页面。
2. 单击【产品开发】，选择 [步骤1](#) 中创建的产品，进入产品详情页面。
3. 单击页面上方的【设备调试】，进入设备调试页面。根据设备类型，按照以下步骤获取设备二维码。

。 **真实设备**：在设备列表中单击【二维码】，页面展示设备二维码。

物联网开发平台
小程序Demo

开发中心

产品开发

应用开发

数据开发

服务中心

固件升级

量产管理

产品开发 / 智能灯

数据模板 > 设备开发 > 交互开发 > **设备调试** > 批量投产

设备调试提供真实、虚拟设备调试功能，便于测试设备上报、接收数据是否正常，可创建测试设备后进行调试，开发中产品最多创建50个设备

新建设备 虚拟设备调试

设备名称 状态 激活时间 最后上线时间 操作

DemoLight	在线	2020-07-16 15:17:45	2020-07-29 08:30:00	调试 二维码 删除

。 **虚拟设备**：单击【虚拟设备调试】，进入虚拟设备调试页面，页面展示虚拟设备调试二维码。

物联网开发平台
小程序Demo

开发中心

产品开发

应用开发

数据开发

服务中心

固件升级

产品开发 / 智能灯

数据模板 > 设备开发 > 交互开发 > **设备调试** > 批量投产

设备调试提供真实、虚拟设备调试功能，便于测试设备上报、接收数据是否正常，可创建测试设备后进行调试，开发中产品最多创建50个设备

新建设备 **虚拟设备调试**

设备名称 状态 激活时间 最后上线时间

DemoLight	在线	2020-07-16 15:17:45	2020-08-04 02:05:06

4. 在小程序的设备列表页面，选择【添加设备】>【扫码绑定设备】，进入扫码页面。



5. 扫描控制台页面展示的二维码。扫码成功后小程序提示绑定设备成功。单击【确定】回到设备列表页面，设备列表中显示已绑定的设备。

步骤8：通过小程序控制设备

1. 小程序进入设备列表页面，单击要控制的设备，进入设备操控页面。



2. 单击设备属性列表中列出的属性，可以修改对应的设备属性。



智能灯 在线

开关	☐
颜色	Red >
亮度	1% >

检查固件更新

分享设备

删除设备

说明：

物联网开发平台提供了真实设备在线调试及虚拟设备调试功能，可通过控制台查询设备上报的当前数据、历史通信日志、事件及上下线记录等。具体操作请参见 [设备调试](#)。

开发指南

最近更新时间：2021-05-11 09:06:51

本文介绍开发者如何基于物联网开发平台小程序 SDK（下称 SDK）构建自主品牌小程序，通过 SDK 使用平台提供的能力。

前提条件

1. 使用 SDK 需要您的运行环境包含 Node.js 以及 npm。
2. 登录 [物联网开发平台控制台](#) 创建项目及产品，具体操作请参见 [产品定义](#)。
3. 登录 [物联网开发平台控制台](#) 获取 AppKey 和 AppSecret，具体操作请参见 [获取应用 AppKey 和 AppSecret](#)。

接入 SDK

安装 SDK

1. 在微信小程序目录下，通过 npm [安装 SDK](#)。

```
npm install qcloud-iotexplorer-appdev-sdk
```

2. 在 [微信开发者工具](#) 的项目界面中，单击界面右上角的【详情】，选择【本地设置】，勾选【使用 npm 模块】。



3. 在微信开发者工具的项目界面中，选择菜单栏的【工具】>【构建 npm】，构建成功后界面提示完成构建。



配置小程序服务器域名

小程序 SDK 通过以下域名连接到物联网开发平台。

- request 域名: <https://iot.cloud.tencent.com>
- socket 域名: <wss://iot.cloud.tencent.com>

在接入小程序 SDK 时，需要将上述域名添加到小程序的服务器域名列表中，步骤如下。

1. 登录 [小程序后台](#)。

2. 选择【开发管理】>【开发设置】。



3. 在“服务器域名”页面单击【开始配置】（若曾经配置过则单击【修改】），根据页面指引完成身份确认。



4. 填写服务器信息。

配置服务器信息



① 身份认证 — ② 配置服务器信息

如需购买服务器资源及域名，可 [前往腾讯云购买](#)。或开通 [云开发](#)，无需服务器及域名配置即可上线小程序。

request合法域名	https://iot.cloud.tencent.com;
socket合法域名	wss://iot.cloud.tencent.com;

- **request 合法域名**：https://iot.cloud.tencent.com
- **socket 合法域名**：wss://iot.cloud.tencent.com

5. 单击【保存并提交】。

接入微信登录

使用物联网开发平台的 [微信号注册登录](#) 应用端 API，可以让小程序用户通过微信注册登录到物联网开发平台。出于安全考虑，不建议在小程序端直接调用微信号注册登录 API，请在自建的后台服务端调用该 API，以避免密钥的泄漏。

小程序接入微信登录需要实现以下流程：

1. 小程序

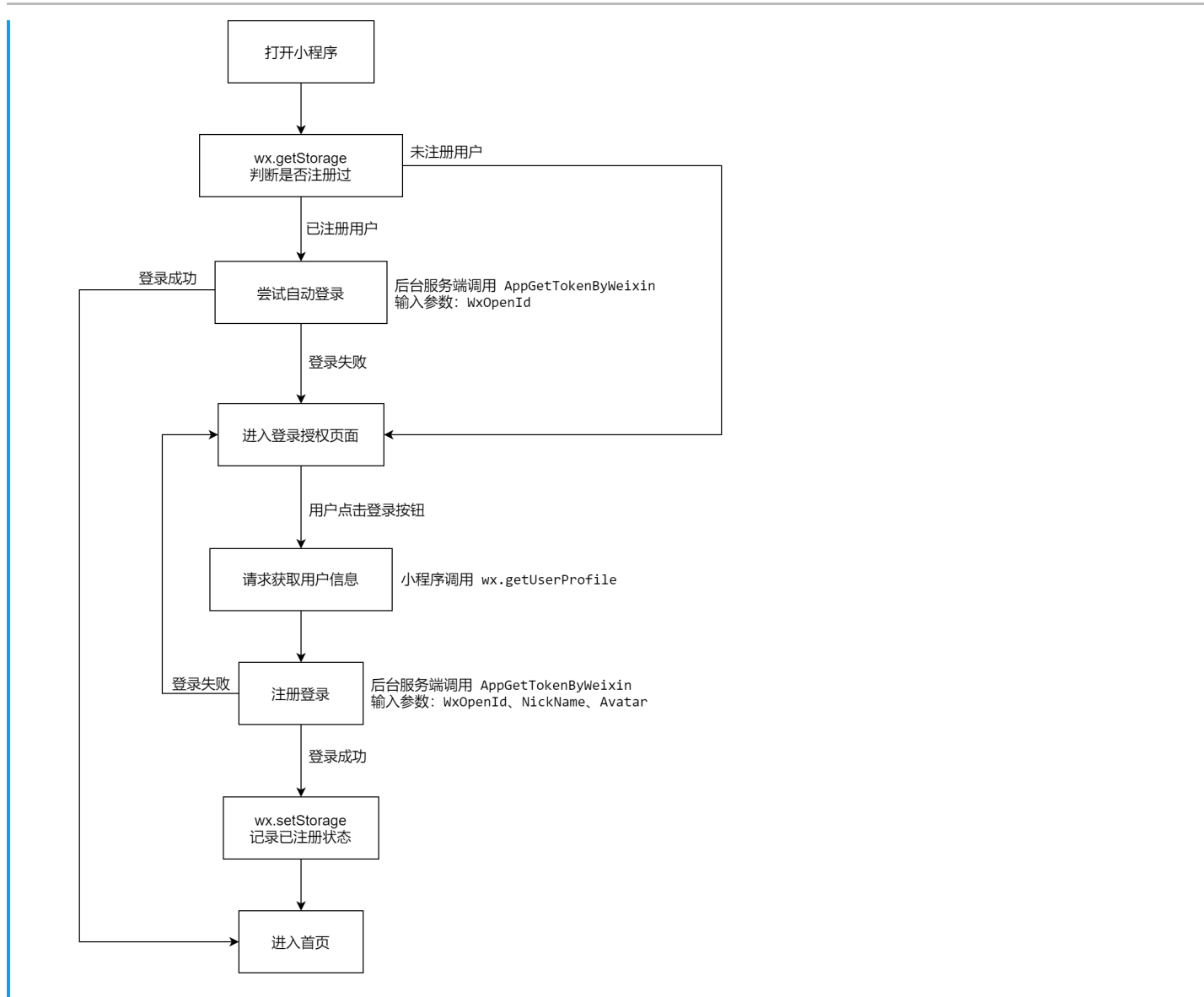
- 小程序调用 [wx.login](#) 获取临时登录凭证 code，并调用 [wx.getUserProfile](#) 获取用户昵称和头像。
- 小程序凭上述信息，调用自建的后台服务端的登录接口以获取 AccessToken。
- 小程序收到后台服务端返回的 AccessToken，传入 SDK 以完成 SDK 的初始化。

2. 后台服务端登录接口

- 调用微信的 [auth.code2Session](#)，通过临时登录凭证 code 获取小程序用户的 OpenID。
- 调用物联网开发平台的 [微信号注册登录](#) 应用端 API，通过小程序用户的 OpenID 获取 AccessToken。
- 将 AccessToken 返回给小程序。

② 说明：

小程序 API [wx.getUserProfile](#) 每次请求均会弹出授权窗口。对于已注册用户，在调用物联网开发平台的 [微信号注册登录](#) 应用端 API 进行登录时，可不传入用户昵称和头像。小程序可参考以下登录流程进行实现。



示例代码

请按照以下步骤并参考示例代码以实现接入微信登录：

1. 在 AppDevSdk 的构造函数参数中，将 appKey 的值修改为您实际获取的 AppKey。
2. 补充 getAccessToken 函数的实现。在 getAccessToken 函数中，您需要调用自建的后台服务端搭建的登录接口，传入登录所需的信息，并将获取到的 AccessToken 作为返回值。

```
// app.js
const { AppDevSdk } = require('qcloud-iotexplorer-appdev-sdk');

App({
  onLaunch() {
    this.sdk = new AppDevSdk({
      appKey: '此处填写您的 AppKey',
      getAccessToken: () => this.getAccessToken()
    });
  }
});
```

```
});  
},  
  
async getAccessToken() {  
  // 小程序登录凭证 code ( 后台服务端凭 code 向微信后台换取 OpenID )  
  const code = await new Promise((resolve, reject) => {  
    wx.login({  
      success: (res) => resolve(res.code),  
      fail: reject  
    });  
  });  
  
  // 用户昵称和头像 ( 通过 wx.getUserProfile 获取 )  
  const userInfo = this.globalData.userInfo;  
  
  // 调用开发者自建的后台服务端获取物联网开发平台的 AccessToken  
  // 请根据实际情况调整实现  
  const resp = await new Promise((resolve, reject) => {  
    wx.request({  
      url: '开发者自建的后台服务端 URL',  
      method: 'POST',  
      data: {  
        code: code,  
        nickName: userInfo.nickName,  
        avatarUrl: userInfo.avatarUrl  
      },  
      header: {  
        'content-type': 'application/json'  
      },  
      success(res) {  
        resolve(res.data);  
      },  
      fail(err) {  
        reject(err);  
      }  
    });  
  });  
  
  return {  
    Token: resp.Token,  
    ExpireAt: resp.ExpireAt,  
  };  
}
```

```
},

globalData: {
  userInfo: {
    nickName: '用户昵称',
    avatarUrl: ''
  }
}

})

// pages/index/index.js
Page({
  onTapLoginButton() {
    // wx.getUserProfile 需要在点击事件中请求
    wx.getUserProfile({
      desc: '授权获取昵称和头像以登录',
      success(res) {
        const app = getApp();
        // 将获取到的用户信息放入 globalData
        app.globalData.userInfo = res.userInfo;

        // 获取到用户信息后，初始化 SDK
        // sdk.init() 会调用 getAccessToken
        app.sdk.init().then(() => {
          // 初始化完成
        });
      },
      fail(err) {
        console.error(err);
      }
    });
  }
})
```

使用 SDK

调用应用端 API

应用端 API 是物联网开发平台为了满足智能家居场景，为用户开发自有品牌的小程序或 App 而提供的云端服务，用户无需实现用户管理、设备管理、设备定时、家庭管理等基础能力，可通过调用应用端 API 快速完成移动应用端的开发。关于应用端 API 的更多信息，请参见 [应用端 API 简介](#)。

SDK 对应用端 API 的调用过程进行了封装，发送请求时会自动带上公共参数 `AccessToken` 与 `RequestId`。

以下示例代码以调用 [获取用户绑定设备列表](#) 应用端 API 为例。

```
sdk.requestApi(  
  'AppGetFamilyDeviceList', // 请求应用端 API 的 Action 名  
  { FamilyId: 'default' } // 请求应用端 API 的数据  
)  
.then(data => {  
  // 请求成功  
  console.log(data);  
})  
.catch(err => {  
  // 请求失败  
  console.error(err);  
});
```

设备配网

SDK 目前支持 softAP 和 SmartConfig 和 simpleConfig 和 AirKiss 方式进行设备配网。

softAP 配网

关于 softAP 方式配网的流程，详情请参见 [softAP 配网开发](#)。具体参数说明和流程说明可参见小程序 SDK 章节中的 [设备配网](#) 部分。

```
const WifiConfConstants = require('qcloud-iotexplorer-appdev-plugin-wificonf-core').constants;  
  
const {  
  // 错误的中文描述  
  WifiConfErrorMsg,  
  // 步骤code  
  WifiConfStepCode,  
  // 步骤code的中文描述  
  WifiConfStepDesp  
} = WifiConfConstants;  
  
/**  
 * softap配网  
 */  
function SoftApConfigure({  
  token,  
  wifiInfo = {
```

```
SSID: "";
password: "";
BSSID: "";
},
familyId = 'default',
roomId,
reporter,
onStepChange,
onStatusChange,
}) {
const onProgress = (data) => {
reporter.info(data.code, data.detail);

switch (data.code) {
case WifiConfStepCode.CREATE_UDP_CONNECTION_SUCCESS:
onStepChange(1);
break;

case WifiConfStepCode.PROTOCOL_SUCCESS:
onStepChange(2);
break;
case WifiConfStepCode.SOFTAP_GET_DEVICE_SIGNATURE_SUCCESS:
case WifiConfStepCode.BUSINESS_QUERY_TOKEN_STATE_SUCCESS:
onStepChange(3);
break;
case WifiConfStepCode.WIFI_CONF_SUCCESS:
onStepChange(4);
break;
}
};

const onComplete = ({ productId, deviceName }) => {
onStatusChange({
status: 'success',
productId,
deviceName,
});
};

const onError = async ({ code, detail }) => {
reporter.error(code, detail);
```

```
onStatusChange({ status: 'error' });
};

sdk.plugins['wifiConfSoftAp'].start({
  wifiConfToken: token,
  targetWifiInfo: wifiInfo,
  autoRetry: true, // 自动处理故障流程
  familyId,
  roomId,
  onProgress,
  onComplete,
  onError
});
}

module.exports = SoftApConfigure;
```

SmartConfig 配网

关于 SmartConfig 方式配网的流程，详情请参见 [SmartConfig 配网开发](#)。具体参数说明和流程说明可参见小程序 SDK 章节中的 [设备配网](#) 部分。

```
const WifiConfConstants = require('qcloud-iotexplorer-appdev-plugin-wificonf-core').constants;

const {
  // 错误的中文描述
  WifiConfErrorMsg,
  // 步骤code
  WifiConfStepCode,
  // 步骤code的中文描述
  WifiConfStepDesp
} = WifiConfConstants;

/**
 * smartconfig一键配网
 */
function SmartConfigConfigure({
  token,
  wifiInfo = {
    SSID: "",
    password: "",
    BSSID: "";
```

```
},
familyId = 'default',
roomId,
reporter,
onStepChange,
onStatusChange,
}) {
const onProgress = (data) => {
reporter.info(data.code, data.detail);

switch (data.code) {
case WifiConfStepCode.PROTOCOL_SUCCESS:
onStepChange(1);
break;
case WifiConfStepCode.CREATE_UDP_CONNECTION_SUCCESS:
onStepChange(2);
break;
case WifiConfStepCode.BUSINESS_QUERY_TOKEN_STATE_SUCCESS:
onStepChange(3);
break;
case WifiConfStepCode.WIFI_CONF_SUCCESS:
onStepChange(4);
break;
}
};

const onComplete = ({ productId, deviceName }) => {
onStatusChange({
status: 'success',
productId,
deviceName,
});
};

const onError = async ({ code, detail }) => {
reporter.error(code, detail);

onStatusChange({ status: 'error' });
};

sdk.plugins['wifiConfSmartConfig'].start({
wifiConfToken: token,
```

```
targetWifiInfo: wifiInfo,
autoRetry: true, // 自动处理故障流程
familyId,
roomId,
onProgress,
onComplete,
onError
});
}

module.exports = SmartConfigConfigure;
```

simpleConfig 配网

关于 simpleConfig 方式配网的流程，详情请参见 [simpleConfig 配网开发](#)。具体参数说明和流程说明可参见小程序 SDK 章节中的 [设备配网](#) 部分。

```
const WifiConfConstants = require('qcloud-iotexplorer-appdev-plugin-wificonf-core').constants;

const {
  // 错误的中文描述
  WifiConfErrorMsg,
  // 步骤code
  WifiConfStepCode,
  // 步骤code的中文描述
  WifiConfStepDesp
} = WifiConfConstants;

/**
 * SimpleConfig一键配网
 */
function SimpleConfigConfigure({
  token,
  wifiInfo = {
    SSID: '',
    password: '',
  },
  familyId = 'default',
  roomId,
  reporter,
  onStepChange,
  onStatusChange,
```

```
}) {  
  const onProgress = (data) => {  
    reporter.info(data.code, data.detail);  
  
    switch (data.code) {  
      case WifiConfStepCode.PROTOCOL_SUCCESS:  
        onStepChange(1);  
        break;  
      case WifiConfStepCode.CREATE_UDP_CONNECTION_SUCCESS:  
        onStepChange(2);  
        break;  
      case WifiConfStepCode.BUSINESS_QUERY_TOKEN_STATE_SUCCESS:  
        onStepChange(3);  
        break;  
      case WifiConfStepCode.WIFI_CONF_SUCCESS:  
        onStepChange(4);  
        break;  
    }  
  };  
  
  const onComplete = ({ productId, deviceName }) => {  
    onStatusChange({  
      status: 'success',  
      productId,  
      deviceName,  
    });  
  };  
  
  const onError = async ({ code, detail }) => {  
    reporter.error(code, detail);  
  
    onStatusChange({ status: 'error' });  
  };  
  
  sdk.plugins['wifiConfSimpleConfig'].start({  
    wifiConfToken: token,  
    targetWifiInfo: wifiInfo,  
    autoRetry: true, // 自动处理故障流程  
    familyId,  
    roomId,  
    onProgress,  
    onComplete,  
  });  
}
```

```
onError
});
}

module.exports = SimpleConfigConfigure;
```

AirKiss 配网

关于 AirKiss 方式配网的流程，详情请参见 [AirKiss 配网开发](#)。具体参数说明和流程说明可参见小程序 SDK 章节中的 [设备配网](#) 部分。

```
const WifiConfConstants = require('qcloud-iotexplorer-appdev-plugin-wificonf-core').constants;

const {
  // 错误的中文描述
  WifiConfErrorMsg,
  // 步骤code
  WifiConfStepCode,
  // 步骤code的中文描述
  WifiConfStepDesp
} = WifiConfConstants;

/**
 * AirKiss一键配网
 */
function AirKissConfigure({
  token,
  wifiInfo = {
    SSID: '',
    password: '',
  },
  familyId = 'default',
  roomId,
  reporter,
  onStepChange,
  onStatusChange,
}) {
  const onProgress = (data) => {
    reporter.info(data.code, data.detail);

    switch (data.code) {
      case WifiConfStepCode.PROTOCOL_SUCCESS:
```

```
onStepChange(1);
break;
case WifiConfStepCode.CREATE_UDP_CONNECTION_SUCCESS:
onStepChange(2);
break;
case WifiConfStepCode.BUSINESS_QUERY_TOKEN_STATE_SUCCESS:
onStepChange(3);
break;
case WifiConfStepCode.WIFI_CONF_SUCCESS:
onStepChange(4);
break;
}
};

const onComplete = ({ productId, deviceName }) => {
onStatusChange({
status: 'success',
productId,
deviceName,
});
};

const onError = async ({ code, detail }) => {
reporter.error(code, detail);

onStatusChange({ status: 'error' });
};

sdk.plugins['wifiConfAirKiss'].start({
wifiConfToken: token,
targetWifiInfo: wifiInfo,
autoRetry: true, // 自动处理故障流程
familyId,
roomId,
onProgress,
onComplete,
onError
});
}

module.exports = AirKissConfigure;
```


控制设备

通过调用 [用户控制设备](#) 应用端 API，可以对用户已绑定的设备发起控制操作。

```
// 指定要控制的设备的属性数据
const deviceData = {
  light_switch: 0,
};
sdk.requestApi('AppControlDeviceData', {
  ProductId: '要控制的设备的产品 ID',
  DeviceName: '要控制的设备的设备名称',
  Data: JSON.stringify(deviceData) // 控制报文 JSON
})
.then(data => {
  // 请求成功
  console.log(data);
})
.catch(err => {
  // 请求失败
  console.error(err);
});
```

长连接通信能力

SDK 支持通过 WebSocket 为用户订阅所绑定设备的上报数据以及状态信息，SDK 初始化时默认自动连接 WebSocket 服务端。

通过设备 ID 列表订阅

```
sdk.subscribeDevices([
  'Product1/Device1',
  'Product1/Device2',
  'Product2/Device3'
]);
```

通过设备列表订阅

subscribeDevices 函数支持传入设备列表（如 [获取用户绑定设备列表](#) 应用端 API 返回的设备列表）以订阅设备信息，要求参数数组中的设备信息包含 DeviceId 字段。

```
sdk.requestApi('AppGetFamilyDeviceList', { FamilyId: 'default' })
.then(data => {
```

```
sdk.subscribeDevices(data.DeviceList);
})
.catch(err => {
  // 请求失败
  console.error(err);
});
```

监听设备上报数据及状态信息

调用 SDK 的 [监听事件](#) 接口，可以监听 WebSocket 订阅的设备上报数据及状态信息，请参照以下示例代码。

```
const { EventTypes } = require('qcloud-iotexplorer-appdev-sdk').AppDevSdk.constants;
// 监听设备上报数据推送
sdk.on(EventTypes.WsReport, ({ deviceId, deviceData }) => {
  console.log('websocket device report', deviceId, deviceData);
});
// 监听设备在线状态变更推送
sdk.on(EventTypes.WsStatusChange, ({ deviceId, deviceStatus }) => {
  console.log('websocket device status change', deviceId, deviceStatus);
});
// 监听设备控制推送
sdk.on(EventTypes.WsControl, ({ deviceId, deviceData }) => {
  console.log('websocket device control', deviceId, deviceData);
});
```

获取用户基本信息

通过读取 SDK 对象上的属性，可以获取用户的基本信息，请参照以下示例代码。

```
// sdk.isLogin: boolean; 用户是否登录
console.log(sdk.isLogin);
// sdk.userId: string; 用户 ID
console.log(sdk.userId);
// sdk.nickName: string; 用户昵称
console.log(sdk.nickName);
// sdk.userInfo: UserInfo; 用户详细信息
console.log(sdk.userInfo);
console.log(sdk.userInfo.Avatar);
// sdk.getDefaultFamilyId: () => Promise< string >; 获取用户默认的家庭 ID
sdk.getDefaultFamilyId().then(familyId => console.log(familyId));
```

UserInfo 数据结构说明

属性名	属性描述	类型
Avatar	头像	string
CountryCode	国家代码	string
Email	邮箱	string
NickName	昵称	string
PhoneNumber	电话号码	string
UserID	用户 ID	string

SDK Demo

- Demo 项目 GitHub 地址：[qcloud-iotexplorer-appdev-miniprogram-sdk-demo](#)。
- Demo 使用指引：请参见 [快速入门](#)。

小程序 SDK

最近更新时间：2021-10-12 16:29:08

SDK 初始化

SDK 初始化后，才能通过 SDK 使用物联网开发平台提供的云端能力。初始化 SDK 需要依次完成以下步骤：

1. 调用 AppDevSdk 的构造函数并传入配置项，创建 SDK 对象。
2. 调用 SDK 对象的 init 函数初始化 SDK。

SDK 对象构造函数

```
AppDevSdk(sdkOptions)
```

配置项

参数名	参数描述	类型	必填
getAccessToken	获取 accessToken 的回调，返回一个 Promise，其值为 微信号注册登录 应用端 API 的返回结果	Promise<{ Token:string, ExpireAt?: number }>	是
appKey	在 物联网开发平台控制台 > 应用开发 > 小程序开发中申请的 AppKey	string	是
debug	是否为调试模式，默认为：false，开启调试模式后会开启打印调试日志	boolean	否
wsConfig	WebSocket 的配置	WsOptions	否
plugins	导入到 SDK 的配网插件数组	AppDevPlugin[]	否
apiUrl	物联网开发平台的接口 URL，默认为： https://iot.cloud.tencent.com/api/exploreropen/，一般无需更改	string	否
defaultUin	未登录状态下的默认 uin，以及调试模式下的固定 uin，默认为：unknown，一般无需更改	string	否
reporter	SDK 运行日志的回调函数	(eventName: string, params: any) => void	否

getAccessToken 回调函数

SDK 初始化时，将调用 `getAccessToken` 回调函数以取得物联网开发平台的 `AccessToken`。开发者需要在 `getAccessToken` 函数中实现获取 `AccessToken` 的流程，请参见 [接入微信登录](#)。

WsOptions 数据结构

属性名	属性描述	类型	必填
<code>autoReconnect</code>	WebSocket 断开后是否自动连接，默认为：true，自动重连每两秒尝试一次	boolean	否
<code>disconnectWhenAppHide</code>	当 <code>App.onHide</code> 触发时，是否自动断开 WebSocket，默认为：true	boolean	否
<code>connectWhenAppShow</code>	当 <code>App.onShow</code> 触发时，是否自动连接 WebSocket，默认为：true	boolean	否
<code>url</code>	websocket 服务的 URL，默认为： <code>wss://iot.cloud.tencent.com/ws/explorer</code>	string	否
<code>heartbeatInterval</code>	心跳包的发送间隔，单位毫秒，默认为：60000	number	否

SDK 初始化

```
sdk.init(options) => Promise< void >
```

调用后将依次执行：

1. 登录（调用 `getAccessToken` 回调函数，取得平台的 `AccessToken`）。
2. 连接 WebSocket。

`init` 函数可同时多次调用（返回同一个缓存的 `Promise`）。若一次执行未完成或已执行成功，多次调用后拿到的会是同一个 `Promise`。若执行失败，则该缓存的 `Promise` 在 `reject` 之后会被释放，再次调用则将重新执行。

参数说明

参数名	参数描述	类型	必填
<code>reload</code>	是否清理缓存的 <code>Promise</code> 并重新执行，默认为 <code>false</code>	boolean	否

示例代码

```
const getAccessToken = async () => {  
  // 获取小程序登录凭证 (code)  
  const code = await new Promise((resolve, reject) => {  
    wx.login({
```

```
success: (res) => resolve(res.code),
fail: reject
});

});

// 获取小程序用户信息
const userInfo = await new Promise((resolve, reject) => {
  wx.getUserInfo({
    success: (res) => resolve(res.userInfo),
    fail: reject
  });
});

// 调用开发者自建的后台服务端获取 AccessToken
// 请根据实际情况调整实现
const resp = await new Promise((resolve, reject) => {
  wx.request({
    url: '开发者自建的后台服务端 URL',
    data: {
      code: code,
      userInfo: userInfo
    },
    header: {
      'content-type': 'application/json'
    },
    success(res) {
      resolve(res.data);
    },
    fail(err) {
      reject(err);
    }
  });
});

return {
  Token: resp.Token,
  ExpireAt: resp.ExpireAt,
};
};
```

```
const sdk = new AppDevSdk({
  appKey: '此处填写您的 AppKey',
  getAccessToken: getAccessToken
});

// SDK 初始化
sdk.init();
```

应用端 API

应用端 API 是物联网开发平台为了满足智能家居场景，为用户开发自有品牌的小程序或 App 而提供的云端服务，包括用户管理、设备管理、设备定时、家庭管理等基础能力。关于应用端 API 的更多信息，请参见 [应用端 API 简介](#)。

调用应用端 API

调用应用端 API 并获得响应数据。

```
sdk.requestApi(Action: string, payload?: object, options?: object) => Promise< response >
```

参数说明

参数名	参数描述	类型	必填
Action	请求应用端 API 的 Action 名	string	是
payload	请求应用端 API 的数据，会自动带上公共参数 AccessToken 与 RequestId	object	否
options	请求的选项，将透传给 wx.request	object	否

返回值

- 请求成功（code=0）：返回一个 resolved 的 Promise，其值为应用端 API 响应中的 Response 部分数据。
- 请求失败：返回一个 rejected 的 Promise，其值的数据结构为：{ code, msg, ...detail }。

示例代码

```
sdk.requestApi('AppGetFamilyDeviceList', { FamilyId: 'default' })
  .then(data => {
    // 请求成功
    console.log(data);
  })
  .catch(err => {
    // 请求失败
```

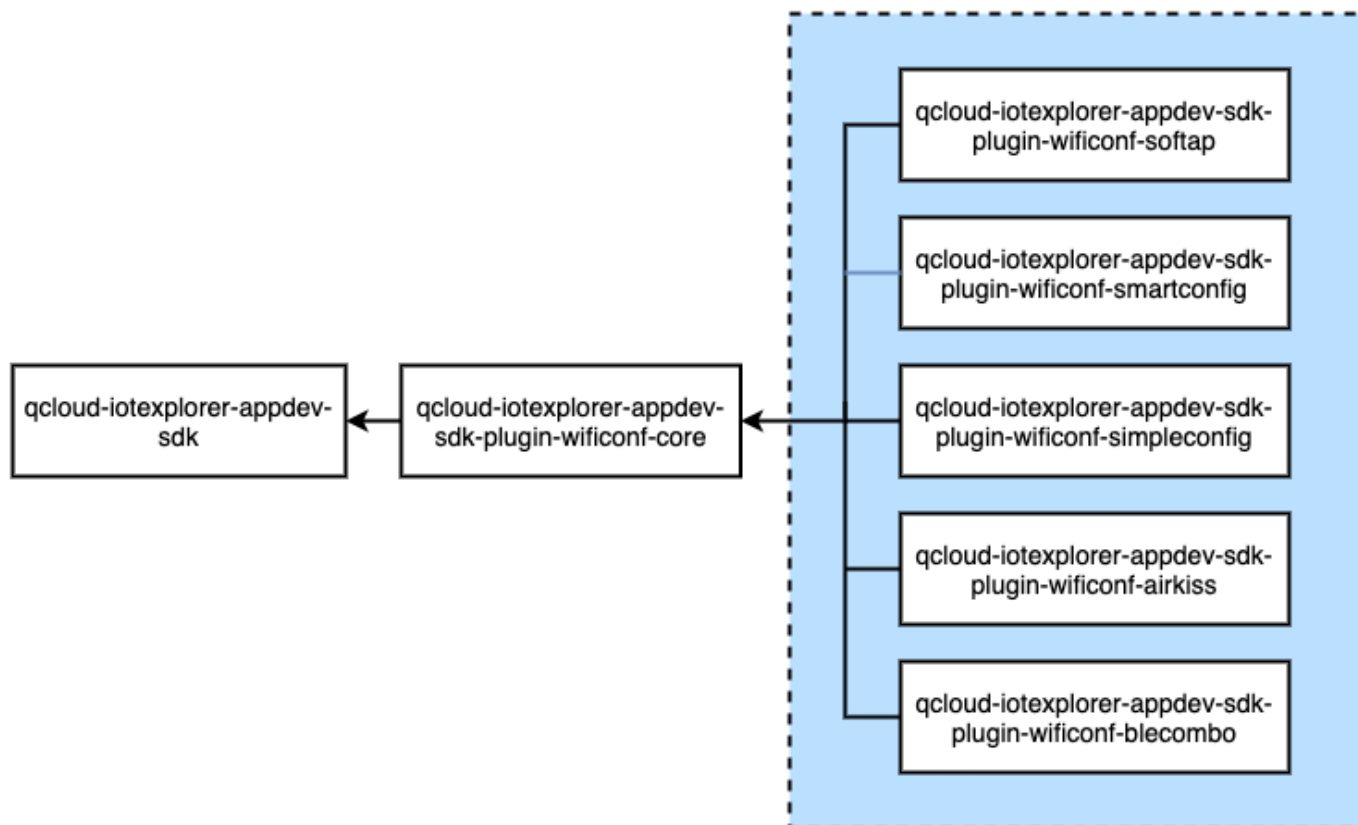
```
console.error(err);  
});
```

⚠ 注意:

- 腾讯云物联网开发平台是基于家庭的设备体系，每个家庭有其对应的 FamilyId，每台设备均归属一个家庭。
- 开发者也可以选择关注家庭这一概念，对所有需要传FamilyId的接口（例如 [获取用户绑定设备列表](#)）传入 default 作为 FamilyId，SDK 会自动完成内部的家庭相关的逻辑（SDK 会为用户创建一个默认家庭，若 FamilyId入参的值为default，SDK 会自动替换为用户默认家庭的FamilyId）。

设备配网

SDK 目前支持 softAP、SmartConfig、simpleConfig、AirKiss、BLE-Combo 这五种方式进行设备配网。这五种配网方式是以插件的方式按需引入的，这里为了方便大家理解，下图是设计思路，可以看出这5个插件的依赖关系



通过4步可以运行插件，以 SoftAp 为例，其余的配网方式步骤一样，后面关于配网步骤和参数说明会具体阐述。

1. 安装依赖。

```
npm install qcloud-iotexplorer-appdev-plugin-wificonf-core
```


2. 注册插件。

```
const SoftApPlug = require('qcloud-iotexplorer-appdev-plugin-wificonf-softap');
SoftApPlug.install(sdk);
```

3. 生成配网 Token，调用 [生成 Wi-Fi 设备配网 Token](#) 应用端 API 获取 Wi-Fi 设备配网 Token。

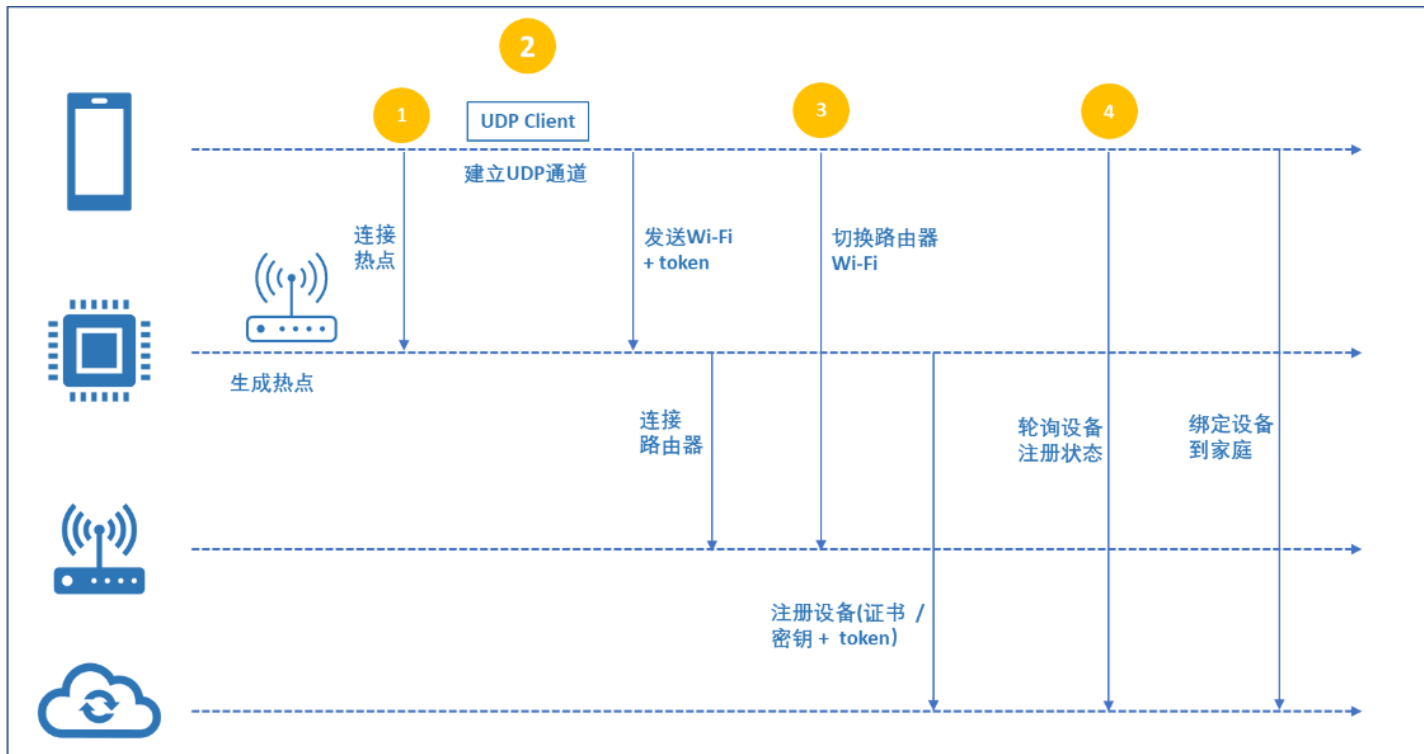
```
sdk.requestApi('AppCreateDeviceBindToken')
  .then(data => {
    const bindDeviceToken = data.Token;
  });
```

4. 运行插件，plugin 注册时的名称分别为：wifiConfSoftAp、wifiConfSmartConfig、wifiConfSmartConfig、wifiConfAirKiss、wifiConfBleCombo。

```
/**
 * - pluginNames: wifiConfSoftAp | wifiConfSmartConfig | wifiConfSimpleConfig | wifiConfAirKiss | wifiC
 * onfBleCombo
 */
sdk.plugins['wifiConfSoftAp'].start({
  wifiConfToken: bindDeviceToken,
  targetWifiInfo: wifiInfo,
  familyId,
  roomId,
  onProgress,
  onComplete,
  onError,
  autoRetry: true, // 自动处理故障流程
});
```

softAP 配网

关于 softAP 方式配网的流程，请参见 [softAP 配网开发](#)。各端交互的简易的流程如下：



腾讯连连中 softAP 配网页面如下，供开发者开发时参考。



softAPOpts 配网参数

参数名	参数描述	类型	必填
wifiConfToken	Wi-Fi 设备配网 Token，从后台接口 生成 Wi-Fi 设备配网 Token 获取	string	是
targetWifiInfo	目标 Wi-Fi 信息，需要设备去连接的 Wi-Fi 的信息	WifiInfo	是
softAPInfo	设备热点信息，如果传该配置，则首先会调用 wx.connectWifi 去连接设备热点；如果不传，则需要自行引导用户去连接设备热点	WifiInfo	否
familyId	家庭ID，默认为: 'default'，即用户默认家庭 ID	'default' \ string	是
roomId	房间ID，默认为: "，即用户默认房间 ID	" string	否

参数名	参数描述	类型	必填
onProgress	配网过程执行到每个步骤时触发的回调，回调中入参为当前步骤的详情 - code: 步骤代码，详见 配网步骤 小节 - msg: 步骤描述，自行从WifiConfStepDesp拿code取 - detail: 步骤详情，根据每个步骤不同而不同	{ code: WifiConfStepCode, detail?: object } => void	否
onError	配网失败时触发 - code: 错误代码，详见 错误码 小节 - msg: 错误描述，自行从WifiConfErrorMsg拿code取 - detail: 错误详情	{ code: WifiConfErrorCode, detail } => void	是
onComplete	配网完成后触发	() => void	是
udpAddress	连接上设备热点后，小程序发起 UDP 通信的地址，默认为: '192.168.4.1'，一般无需更改	string	否
udpPort	连接上设备热点后，小程序发起 UDP 通信的端口，默认为: 8266，一般无需更改	number	否
stepInterval	配网过程中，每一步中间等待的间隔，单位毫秒，默认为: 1000，一般无需更改	number	否
autoRetry	配网失败之后是否要启动自动错误处理后直接重试， 自动错误处理 章节，默认为: false	boolean	否

WifiInfo 数据结构

属性名	属性描述	类型	必填
SSID	Wi-Fi 的 SSID	string	是
password	Wi-Fi 的 密码	string	是

示例代码

```
const WifiConfConstants = require('qcloud-iotexplorer-appdev-plugin-wificonf-core').constants;

const {
  // 错误的中文描述
  WifiConfErrorMsg,
  // 步骤code
  WifiConfStepCode,
```

```
// 步骤code的中文描述
WifiConfStepDesp
} = WifiConfConstants;

/**
 * softap配网
 */
function SoftApConfigure({
  token,
  wifiInfo = {
    SSID: "",
    password: "",
    BSSID: "",
  },
  familyId = 'default',
  roomId,
  reporter,
  onStepChange,
  onStatusChange,
}) {
  const onProgress = (data) => {
    reporter.info(data.code, data.detail);

    switch (data.code) {
      case WifiConfStepCode.CREATE_UDP_CONNECTION_SUCCESS:
        onStepChange(1);
        break;

      case WifiConfStepCode.PROTOCOL_SUCCESS:
        onStepChange(2);
        break;
      case WifiConfStepCode.SOFTAP_GET_DEVICE_SIGNATURE_SUCCESS:
      case WifiConfStepCode.BUSINESS_QUERY_TOKEN_STATE_SUCCESS:
        onStepChange(3);
        break;
      case WifiConfStepCode.WIFI_CONF_SUCCESS:
        onStepChange(4);
        break;
    }
  };
};
```

```
const onComplete = ({ productId, deviceName }) => {
  onChange({
    status: 'success',
    productId,
    deviceName,
  });
};

const onError = async ({ code, detail }) => {
  reporter.error(code, detail);

  onChange({ status: 'error' });
};

sdk.plugins['wifiConfSoftAp'].start({
  wifiConfToken: token,
  targetWifiInfo: wifiInfo,
  autoRetry: true, // 自动处理故障流程
  familyId,
  roomId,
  onProgress,
  onComplete,
  onError
});

module.exports = SoftApConfigure;
```

配网步骤v1.0（已废弃，不建议使用）

步骤	描述
WifiConfStepCode.WIFI_CONF_START	开始配网
WifiConfStepCode.PROTOCOL_START	配网协议开始
WifiConfStepCode.CONNECT_SOFTAP_START	开始连接设备热点
WifiConfStepCode.CONNECT_SOFTAP_SUCCESS	连接设备热点成功
WifiConfStepCode.CREATE_UDP_CONNECTION_START	开始与设备建立 UDP 连接

步骤	描述
WifiConfStepCode.CREATE_UDP_CONNECTION_SUCCESS	与设备建立 UDP 连接成功
WifiConfStepCode.SOFTAP_SEND_TARGET_WIFIINFO_START	开始发送目标 Wi-Fi 信息
WifiConfStepCode.SOFTAP_SEND_TARGET_WIFIINFO_SUCCESS	发送目标 Wi-Fi 信息成功 detail: { response }, 收到设备的具体响应
WifiConfStepCode.SOFTAP_GET_DEVICE_SIGNATURE_START	开始获取设备签名
WifiConfStepCode.SOFTAP_GET_DEVICE_SIGNATURE_SUCCESS	获取设备签名成功 detail: { signature }
WifiConfStepCode.SOFTAP_RECONNECT_TARGET_WIFI_START	开始手机连接目标 Wi-Fi
WifiConfStepCode.SOFTAP_RECONNECT_TARGET_WIFI_SUCCESS	手机连接目标 Wi-Fi 成功
WifiConfStepCode.PROTOCOL_SUCCESS	配网协议成功
WifiConfStepCode.BUSINESS_START	业务流程开始
WifiConfStepCode.BUSINESS_ADD_DEVICE_START	开始添加设备
WifiConfStepCode.BUSINESS_ADD_DEVICE_SUCCESS	添加设备成功
WifiConfStepCode.BUSINESS_SUCCESS	业务流程成功 detail: { productId, deviceName }, 请求参数
WifiConfStepCode.WIFI_CONF_SUCCESS	配网成功

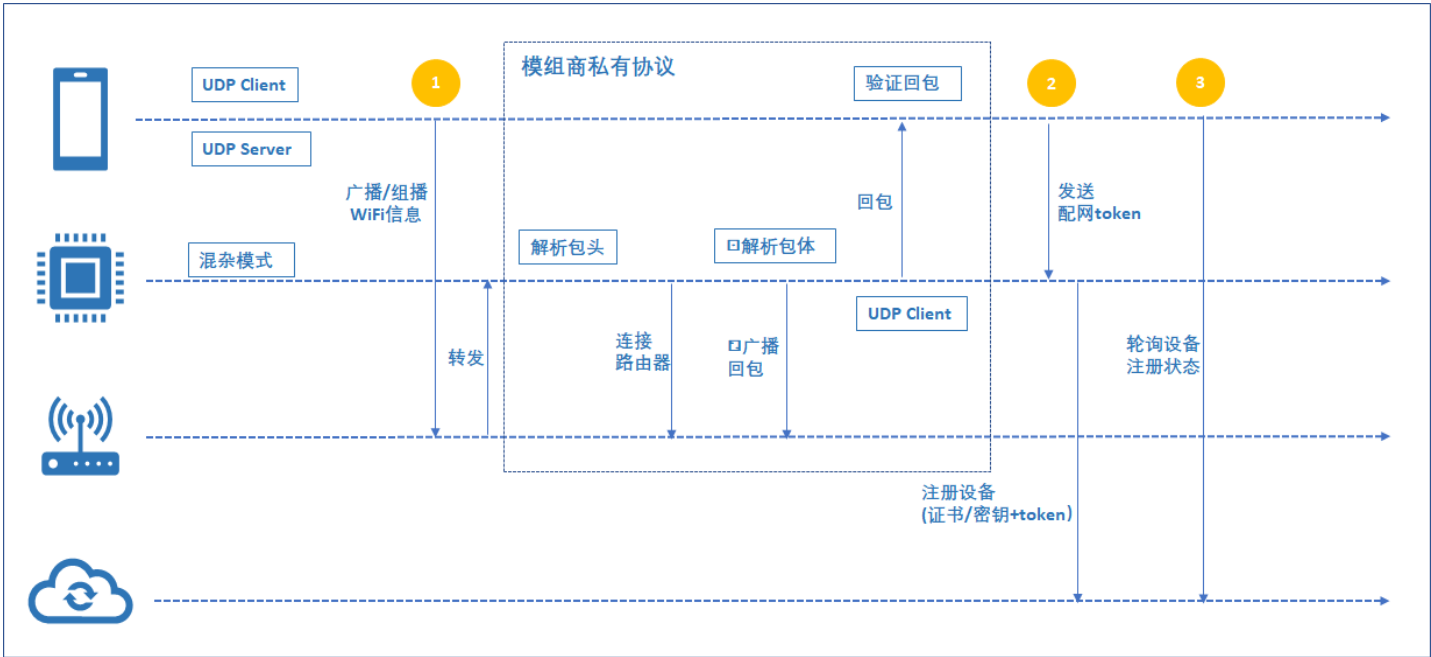
配网步骤v2.0

步骤	描述
WifiConfStepCode.WIFI_CONF_START	开始配网
WifiConfStepCode.PROTOCOL_START	配网协议开始
WifiConfStepCode.CONNECT_SOFTAP_START	开始连接设备热点
WifiConfStepCode.CONNECT_SOFTAP_SUCCESS	连接设备热点成功
WifiConfStepCode.CREATE_UDP_CONNECTION_START	开始与设备建立 UDP 连接
WifiConfStepCode.CREATE_UDP_CONNECTION_SUCCESS	与设备建立 UDP 连接成功
WifiConfStepCode.SOFTAP_SEND_TARGET_WIFIINFO_START	开始发送目标 Wi-Fi 信息

步骤	描述
WifiConfStepCode.SOFTAP_SEND_TARGET_WIFIINFO_SUCCESS	发送目标 Wi-Fi 信息成功 detail: { response }, 收到设备的具体响应
WifiConfStepCode.SOFTAP_RECONNECT_TARGET_WIFI_START	开始手机连接目标 Wi-Fi
WifiConfStepCode.SOFTAP_RECONNECT_TARGET_WIFI_SUCCESS	手机连接目标 Wi-Fi 成功
WifiConfStepCode.PROTOCOL_SUCCESS	配网协议成功
WifiConfStepCode.BUSINESS_START	业务流程开始
WifiConfStepCode.BUSINESS_QUERY_TOKEN_STATE_START	开始查询配网TOKEN状态
WifiConfStepCode.BUSINESS_QUERY_TOKEN_STATE_SUCCESS	查询配网TOKEN状态成功
WifiConfStepCode.BUSINESS_ADD_DEVICE_START	开始添加设备
WifiConfStepCode.BUSINESS_ADD_DEVICE_SUCCESS	添加设备成功
WifiConfStepCode.BUSINESS_SUCCESS	业务流程成功 detail: { productId, deviceName}, 请求参数
WifiConfStepCode.WIFI_CONF_SUCCESS	配网成功

SmartConfig 配网

关于 SmartConfig 方式配网的流程，请参见 [SmartConfig 配网开发](#)。一键配网的配网流程图文版本如下：



腾讯连连中一键配网的页面交互流程如下，也给出来作为参考。



smartConfigOpts 配网参数

参数名	参数描述	类型	必填
wifiConfToken	Wi-Fi 设备配网 Token，从后台接口 生成 Wi-Fi 设备配网 Token 获取	string	是
targetWifiInfo	目标 Wi-Fi 信息，需要设备去连接的 Wi-Fi 的信息	WifiInfo	是
familyId	家庭 ID，默认为: 'default'，即用户默认家庭 ID	'default' string	否
roomId	房间ID，默认为: "，即用户默认房间 ID	" string	否

参数名	参数描述	类型	必填
onProgress	- code: 步骤代码, 详见 配网步骤 小节 - msg: 步骤描述, 自行从 WifiConfStepDesp 拿 code 取 - detail: 步骤详情, 根据每个步骤不同而不同	({ code: WifiConfStepCode, detail?: object }) => void	否
onError	配网失败时触发 - code: 错误代码, 详见 错误码 小节 - msg: 错误描述, 自行从 WifiConfErrorMsg 拿 code 取 - detail: 错误详情	({ code: WifiConfErrorCode, msg: string, detail }) => void	是
onComplete	配网完成后触发	() => void	是
udpPort	小程序和设备连上同一个局域网之后, 小程序发起 UDP 通信的端口, 默认为: 8266, 一般无需更改	number	否
stepInterval	配网过程中, 每一步中间等待的间隔, 单位毫秒, 默认为: 1000, 一般无需更改	number	否
autoRetry	配网失败之后是否要启动自动错误处理后直接重试, 自动错误处理 章节, 默认为: false	boolean	否

WifiInfo 数据结构

属性名	属性描述	类型	必填
SSID	Wi-Fi 的 SSID	string	是
BSSID	Wi-Fi 的 BSSID	string	是
password	Wi-Fi 的 密码	string	是

示例代码

```
const WifiConfConstants = require('qcloud-iotexplorer-appdev-plugin-wificonf-core').constants;

const {
  // 错误的中文描述
  WifiConfErrorMsg,
  // 步骤code
  WifiConfStepCode,
  // 步骤code的中文描述
  WifiConfStepDesp
}
```

```
} = WifiConfConstants;

/**
 * smartconfig一键配网
 */
function SmartConfigConfigure({
  token,
  wifiInfo = {
    SSID: "",
    password: "",
    BSSID: "",
  },
  familyId = 'default',
  roomId,
  reporter,
  onStepChange,
  onStatusChange,
}) {
  const onProgress = (data) => {
    reporter.info(data.code, data.detail);

    switch (data.code) {
      case WifiConfStepCode.PROTOCOL_SUCCESS:
        onStepChange(1);
        break;
      case WifiConfStepCode.CREATE_UDP_CONNECTION_SUCCESS:
        onStepChange(2);
        break;
      case WifiConfStepCode.BUSINESS_QUERY_TOKEN_STATE_SUCCESS:
        onStepChange(3);
        break;
      case WifiConfStepCode.WIFI_CONF_SUCCESS:
        onStepChange(4);
        break;
    }
  };

  const onComplete = ({ productId, deviceName }) => {
    onStatusChange({
      status: 'success',
    });
  };
}
```

```
productId,
deviceName,
});
};

const onError = async ({ code, detail }) => {
  reporter.error(code, detail);

  onChange({ status: 'error' });
};

sdk.plugins['wifiConfSmartConfig'].start({
  wifiConfToken: token,
  targetWifiInfo: wifiInfo,
  autoRetry: true, // 自动处理故障流程
  familyId,
  roomId,
  onProgress,
  onComplete,
  onError
});
}

module.exports = SmartConfigConfigure;
```

配网步骤

`sdk.plugins['wifiConfSmartConfig'].start()` 配网过程中，每执行完一个步骤就会触发一次 `onProgress` 回调，入参为： `{ code, detail }` 形式。

步骤	描述
<code>WifiConfStepCode.WIFI_CONF_START</code>	开始配网
<code>WifiConfStepCode.PROTOCOL_START</code>	配网协议开始
<code>WifiConfStepCode.PROTOCOL_DETAIL</code>	配网协议的细节，详细日志
<code>WifiConfStepCode.PROTOCOL_SUCCESS</code>	配网协议成功，获取到设备地址 detail: { data: { address } } ，收到设备局域网地址，用于给设备发送信息

simpleConfig 配网

The diagram illustrates the M3UA protocol flow across four layers: UDP Client, UDP Server, 混合模式 (Hybrid Mode), and 模组商私有协议 (Vendor Private Protocol). The flow is divided into three numbered steps:

- Step 1:** The UDP Client sends "广播/组播 WiFi 信息" (Broadcast/Multicast WiFi Information) to the UDP Server. The UDP Server then sends "转发" (Forward) to the 混合模式 layer. The 混合模式 layer sends "解析包头" (Parse Packet Header) to the 模组商私有协议 layer. The 模组商私有协议 layer sends "连接路由器" (Connect Router) to the UDP Server. The UDP Server then sends "回包" (Return Packet) to the 混合模式 layer. The 混合模式 layer sends "口解析包体" (Interface Parse Packet Body) to the 模组商私有协议 layer. The 模组商私有协议 layer sends "验证回包" (Verify Return Packet) to the UDP Client.
- Step 2:** The UDP Client sends "发送配网 token" (Send Provisioning Token) to the UDP Server. The UDP Server then sends "注册设备 (证书/密钥+token)" (Register Device (Certificate/Key+token)) to the 混合模式 layer. The 混合模式 layer sends "轮询设备注册状态" (Poll Device Registration Status) to the 模组商私有协议 layer.
- Step 3:** The 模组商私有协议 layer sends "轮询设备注册状态" (Poll Device Registration Status) to the UDP Server. The UDP Server then sends "轮询设备注册状态" (Poll Device Registration Status) to the 混合模式 layer. The 混合模式 layer sends "轮询设备注册状态" (Poll Device Registration Status) to the 模组商私有协议 layer.

腾讯连连中一键配网的页面交互流程如下，也给出来作为参考。



simpleConfigOpts 配网参数

参数名	参数描述	类型	必填
wifiConfToken	Wi-Fi 设备配网 Token，从后台接口 生成 Wi-Fi 设备配网 Token 获取	string	是
targetWifiInfo	目标 Wi-Fi 信息，需要设备去连接的 Wi-Fi 的信息	WifiInfo	是
familyId	家庭 ID，默认为: 'default'，即用户默认家庭 ID	'default' string	否
roomId	房间ID，默认为: "，即用户默认房间 ID	" string	否

参数名	参数描述	类型	必填
onProgress	- code: 步骤代码, 详见 配网步骤 小节 - msg: 步骤描述, 自行从WifiConfStepDesp拿code取 - detail: 步骤详情, 根据每个步骤不同而不同	({ code: WifiConfStepCode, detail?: object }) => void	否
onError	配网失败时触发 - code: 错误代码, 详见 错误码 小节 - msg: 错误描述, 自行从WifiConfErrorMsg拿code取 - detail: 错误详情	({ code: WifiConfErrorCode, detail }) => void	是
onComplete	配网完成后触发	() => void	是
udpPort	小程序和设备连上同一个局域网之后, 小程序发起UDP 通信的端口, 默认为: 8266, 一般无需更改	number	否
stepInterval	配网过程中, 每一步中间等待的间隔, 单位毫秒, 默认为: 1000, 一般无需更改	number	否
autoRetry	配网失败之后是否要启动自动错误处理后直接重试, 自动错误处理 章节, 默认为:false	boolean	否

WifiInfo 数据结构

属性名	属性描述	类型	必填
SSID	Wi-Fi 的 SSID	string	是
password	Wi-Fi 的 密码	string	是

示例代码

```
const WifiConfConstants = require('qcloud-iotexplorer-appdev-plugin-wificonf-core').constants;

const {
  // 错误的中文描述
  WifiConfErrorMsg,
  // 步骤code
  WifiConfStepCode,
  // 步骤code的中文描述
  WifiConfStepDesp
} = WifiConfConstants;
```

```
/**
 * simpleConfig一键配网
 */
function SimpleConfigConfigure({
  token,
  wifiInfo = {
    SSID: "",
    password: "",
  },
  familyId = 'default',
  roomId,
  reporter,
  onStepChange,
  onStatusChange,
}) {
  const onProgress = (data) => {
    reporter.info(data.code, data.detail);

    switch (data.code) {
      case WifiConfStepCode.PROTOCOL_SUCCESS:
        onStepChange(1);
        break;
      case WifiConfStepCode.CREATE_UDP_CONNECTION_SUCCESS:
        onStepChange(2);
        break;
      case WifiConfStepCode.BUSINESS_QUERY_TOKEN_STATE_SUCCESS:
        onStepChange(3);
        break;
      case WifiConfStepCode.WIFI_CONF_SUCCESS:
        onStepChange(4);
        break;
    }
  };

  const onComplete = ({ productId, deviceName }) => {
    onStatusChange({
      status: 'success',
      productId,
      deviceName,
    });
  };
}
```



```
};

const onError = async ({ code, detail }) => {
  reporter.error(code, detail);

  onChange({ status: 'error' });
};

sdk.plugins['wifiConfSimpleConfig'].start({
  wifiConfToken: token,
  targetWifiInfo: wifiInfo,
  autoRetry: true, // 自动处理故障流程
  familyId,
  roomId,
  onProgress,
  onComplete,
  onError
});

module.exports = SimpleConfigConfigure;
```

配网步骤

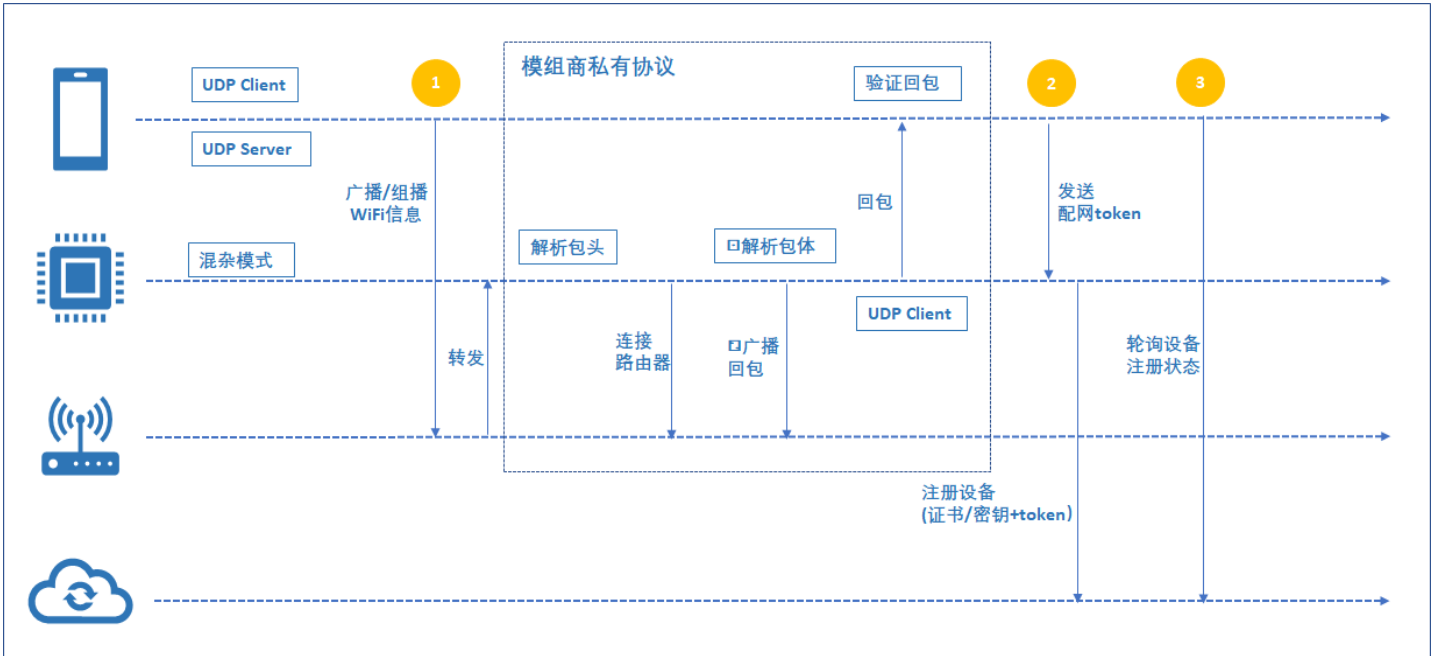
sdk.plugins['wifiConfSimpleConfig'].start() 配网过程中，每执行完一个步骤就会触发一次 onProgress 回调，入参为：{ code, detail } 形式。

步骤	描述
WifiConfStepCode.WIFI_CONF_START	开始配网
WifiConfStepCode.PROTOCOL_START	配网协议开始
WifiConfStepCode.PROTOCOL_DETAIL	配网协议的细节，详细日志
WifiConfStepCode.PROTOCOL_SUCCESS	配网协议成功，获取到设备地址 detail: { data: { address } }，收到设备局域网地址，用于给设备发送信息
WifiConfStepCode.BUSINESS_START	业务流程开始
WifiConfStepCode.BUSINESS_QUERY_TOKEN_STATE_START	开始查询配网TOKEN状态

步骤	描述
WifiConfStepCode.BUSINESS_QUERY_TOKEN_STATE_SUCCESS	查询配网TOKEN状态成功
WifiConfStepCode.BUSINESS_ADD_DEVICE_START	开始添加设备
WifiConfStepCode.BUSINESS_ADD_DEVICE_SUCCESS	添加设备成功
WifiConfStepCode.BUSINESS_SUCCESS	业务流程成功 detail: { productId, deviceName}, 请求参数
WifiConfStepCode.WIFI_CONF_SUCCESS	配网成功

AirKiss 配网

关于 AirKiss 方式配网的流程，请参见 [AirKiss 配网开发](#)。一键配网的配网流程图文版本如下：



腾讯连连中一键配网的页面交互流程如下，也给出来作为参考。



airKissOpts 配网参数

参数名	参数描述	类型	必填
wifiConfToken	Wi-Fi 设备配网 Token，从后台接口 生成 Wi-Fi 设备配网 Token 获取	string	是
targetWifiInfo	目标 Wi-Fi 信息，需要设备去连接的 Wi-Fi 的信息	WifiInfo	是
familyId	家庭 ID，默认为: 'default'，即用户默认家庭 ID	'default' string	否
roomId	房间ID，默认为: "，即用户默认房间 ID	" string	否

参数名	参数描述	类型	必填
onProgress	- code: 步骤代码, 详见 配网步骤 小节 - msg: 步骤描述, 自行从WifiConfStepDesp拿code取 - detail: 步骤详情, 根据每个步骤不同而不同	({ code: WifiConfStepCode, detail?: object }) => void	否
onError	配网失败时触发 - code: 错误代码, 详见 错误码 小节 - msg: 错误描述, 自行从WifiConfErrorMsg拿code取 - detail: 错误详情	({ code: WifiConfErrorCode, detail }) => void	是
onComplete	配网完成后触发	() => void	是
udpPort	小程序和设备连上同一个局域网之后, 小程序发起UDP 通信的端口, 默认为: 8266, 一般无需更改	number	否
stepInterval	配网过程中, 每一步中间等待的间隔, 单位毫秒, 默认为: 1000, 一般无需更改	number	否
autoRetry	配网失败之后是否要启动自动错误处理后直接重试, 自动错误处理 章节, 默认为:false	boolean	否

WifiInfo 数据结构

属性名	属性描述	类型	必填
SSID	Wi-Fi 的 SSID	string	是
password	Wi-Fi 的 密码	string	是

示例代码

```
const WifiConfConstants = require('qcloud-iotexplorer-appdev-plugin-wificonf-core').constants;

const {
  // 错误的中文描述
  WifiConfErrorMsg,
  // 步骤code
  WifiConfStepCode,
  // 步骤code的中文描述
  WifiConfStepDesp
} = WifiConfConstants;
```

```
/**
 * AirKiss一键配网
 */
function AirKissConfigure({
  token,
  wifiInfo = {
    SSID: "",
    password: "",
  },
  familyId = 'default',
  roomId,
  reporter,
  onStepChange,
  onStatusChange,
}) {
  const onProgress = (data) => {
    reporter.info(data.code, data.detail);

    switch (data.code) {
      case WifiConfStepCode.PROTOCOL_SUCCESS:
        onStepChange(1);
        break;
      case WifiConfStepCode.CREATE_UDP_CONNECTION_SUCCESS:
        onStepChange(2);
        break;
      case WifiConfStepCode.BUSINESS_QUERY_TOKEN_STATE_SUCCESS:
        onStepChange(3);
        break;
      case WifiConfStepCode.WIFI_CONF_SUCCESS:
        onStepChange(4);
        break;
    }
  };

  const onComplete = ({ productId, deviceName }) => {
    onStatusChange({
      status: 'success',
      productId,
      deviceName,
    });
  };
}
```

```
};

const onError = async ({ code, detail }) => {
  reporter.error(code, detail);

  onChange({ status: 'error' });
};

sdk.plugins['wifiConfAirKiss'].start({
  wifiConfToken: token,
  targetWifiInfo: wifiInfo,
  autoRetry: true, // 自动处理故障流程
  familyId,
  roomId,
  onProgress,
  onComplete,
  onError
});

module.exports = AirKissConfigure;
```

配网步骤

sdk.plugins['wifiConfAirKiss'].start() 配网过程中，每执行完一个步骤就会触发一次 onProgress 回调，入参为：{ code, detail } 形式。

步骤	描述
WifiConfStepCode.WIFI_CONF_START	开始配网
WifiConfStepCode.PROTOCOL_START	配网协议开始
WifiConfStepCode.PROTOCOL_DETAIL	配网协议的细节，详细日志
WifiConfStepCode.PROTOCOL_SUCCESS	配网协议成功，获取到设备地址 detail: { data: { address } }，收到设备局域网地址，用于给设备发送信息
WifiConfStepCode.BUSINESS_START	业务流程开始
WifiConfStepCode.BUSINESS_QUERY_TOKEN_STATE_START	开始查询配网TOKEN状态

步骤	描述
WifiConfStepCode.BUSINESS_QUERY_TOKEN_STATE_SUCCESS	查询配网TOKEN状态成功
WifiConfStepCode.BUSINESS_ADD_DEVICE_START	开始添加设备
WifiConfStepCode.BUSINESS_ADD_DEVICE_SUCCESS	添加设备成功
WifiConfStepCode.BUSINESS_SUCCESS	业务流程成功 detail: { productId, deviceName}, 请求参数
WifiConfStepCode.WIFI_CONF_SUCCESS	配网成功

错误码

错误码	描述
UDP_NOT_RESPONSED	超时未收到设备响应
UDP_CLOSED	设备连接中断
UDP_ERROR	配网过程中触发 udp.onError 事件 detail: { errMsg }, 错误信息
UDP_SEND_MSG_FAIL	与设备 UDP 通信时, 发送消息失败
CONNECT_SOFTAP_FAIL	手机连接设备热点失败 detail: { errMsg }, 错误信息
BUSINESS_WIFI_RECONNECT_FAIL	手机连接 Wi-Fi 路由器失败 detail: { errMsg }, 错误信息
BUSINESS_DEVICE_ERROR	收到设备响应的错误 detail: { errMsg }, 错误信息
BUSINESS_INVALID_RESPONSE	收到非法的设备响应 detail: { response }, 具体的设备端响应
BUSINESS_DEVICE_CONNECT_MQTT_FAIL	设备连接 MQTT 服务失败
BUSINESS_DEVICE_CONNECT_WIFI_FAIL	设备连接目标 Wi-Fi 失败
BUSINESS_QUERY_BIND_TOKEN_TIMEOUT	设备连接云端超时
WIFI_CONF_FAIL	配网流程失败 detail: { errMsg }, 错误信息

错误码	描述
PROTOCOL_FAIL	配网协议失败 detail: { errMsg }, 错误信息
PROTOCOL_TIMEOUT	配网协议超时
PROTOCOL_INVALID_RESPONSE	配网协议收到非法响应 detail: { errMsg }, 错误信息

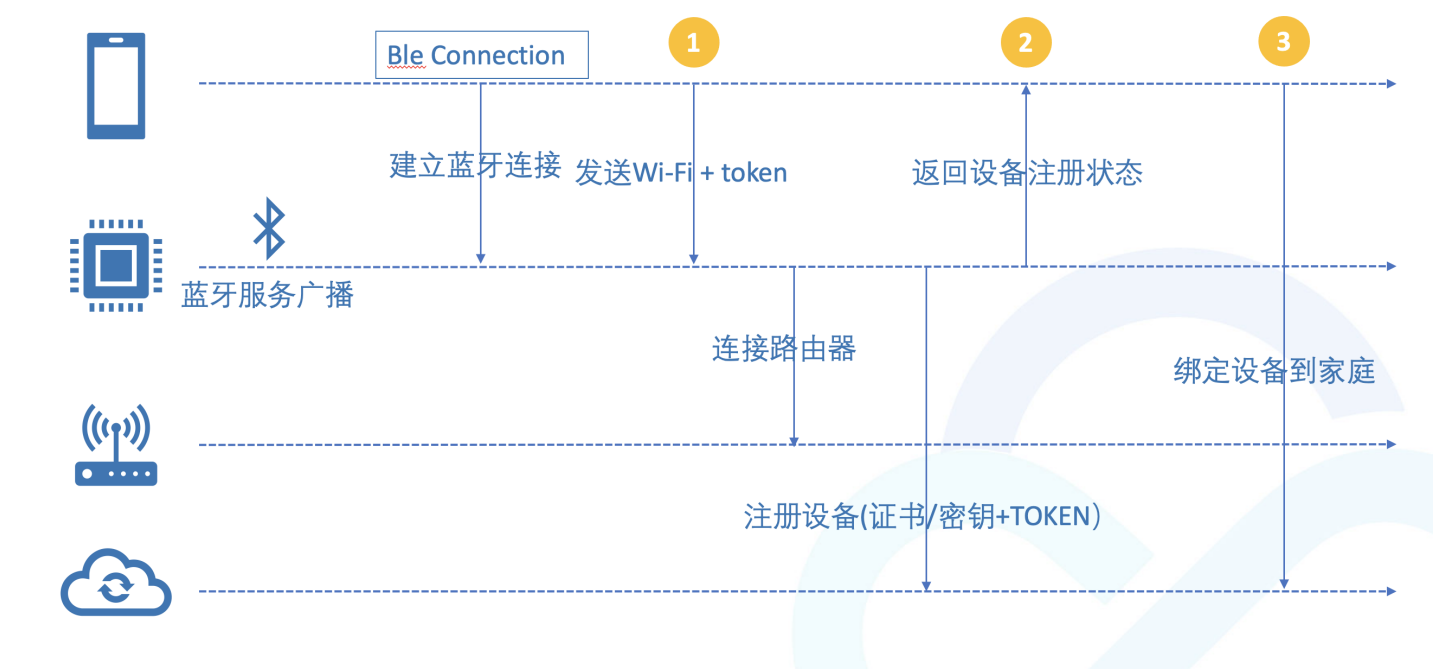
自动处理故障

配网流程中会出现一些错误，具体在 [错误码](#) 章节有展开叙述，在成功率不断优化的实践当中，总结一些可以自动处理的错误类型，处理成功后自动重试，全程用户无感知。

错误码	错误处理方式
PROTOCOL_TIMEOUT	超时未收到设备响应，其原因可能是中途网络被切走，导致设备和手机无法通信造成超时；自动处理方式：检查当前网络是不是目标网络，否则切到目标网络，重新配网。
UDP_ERROR	UDP 通道发生错误，可能的原因其一如上（中途网络被切走，导致设备和手机无法通信造成超时），其二是 Wi-Fi 切换之后，底层 UDP 还未切换，会发包失败；自动处理方式：延迟2s之后重新配网
UDP_SEND_MSG_FAIL	同上两种处理方式

蓝牙辅助配网

关于 蓝牙辅助方式配网的流程，请参见 [蓝牙辅助配网开发](#)。一键配网的配网流程图文版本如下：



腾讯连连中一键配网的页面交互流程如下，也给出来作为参考。



DeviceAdapter

如上所述，在配网之前，我们需要先发现设备，然后与设备建立蓝牙连接，并获得一个 DeviceAdapter 实例，来实现和蓝牙设备的通信。这个过程可以通过 BlueToothAdapter 完成。整个流程如下：

1. 创建一个 bluetoothAdapter

bluetoothAdapter 可以用来搜索设备，连接到设备。代码如下：

```
import { BleComboEspDeviceAdapter, BleComboLLSyncDeviceAdapter } from 'qcloud-iotexplorer-app-dev-plugin-wificonf-blecombo';
import { BlueToothAdapter } from 'qcloud-iotexplorer-bluetooth-adapter';
export const bluetoothAdapter = new BlueToothAdapter({
  deviceAdapters: [
    BleComboEspDeviceAdapter,
    BleComboLLSyncDeviceAdapter,
  ],
});
```

在实例化 `blueToothAdapter` 时，我们需要传入想要支持设备的 `DeviceAdapter`。目前插件内置了两种 `DeviceAdapter`：

- `BleComboEspDeviceAdapter`: 支持通过 [BluFi协议](#) 进行蓝牙辅助配网
- `BleComboLLSyncDeviceAdapter`: 支持通过 [LLSync协议](#) 进行蓝牙辅助配网

2. 获取蓝牙设备列表

通过 `bluetoothAdapter.startSearch` 方法，我们可以发现设备，获得设备列表。

```
await bluetoothAdapter.startSearch({
  ignoreDeviceIds,
  serviceIds,
  ignoreServiceIds,
  onError: (error) => {
    console.log('----error', error);
    // 搜索设备出错
    bluetoothAdapter.stopSearch();
  },
  onSearch: (devices) => {
    console.log('searched devices', devices);
    if (devices.length > 0) {
      console.log('找到设备', devices); // 此时可以在页面上展示
    }
  },
  timeout: 1.4 * 15 * 1000,
});
```

在上面的 `onSearch` 回调函数中，我们可以获得搜寻到的设备列表，这时可以将设备列表展示到页面上，供用户选择要连接哪个设备。此后可以调用 `bluetoothAdapter.stopSearch()` 来结束搜索。

3. 连接设备

用户从上面获取到的设备中选择一个，并发起连接操作时，可以调用 `bluetoothAdapter.connectDevice` 方法进行连接。连接成功后会返回一个 `deviceAdapter`，可以用来向连接的设备发送 Wi-Fi, token 等数据。

```
try {
  // device参数是上一步获取的devices中的某一个item
  const deviceAdapter = await bluetoothAdapter.connectDevice(device);

  if (!deviceAdapter) {
    throw {
```

```
code: 'CONNECT_ERROR',
};
}
} catch (err) {
console.error('连接到设备出错');
}
```

在上面三步完成之后，我们已经通过蓝牙连接到了设备，并获得了可以跟设备通信的 `deviceAdapter`，接下来就可以正式进行配网了。

bleComboOpts 配网参数

参数名	参数描述	类型	必填
wifiConfToken	Wi-Fi 设备配网 Token，从后台接口 生成 Wi-Fi 设备配网 Token 获取	string	是
targetWifiInfo	目标 Wi-Fi 信息，需要设备去连接的 Wi-Fi 的信息	WifiInfo	是
familyId	家庭 ID，默认为：'default'，即用户默认家庭 ID	'default' string	否
roomId	房间ID，默认为：''，即用户默认房间 ID	'' string	否
onProgress	- code：步骤代码，详见 配网步骤 小节 - msg：步骤描述，自行从 WifiConfStepDesp 拿 code 取 - detail：步骤详情，根据每个步骤不同而不同	({ code: WifiConfStepCode, detail?: object }) => void	否
onError	配网失败时触发 - code：错误代码，详见 错误码 小节 - msg：错误描述，自行从 WifiConfErrorMsg 拿 code 取 - detail：错误详情	({ code: WifiConfErrorCode, detail }) => void	是
onComplete	配网完成后触发	() => void	是
deviceAdapter	用于和设备进行蓝牙通信的设备适配器实例，连接蓝牙之后获得，详见 DeviceAdapter	DeviceAdapter	是
bleComboProto	使用的蓝牙配网协议，目前支持 ESP 官方和 LLSync 两种协议	'BLE_COMBO_ESP' / 'BLE_COMBO_LLSYNC'	否

WifiInfo 数据结构

属性名	属性描述	类型	必填
SSID	Wi-Fi 的 SSID	string	是
password	Wi-Fi 的 密码	string	是

示例代码

```
// 这里可以进行一些UI进度更新操作
const onStepChange = (progress) => {
  console.log(progress);
}

// 这里是配网进行过程中的回调函数
const onProgress = (data) => {
  console.info(data.code, data.detail);
  switch (data.code) {
    case WifiConfStepCode.PROTOCOL_START: // 开始配网
      onStepChange(1);
      break;
    case WifiConfStepCode.PROTOCOL_SUCCESS: // 设备联网成功，设备可以访问互联网
      onStepChange(2);
      break;
    case WifiConfStepCode.BUSINESS_QUERY_TOKEN_STATE_SUCCESS: // 发送token到设备成功，设备开始连接云端
      onStepChange(3);
      break;
    case WifiConfStepCode.WIFI_CONF_SUCCESS: // 配网成功
      onStepChange(4);
      break;
  }
};

const onComplete = ({ productId, deviceName }) => {
  // 配网成功后，可以拿到设备的 productId 和 设备名称
  console.log('配网成功', productId, deviceName);
};

const onError = async ({ code, detail }) => {
  console.error('配网出错', code, detail);
};
```

```
const config = {
  wifiConfToken, // 用于设备连接云端的token
  targetWifiInfo: { // 用于设备联网的wifi信息，由用户填入
    SSID: '您的Wi-Fi名称';
    password: '您的Wi-Fi密码';
    BSSID: '';
  },
  deviceAdapter, // 由连接设备之后获得
  wifiConfType: 'ble', // 'ble' | 'llsyncble'
  familyId: 'default',
  roomId,

  onProgress, // 用来更新页面的进度条
  onError,
  OnComplete,
}

// 开始执行配网逻辑 go!
sdk.plugins['wifiConfBleCombo'].start(config);
```

配网状态码

`sdk.plugins['wifiConfBleCombo'].start(bleComboOpts)` 配网过程中，每执行完一个步骤就会触发一次 `onProgress` 回调，入参为：{ code, detail } 形式。

步骤	描述
<code>WifiConfStepCode.PROTOCOL_START</code>	开始配网
<code>WifiConfStepCode.PROTOCOL_SUCCESS</code>	设备联网成功，设备可以访问互联网
<code>WifiConfStepCode.BLE_SEND_TOKEN_START</code>	开始发送token到设备，用于连接云端
<code>WifiConfStepCode.BLE_SEND_TOKEN_SUCCESS</code>	发送token到设备成功，设备开始连接云端
<code>WifiConfStepCode.WIFI_CONF_SUCCESS</code>	配网成功

蓝牙辅助配网错误码

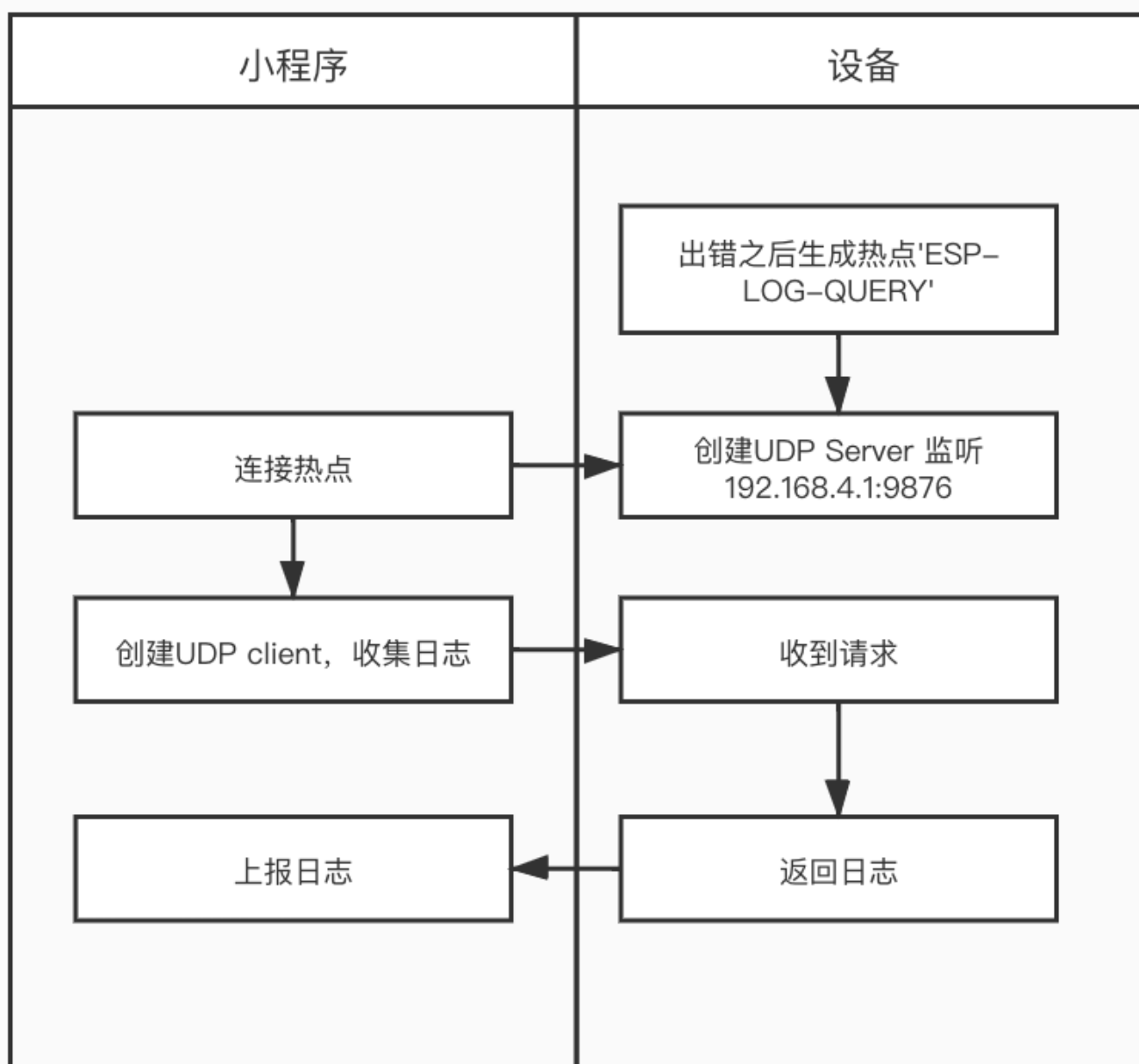
在 `onError` 回调函数中，我们可以拿到配网失败的错误码。

CODE	描述
<code>PROTOCOL_FAIL</code>	设备连接失败

CODE	描述
BLE_SEND_TOKEN_ERROR	发送token到设备失败
WIFI_CONF_FAIL	配网失败

模组日志收集

当发生错误的时候，只看 onProgress 或者 onError 里面打印出来的信息并不能准确定位到问题，需要结合设备端日志查看，我们制定了跟设备端的日志交互协议，原理如下：



示例代码

```
const { collectModuleLog } = require('qcloud-iotexplorer-appdev-plugin-wificonf-core').utils;

collectModuleLog({
  // 用于上报的对象
  reporter: console,
  sdk: sdk,
})
```

日志上报打印详情

日志收集的过程以及结果会通过 `reporter.info(code, detail)` 的回调打印出来，可以通过这个方法收集信息以及日志的上报。

步骤	描述
WifiConfStepCode.MODULE_REPORT_START	开始配网日志收集
WifiConfStepCode.MODULE_REPORT_CONNECT_WIFI_START	日志收集开始连接设备热点
WifiConfStepCode.MODULE_REPORT_CONNECT_WIFI_SUCCESS	日志收集连接设备热点成功
WifiConfStepCode.MODULE_REPORT_COMMUNICATE_AP_START	开始收集设备端日志
WifiConfStepCode.MODULE_REPORT_COMMUNICATE_AP_SUCCESS	收集成功 detail: { moduleDetail}, 日志详情

错误码

错误码	描述
MODULE_REPORT_COMMUNICATE_AP_ERROR	和设备端通信失败
MODULE_REPORT_TIMEOUT	收集日志超时

长连接通信

订阅设备信息

通过 WebSocket 监听服务端实时推送的设备上下线状态及属性数据。

```
sdk.subscribeDevices(deviceList: string[] | deviceInfo[]): Promise< void >
```

参数

参数名	参数描述	类型	必填
deviceList	设备 ID 列表，或设备信息列表（deviceInfo 需包含 DeviceId 字段）	string[] deviceInfo[]	是

示例代码

通过设备 ID 列表订阅

```
sdk.subscribeDevices([
  'Product1/Device1',
  'Product1/Device2',
  'Product2/Device3'
]);
```

通过设备列表订阅

```
sdk.requestApi('AppGetFamilyDeviceList', { FamilyId: 'default' })
  .then(data => {
    sdk.subscribeDevices(data.DeviceList);
  })
```

手动建立长连接

手动连接 WebSocket。一般不需要调用，除非关闭了 sdkOptions.disconnectWhenAppHide 选项。

```
sdk.connectWebsocket() => Promise< void >
```

示例代码

```
sdk.connectWebsocket();
```

手动断开长连接

手动断开 WebSocket。一般不需要调用，除非关闭了 sdkOptions.autoReconnect 与 sdkOptions.connectWhenAppShow 选项。

```
sdk.disconnectWebsocket() => Promise< void >
```


示例代码

```
sdk.disconnectWebsocket();
```

监听事件

监听 WebSocket 事件。

```
sdk.on(type: EventTypes, listener: (...args) => void) => void
```

参数

参数名	参数描述	类型	必填
type	要监听的事件	EventTypes	是
listener	事件触发时的回调函数	(param) => void	是

示例代码

```
const { EventTypes } = require('qcloud-iotexplorer-appdev-sdk').AppDevSdk.constants;

// 监听设备上报数据推送
sdk.on(EventTypes.WsReport, ({ deviceId, deviceData }) => {
  console.log('websocket device report', deviceId, deviceData);
});
```

取消监听事件

取消监听 WebSocket 事件。

```
sdk.off(type: EventTypes, listener: (...args) => void) => void
```

参数

参数名	参数描述	类型	必填
type	要取消监听的事件	EventTypes	是
listener	要取消监听的事件的回调函数，不传则清除该事件的所有回调函数	(param) => void null	否

示例代码

```
const { EventTypes } = require('qcloud-iotexplorer-appdev-sdk').AppDevSdk.constants;

const listener = ({ deviceId, deviceData }) => {
  console.log('websocket device report', deviceId, deviceData);
};

// 监听设备上报数据推送
sdk.on(EventTypes.WsReport, listener);

// 取消监听设备上报数据推送
sdk.off(EventTypes.WsReport, listener);
```

事件列表

事件类型	描述	参数
EventTypes.WsReport	设备上报数据	{ deviceId, deviceData }
EventTypes.WsControl	设备控制数据	{ deviceId, deviceData }
EventTypes.WsStatusChange	设备在线状态变更	{ deviceId, deviceStatus }
EventTypes.WsEventReport	设备事件上报	{ Payload }
EventTypes.WsActionReport	设备行为上报	{ Payload }
EventTypes.WsActionPush	设备行为下发	{ Payload }
EventTypes.WsPush	WebSocket 推送原始数据	{ push, action, params }
EventTypes.WsError	WebSocket 发生错误	WebSocket error 事件的原始错误信息
EventTypes.WsClose	WebSocket 连接关闭	{ code, reason }

错误处理

SDK 所有接口的错误都经过标准化处理为 { code, msg, ...detail } 的形式。具体取值根据接口的不同而不同。

全局错误码

⚠ 注意：

下文中描述为一个对象的 detail，实际上是解构到错误对象当中的。例如 INTERNAL_ERROR 的具体 Error 为 { code: 'INTERNAL_ERROR', msg: Error.message, stack: Error.stack, error: Error }。

错误码	描述
ErrorCode.VERIFY_LOGIN_FAIL	未登录或登录态已失效
ErrorCode.INTERNAL_ERROR	JS Error 和 detail: { stack, error } 分别为错误堆栈和原始错误对象
ErrorCode.GET_USERINFO_NEED_AUTH	调用 wx.getUserInfo 时用户未授权用户信息权限，遇到该错误时需要引导用户授权 detail: { errMsg }, 小程序 API 的原始错误信息
ErrorCode.WX_API_FAIL	调用小程序 API 报错 detail: { errMsg }, 小程序 API 的原始错误信息

调用应用端 API 错误码

除以上全局错误码，其余错误码为应用端 API 响应中的错误码，具体错误码请查看对应的应用端 API 文档。同时接口的错误中会包含标识该次请求的 detail.reqId，可用来查询该次请求的详细日志。

迁移指南

从 v0.x 版本迁移到 v1.x 版本

更新项目依赖

更新小程序 SDK 到 v1.x 版本，需要更新项目依赖，请在项目目录下的命令行中执行以下命令：

```
npm install qcloud-iotexplorer-appdev-sdk@1
```

❓ 说明：

如果您使用了微信开发者工具的 npm 支持，在更新项目依赖后，需要选择微信开发者工具菜单栏的**工具 > 构建 npm**以重新构建 npm 依赖。

调整导入 SDK 的方式

导入 v0.x 版本小程序 SDK 的方式为：

```
// 导入 SDK
const {
  QcloudIotExplorerAppDevSdk
} = require('qcloud-iotexplorer-appdev-sdk/qcloud-iotexplorer-appdev-sdk');
```

```
// 导入常量
const {
  EventTypes,
} = require('qcloud-iotexplorer-appdev-sdk/qcloud-iotexplorer-appdev-sdk');

const sdk = new QcloudIotExplorerAppDevSdk({
  // 参数略
});
```

小程序 SDK 升级到 v1.x 版本后，导入的方式需要调整为：

```
// 导入 SDK
const { AppDevSdk } = require('qcloud-iotexplorer-appdev-sdk');

// 导入常量
const { EventTypes } = AppDevSdk.constants;

const sdk = new AppDevSdk({
  // 参数略
});
```

调整配网代码

v0.x 版本小程序 SDK 内置支持 Softap 配网及 SmartConfig 配网。v1.x 版本小程序 SDK 增加了更多配网方式的支持，并将配网模块独立出来，开发者可以按需导入。关于小程序 SDK 具体支持的配网方式，请参见 [设备配网](#)。

选择需要安装的配网插件，并在命令行中执行相应的安装命令。下面以安装 AirKiss 配网插件为例。

```
npm install qcloud-iotexplorer-appdev-plugin-wificonf-airkiss
```

向 SDK 注册配网插件，请参照以下代码（以 AirKiss 配网插件为例）。

```
import AirKissPlug from 'qcloud-iotexplorer-appdev-plugin-wificonf-airkiss';

AirKissPlug.install(sdk);
```

调用配网插件进行配网，请参照以下代码（以 AirKiss 配网插件为例）。

```
import { constants as WifiConfConstants } from 'qcloud-iotexplorer-appdev-plugin-wificonf-core';

const {
  WifiConfErrorMsg, WifiConfStepCode,
} = WifiConfConstants;

/**
 * airkiss一键配网
 */
export function AirKissConfigure({
  token,
  wifiInfo = {
    SSID: "",
    password: "",
    BSSID: "",
  },
  familyId = 'default',
  roomId,
  reporter,
  onStepChange,
  onStatusChange,
}) {
  const onProgress = (data) => {
    reporter.info(data.code, data.detail);

    switch (data.code) {
      case WifiConfStepCode.PROTOCOL_SUCCESS:
        onStepChange(1);
        break;
      case WifiConfStepCode.CREATE_UDP_CONNECTION_SUCCESS:
        onStepChange(2);
        break;
      case WifiConfStepCode.BUSINESS_QUERY_TOKEN_STATE_SUCCESS:
        onStepChange(3);
        break;
      case WifiConfStepCode.WIFI_CONF_SUCCESS:
        onStepChange(4);
        break;
    }
  };
};
```

```
const onComplete = ({ productId, deviceName }) => {
  onChange({
    status: 'success',
    productId,
    deviceName,
  });
};

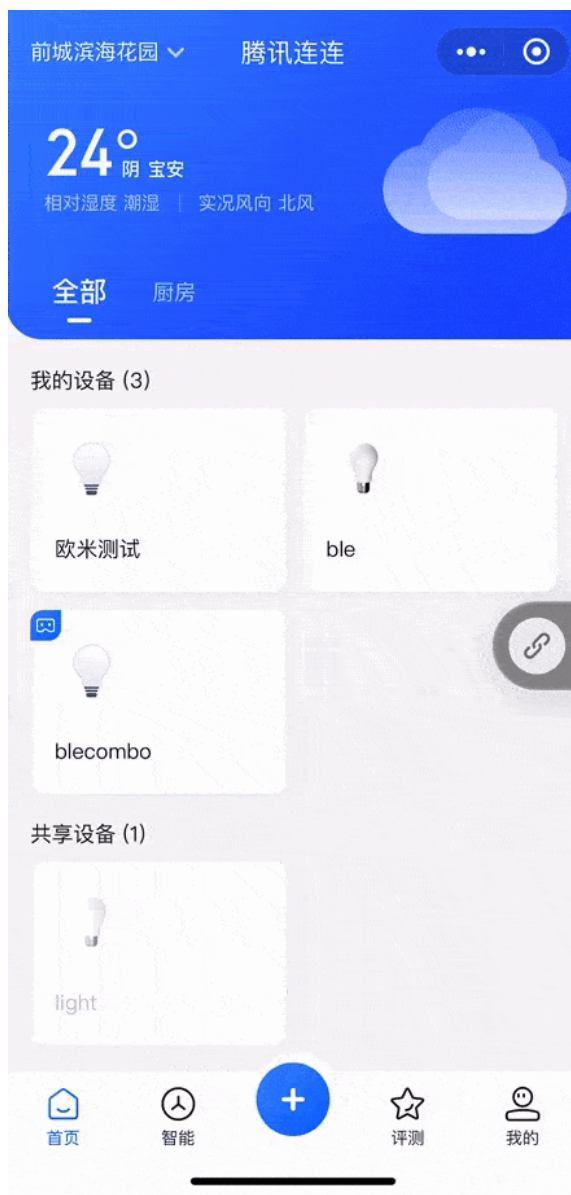
const onError = async ({ code, detail }) => {
  reporter.error(code, detail);

  onChange({ status: 'error' });
};

sdk.plugins['wifiConfAirKiss'].start({
  wifiConfToken: token,
  targetWifiInfo: wifiInfo,
  autoRetry: true, // 自动处理故障流程
  familyId,
  roomId,
  onProgress,
  onComplete,
  onError
});
}
```

添加 LLSync 蓝牙设备

通过 [qcloud-iotexplorer-bluetooth-adapter-llsync](#) sdk，小程序可以完成和标准蓝牙设备进行连接，绑定，控制等流程。通过下面的图片可以直观地了解整个流程：



您也可以通过官方的 [小程序 SDK demo](#) 的添加标准蓝牙设备部分来掌握连接标准蓝牙设备的流程。

1. 创建一个 bluetoothAdapter

bluetoothAdapter 可以用来搜索设备，连接到设备。代码如下：

```
import { BluetoothAdapter } from 'qcloud-iotexplorer-bluetooth-adapter';
import { LLSyncDeviceAdapter } from 'qcloud-iotexplorer-bluetooth-adapter-llsync';

// 关于appDevSdk文档，详见https://www.npmjs.com/package/qcloud-iotexplorer-appdev-sdk
const options = {
  appDevSdk, // 通过qcloud-iotexplorer-appdev-sdk得到的实例
}
LLSyncDeviceAdapter.injectOptions(options);
export const bluetoothAdapter = new BluetoothAdapter({
  deviceAdapters: [
```

```
LLSyncDeviceAdapter,  
],  
});
```

2. 获取蓝牙设备列表

通过 `bluetoothAdapter.startSearch` 方法，我们可以发现设备，获得设备列表。

```
const serviceIds = [LLSyncDeviceAdapter.serviceId];  
await bluetoothAdapter.startSearch({  
  serviceIds,  
  onError: (error) => {  
    console.log('----error', error);  
    // 搜索设备出错  
    bluetoothAdapter.stopSearch();  
  },  
  onSearch: (devices) => {  
    console.log('searched devices', devices);  
    if (devices.length > 0) {  
      console.log('找到设备', devices); // 此时可以在页面上展示  
    }  
  },  
  timeout: 1.4 * 15 * 1000,  
});
```

在上面的 `onSearch` 回调函数中，我们可以获得搜寻到的设备列表，这时可以将设备列表展示到页面上，供用户选择要连接哪个设备。

tip: 如果设备无法搜索到，请确认设备没有被绑定。

3. 连接设备

用户从上面获取到的设备中选择一个，并发起连接操作时，可以调用 `bluetoothAdapter.connectDevice` 方法进行连接。连接成功后会返回一个 `deviceAdapter`，可以用来向连接的设备发送Wi-Fi，token等数据。

tip: 如果在连接时提示没有权限操作该产品，请到控制台/应用开发对应用和产品进行关联

```
try {  
  // device 参数是上一步 onSearch 回调中获取 devices 数组的某一项  
  const deviceAdapter = await bluetoothAdapter.connectDevice(device);  
  
  if (!deviceAdapter) {
```



```
throw {  
  code: 'CONNECT_ERROR',  
}  
}  
} catch (err) {  
  console.error('连接到设备出错');  
}
```

在这一步中，可以通过连接设备获得 deviceAdapter实例，通过 deviceAdapter，我们可以完成后续设备的绑定和解绑操作。

4. 绑定设备

绑定设备时，可以传入familyId, roomId, 从而将设备绑定到特定的家庭和房间，绑定完成后，可以在设备列表中看到该设备。

```
try {  
  const deviceId = await deviceAdapter.bindDevice({ familyId, roomId });  
} catch (err) {  
  console.log(err);  
}
```

5. 解绑设备

设备绑定后，也可以通过小程序发起删除设备的操作，这时需要调用解绑api，解绑完成后设备会恢复未绑定的状态。

```
try{  
  await deviceAdapter.unbindDevice({ familyId, deviceId });  
} catch (err) {  
  console.log(err);  
}
```

6. 断开设备

我们可以通过 deviceAdapter.disconnectDevice() 断开设备连接：

```
await deviceAdapter.disconnectDevice()
```

7. deviceAdapter 事件

事件	描述	参数

事件	描述	参数
connect	蓝牙设备连接时触发	DeviceInfo
disconnect	蓝牙设备连接断开时触发	DeviceInfo
authorized	蓝牙设备授权完成时触发	{ version, mtu, otaVersion, }

音乐服务

公共说明

获取实例

```
npm install qcloud-iotexplorer-tme-sdk

import { TMEsdkForMiniProgram } from 'qcloud-iotexplorer-tme-sdk';
// 注：appDevSdk为初始化后的实例
const tmeSdk = new TMEsdkForMiniProgram(appDevSdk);
```

接口通用返回

接口统一返回值

接口调用的返回值统一为 Promise<TMEResponse> 类型

```
interface TMEResponse {
  error_code: number,
  error_msg: string,
  data?: any;
}
```

- 调用成功：返回一个 resolved 的 Promise，其值为 TMEResponse 类型，error_code=0，data 为返回结果。
- 调用失败：返回一个 rejected 的 Promise，包含错误码（error_code）及提示信息（error_msg）

属性名	描述	类型
error_code	错误码	number
error_msg	错误信息	string
data	响应数据	object

错误码列表

错误码	说明
200001	参数错误
200002	系统繁忙,如幂等接口并发调用等，通常由于用户并发操作造成
200003	认证信息过期或错误,请重新登录
200004	设备未激活
200005	当前sp暂未支持此接口
200006	系统错误,如内部调用超时等，由于服务内部异常导致
200200	可直充剩余次数为0
400000	登录授权失败
400001	设备端超时无响应
400002	调用SDK参数错误

登录授权部分

用户设备登录授权

跳转酷狗音乐小程序授权，当再次返回 h5 或小程序时，Promise 状态改变。

接口定义

```
tmeSdk.login(deviceId: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
deviceId	设备Id	string	是

返回值

返回一个 Promise<TMEResponse>

用户设备登出

原 token 将登出。

接口定义

```
tmeSdk.getUserInfo(deviceId: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
deviceId	设备 Id	string	是

返回值

返回一个 Promise<TMEResponse>

校验设备授权

接口定义

```
tmeSdk.checkDeviceAuth(deviceId: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
deviceId	设备 Id	string	是

返回值

返回一个 Promise<TMEResponse>

获取用户信息

接口定义

```
tmeSdk.getUserInfo(deviceId: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
deviceId	设备 Id	string	是

返回值

返回一个 Promise<TMEResponse>，其中data如下：

属性名	描述	类型
userid	用户id	string
nick_name	用户昵称	string
img	用户头像	string
is_vip	是否 vip 0:否 1:是	enum: 0 1
vip_end_time	vip 有效期终止时间	string
car_vip_end_time	车机会员有效期终止时间	string
svip_end_time	豪V有效期终止时间	string

播控部分

接口描述

调用播控 SDK，会下发物模型属性 +control_seq，需要设备上报相同的 control_seq

- 若在超时范围内收到上报，视为下发播控成功，返回resolved状态的 Promise<TMEResponse>
- 若超时未收到上报，返回 rejected 状态的 Promise<TMEResponse>

超时设置可以通过 tmeSdk.config.timeout 来配置，默认值为10000，单位：毫秒(ms)

通用播控接口

接口定义

```
tmeSdk.controlKugouDeviceData(deviceData, deviceId: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
deviceData	设备物模型数据	object	是
deviceId	设备 Id	string	是

返回值

返回一个 Promise<TMEResponse>

播放

接口定义

```
tmeSdk.play(deviceId: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
deviceId	设备 Id	string	是

返回值

返回一个 Promise<TMEResponse>

暂停

接口定义

```
tmeSdk.pause(deviceId: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
deviceId	设备 Id	string	是

返回值

返回一个 Promise<TMEResponse>

上一首

接口定义

```
tmeSdk.preSong(deviceId: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
deviceId	设备 Id	string	是

返回值

返回一个 Promise<TMEResponse>

[下一首](#)[接口定义](#)

```
tmeSdk.nextSong(deviceId: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
deviceId	设备 Id	string	是

返回值

返回一个 Promise<TMEResponse>

设置播放模式[接口定义](#)

```
tmeSdk.setPlayMode(playMode: number, deviceId: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
playMode	播放模式：0:顺序播放 1:单曲循环 2:随机播放	enum: 0 1 2	是
deviceId	设备Id	string	是

返回值

返回一个 Promise<TMEResponse>

设置音量[接口定义](#)

```
tmeSdk.setVolume(volume: number, deviceId: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
volume	音量：0-100之间	number	是

参数名	参数描述	类型	必填
deviceId	设备Id	string	是

返回值

返回一个 Promise<TMEResponse>

设置播放进度

接口定义

```
tmeSdk.setPlayPosition(playPosition: number, deviceId: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
playPosition	播放进度：单位:秒(s)	number	是
deviceId	设备Id	string	是

返回值

返回一个 Promise<TMEResponse>

设置播放质量

接口定义

```
tmeSdk.setPlayQuality(recommendQuality: number, deviceId: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
recommendQuality	播放质量：0:标准 1:高清 2:无损	enum: 0 1 2	是
deviceId	设备Id	string	是

返回值

返回一个 Promise<TMEResponse>

设置当前播放歌曲

接口定义

```
tmeSdk.playSong(songId: string, songIndex: string, newQueueType: string, newQueueId: string | number, deviceId: string) => Promise<TMEResponse>
```

参数说明

参数名	参数描述	类型	必填
songId	歌曲 ID	string	是
songIndex	歌曲所在播放列表的位置，从0开始	number	是
newQueueType	播放列表的类型： playlist newSongs recommendDaily	string	是
newQueueId	播放列表 ID（当类型为"每日推荐"时，不存在id，传 undefined）	string number	是
deviceId	设备 Id	string	是

播放列表目前支持三种类型：歌单(playlist)、新歌首发(newSongs)、每日推荐(recommendDaily)

返回值

返回一个 Promise<TMEResponse>

内容部分

拉取内容通用接口

请求酷狗 API 拉取内容通用接口

接口定义

```
tmeSdk.requestTMEApi(action: string, params, deviceId: string) => Promise<TMEResponse>
```

参数说明

参数名	描述	类型	必填
action	接口 action	string	是
params	请求参数，无请求参数时，传{}	object	是
deviceId	设备 Id	string	是

返回值

返回一个 Promise<TMEResponse>

注：action、params 及 返回值 data 参考 [音乐服务](#)

获取设备当前播放歌曲

接口定义

```
tmeSdk.getCurrentPlaySong(deviceId: string) => Promise<TMEResponse>
```

参数说明

参数名	描述	类型	必填
deviceId	设备 Id	string	是

返回值

返回一个 Promise<TMEResponse>，data 为歌曲信息

获取设备当前播放列表

根据目前支持的播放类型(playType)，拉取对应的歌单列表，并查出歌曲的详细信息

接口定义

```
tmeSdk.getCurrentPlayQueue(deviceId: string) => Promise<TMEResponse>
```

参数说明

参数名	描述	类型	必填
deviceId	设备 Id	string	是

返回值

返回一个 Promise<TMEResponse>

TMEResponse 中 data 如下

属性名	描述	类型
playType	播放列表类型	enum: playlist newSongs recommendDaily

属性名	描述	类型
queueId	当前播放列表 id，根据 playType 对应 playlist_id、album_id、top_id	string number
total	列表中歌曲总数	number
songs	歌曲数组，具体歌曲属性参考 TME 文档中 Song 属性	Array[]

获取歌曲详细信息

接口定义

```
tmeSdk.getSongDetail(songId: string, deviceId: string) => Promise<TMEResponse>
```

通过调用 requestTMEApi，请求歌曲播放链接与歌曲信息，返回歌曲的详细信息

参数说明

参数名	描述	类型	必填
songId	歌曲 ID	string	是
deviceId	设备 Id	string	是

返回值

返回一个 Promise<TMEResponse>，data 为歌曲信息

获取歌单详细信息

接口定义

```
tmeSdk.getPlaylistDetail(action: string, params, deviceId: string) => Promise<TMEResponse>
```

通过调用 requestTMEApi，请求歌单列表与歌曲信息，丰富列表中的歌曲信息，返回歌单列表

参数说明

参数名	描述	类型	必填
action	新歌首发(awesome_newsong)、每日推荐(awesome_everyday)、歌单歌曲(playlist_song)	string	是
params	参考应用端 API=>音乐服务中对应 API 的 KugouParams	object	是
deviceId	设备 Id	string	是

返回值

返回一个 Promise<TMEResponse>，data 为歌单列表

连连企业接入指南

最近更新时间：2021-06-07 17:30:29

连连企业是腾讯云物联网面向产业领域客户提供的企业服务平台，以连连企业小程序为载体，基于腾讯云物联网开发平台开发场景化行业应用，同时开放二级入口，提供给第三方厂商基于业务场景开发自主品牌小程序。厂商开发自主品牌小程序完成后，可以申请入驻连连企业，基于连连企业小程序提供统一产业服务入口，客户可以自主选择申请开通连连企业内不同的行业场景应用。

接入步骤

1. 基于连连企业小程序 SDK 开发自主品牌小程序，详情请参见 [开发指南](#)。
2. 申请接入腾讯连连企业版。通过邮箱 IoT_Explorer_Support@tencent.com 发送接入申请，申请邮件中需提供对应的公司名称、营业执照编号、联系人姓名电话以及您的诉求。

说明：

如在使用物联网开发平台的过程中遇到问题，请通过QQ群 737640300 联系我们，我们将有专门的工作人员为您解答。

小程序开发

腾讯云物联网开发平台向开发者开放小程序 SDK，以及一个 Demo 小程序。小程序 SDK 封装了平台的通用能力，开发者可基于小程序 SDK 开发自定义业务场景的小程序。Demo 小程序以开源的方式向开发者开放，便于开发者在开发自定义业务场景的小程序时进行参考。

小程序 SDK

小程序 SDK 提供的通用能力如下：

- [设备配网](#)
- [设备管理](#)
- [长连接通信](#)
- [调用应用端 API](#)
- [设备定时](#)
- [固件升级](#)
- [用户管理](#)
- [消息管理](#)

说明：

设备管理、设备定时、固件升级、用户管理和消息管理等能力通过应用端 API 提供。小程序 SDK 封装了调用应用端 API 的能力，开发者只需参照 [应用端 API](#) 文档，[调用应用端 API](#) 即可使用相应的能力。

关于小程序 SDK 的更多信息，详情请参见 [开发指南](#) 以及 [小程序 SDK](#)。

小程序 Demo

腾讯云物联网开发平台为开发者提供一个 Demo 小程序例程，开发者可以根据文档指引，部署并体验一个属于自己的 Demo 小程序。关于小程序 Demo 的获取及使用方式，详情请参见 [自主品牌小程序快速入门](#)。

App 开发

通用版 App

最近更新时间：2021-08-16 15:38:15

通用版 App，用户无需开发 App 即可构建跨产品应用。腾讯连连是腾讯云面向物联网行业提供的一整套 C2B2C 开放平台服务，其中腾讯连连通用版 App 是腾讯云 IoT 团队为用户打造的应用开发服务，基于 Android 和 iOS 系统，实现智能设备端、云端、移动端的相互打通。具备设备管理、设备控制、群组管理、消息通知、场景联动、产品评测、实时音视频等多功能模块，能够满足用户远程操控智能设备等多种场景，消费者只需要通过一次设备配网，即可轻松构建智能生活。

多设备管理
一键操作

远程控制
省心安心

智能配网
轻松搞定

邀请成员
全家享受



下载方式

• iOS

iOS 通用版 App 目前发布 Beta 版本，可以通过 [TestFlight](#) 进行下载体验。

❓ 说明：

您需要使用 iOS 系统手机访问上述链接。

• Android

Android 通用版 App 目前已经上架到各大 Android 应用市场，包括小米应用商店、华为应用市场、OPPO 应用商店、vivo 应用商店、360手机助手以及安智市场，可以前往对应的手机应用市场搜索[腾讯连连](#)进行下载体验。

版本更新

V1.5.0

- 发布日期：2021/08/13
- 更新内容：
 - 支持英文语言，根据系统语言切换中英文。
 - 家庭定位地图选址优化。
 - 视觉优化。
 - 修复已知问题。

V1.4.0

- 发布日期：2021/04/15
- 内容如下：
 - 首页视觉优化。
 - 首页添加天气功能。
 - 家庭定位支持地图选址。
 - 切换时区时提示优化。
 - 修复部分已知问题。

V1.3.0

- 发布日期：2021/01/19
- 内容如下：
 - 新增扫一扫产品落地详情页，支持控制台详情页配置。
 - 优化实时音视频信令逻辑，提升用户体验。
 - 修复 App H5 自定义控制面板的已知问题。
 - App 页面视觉更新。
 - 设备绑定优化，为用户自动填写默认设备名称。

V1.2.0

- 发布日期：2020/12/04
- 内容如下：
 - 新增智能场景联动功能。
 - 新增 H5 自定义控制面板。
 - 新增 TRTC 场景通话，满足用户与设备端音视频通话需求。
 - 修复了部分已知 bug。

V1.1.0

- 发布日期：2020/11/02
- 内容如下：
 - 新增评测功能。

- 优化登录异常信息提示。
- 消息通知界面交互优化。
- 新增通过 App 扫码，控制台批量投产二维码进入配网流程功能。

功能模块介绍

账号功能

注册

- 手机号注册：通过手机号获取验证码注册。
- 邮箱注册：通过邮箱账号获取验证码注册。

登录

- 手机号登录：通过手机号和密码登录，登录成功后，用户信息储存在云端。
- 邮箱登录：通过邮箱和密码登录，登录成功后，用户信息储存在云端。
- 第三方登录：使用微信登录，连通腾讯连连的账号体系。

密码找回

- 手机号找回密码：用户通过已注册腾讯连连的手机号找回登录密码。
- 邮箱账号找回密码：用户通过已注册腾讯连连的邮箱账号找回登录密码。

登出

退出登录：用户可以在个人中心的账户中退出当前的登录状态。

账号注销

账号注销：用户可以根据自己的实际情况选择注销账号，删除账号的所有信息。

账号设置

- 设置头像：用户可以自定义上传个性头像。
- 设置昵称：用户可以自行设置昵称。
- 修改密码：用户可以自行按照要求进行密码修改。

设备管理

- 添加设备：可以通过二维码扫码、蓝牙搜索、设备列表选择添加设备。
- 设备列表：展示所有已添加的设备。
- 设备分组：根据不同的房间对设备进行不同的分组。
- 查看设备信息：可以查看设备名称、设备信息、房间信息。
- 修改设备名称：可以修改设备的名称。
- 设置设备房间：可以自定义设备归属房间。
- 设备分享：可以通过亲友的手机号或者邮箱账号分享设备共同使用。

- 删除设备：可以手动删除已经添加的设备。

设备控制

- 状态查询与操作：可以实时在线查询操作设备的开启与关闭状态。
- 智能操控：按照应用的设备面板和设备支持的功能进行控制设备。

家庭管理

- 家庭列表管理：一个账号支持多个家庭给管理。
- 家庭切换：支持多家庭的用户，切换家庭进行设备使用。
- 添加家庭：可以新增家庭。
- 家庭地址：可以备注家庭所在地址。
- 家庭名称修改：可以修改家庭名称。
- 房间管理：可以新增或者删除家庭下的房间。
- 家庭成员：可以邀请或者移除家庭成员。

消息通知

- 消息通知：问题反馈的最新进展将会通过手机消息通知栏的方式发送给用户。
- 应用内消息：分为设备信息、家庭信息、其他通知，用户可进行点击查看。

场景联动

- 手动智能：用户创建手动智能场景，能够一键控制所有添加了执行任务的智能设备。同时提供执行日志，用户可查看智能执行结果。
- 自动智能：用户创建自动智能场景，对场景触发条件和执行任务进行设置，能够使不同品类的智能设备在特定情况下产生联动交互，同时支持发送通知功能，为用户提供场景化服务。
- 功能属性配置：支持自定义配置产品在智能场景中，可被用户添加为触发条件或执行任务的功能属性。

产品评测

对于已接入腾讯连连的厂商，支持产品评测文章展示，能够将精选产品触达更多用户。

实时音视频

- 支持腾讯连连与设备端实时音视频通话。
- 支持呼叫消息信鸽推送。

其他功能

- 帮助中心：可以通过帮助中心，查看常见问题，快速上手使用。
- 问题反馈：用户使用的问题，可以通过问题反馈到后台，进行产品优化。
- 设备日志上传：用户可在遇到问题的时候进行日志上报，以定位和解决用户在使用过程中遇到的问题。

基于 App 开源版概述

最近更新时间：2021-04-26 15:13:24

腾讯连连开源版 App 是腾讯云 IoT 官方为开发者提供的 App Demo，开发者可基于 App Demo 进行 App 的定制开发，打造个性化品牌，基于现有资源快速实现产品落地并投入市场。开发过程中开发者无需从头搭建，只需要根据官网提供的 App Demo 进行配置或者二次开发，即可方便快速地拥有自己的专属品牌 App。

产品优势

快速搭建

无需从头开始搭建 App，只需要根据官网提供的 App 开源版进行配置或者二次开发，开发者即可拥有自己的专属品牌 App。

灵活开发

开发者可以根据实际需要，自由定制 App 的功能和外观，不局限于官方提供 App Demo 原本功能，按需所取，灵活开发。

降低成本

App 底层开发、Bug 修复、文档编写都需要花费大量的人力、物力、财力，选择开源版 App，可以基于腾讯云 IoT 官方提供的开源 App 既有成果，获取最新的产品功能。

开源共建

开源 App 生态由腾讯云 IoT 官方和第三方开发者共同参与建设，在官方提供的 App 基本能力框架之上丰富功能和特性，完善腾讯连连应用生态。

接入指南

最近更新时间：2021-03-17 11:46:04

本文为您详细介绍获取第三方定制 App 部署接入参数的相关操作。

登录腾讯云

注册 [腾讯云账号](#) 并完成 [实名认证](#)。

App 源码获取

- iOS 版本 App 可通过 [腾讯连连-iOS](#) 下载源码。
- Android 版本 App 可通过 [腾讯连连-Android](#) 下载源码。

物联网开发平台控制台

1. 进入 [物联网开发平台控制台](#)，单击【新建项目】，填写“项目名称”、“项目描述”，单击【保存】。

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

2. 单击项目详情进入产品开发详情页，单击【应用开发】>【新建应用】，填写“应用名称”、“备注”，单击【保存】。

新建应用



应用名称 *

支持中文、英文、数字、下划线的组合，最多不超过50个字符

备注

选填

最多不超过1024个字符

保存

取消

3. 应用创建成功后，单击“应用名称”进入应用信息详情页，获取 App Key 和 App Secret。

应用信息

应用名称 腾讯连连

应用描述 -

创建时间 2020-09-01 18:35:40

Andriod应用 ①

APP Key

APP Secret

iOS应用 ①

APP Key

APP Secret

小程序应用 ①

APP Key

APP Secret

4. 将App Key写入 [配置文件](#)，将App Secret保存在 [自建后台](#)，将 appapi 调用由设备端发起切换为由自建后台服务发起。

搭建后台服务, 将 App API 调用由设备端发起切换为由自建后台服务发起

关于应用端 API，请参见 [应用端 API 简介](#)。

⚠ 注意：

登录前所使用的 API URL 为 <https://iot.cloud.tencent.com/api/exploreropen/appapi>，不建议在设备端调用，需要替换为自建的后台服务，以避免密钥的泄漏。

- [api/studioapp/*](#) 为公版 App 专用，OEM 的 App 使用的是应用端 API（[api/exploreropen/](#)），当在 [App 参数写入配置文件](#) 中配置 [TencentIotLinkAppkey](#) 后，[api/studioapp](#) 调用将自动切换为应用端 API 调用。

- App API（[api/exploreropen/appapi](#)）请在自建后台进行调用，Token API（[api/exploreropen/tokenapi](#)）可安全在设备端调用。
- iOS 版本 可通过 [TloTAppEnvironment.m](#) 的 `selectEnvironmentType` 方法中设置此 API 。
登录前所使用的 API URL 在 `environment.oemAppApi` 配置，请务必替换成自建的后台服务地址。

```
- (void)selectEnvironmentType {
    TloTAppConfigModel *model = [TloTAppConfig loadLocalConfigList];
    TloTCoreAppEnvironment *environment = [TloTCoreAppEnvironment shareEnvironment];
    [environment setEnvironment];
    environment.appKey = model.TencentIotLinkAppkey;
    environment.appSecret = model.TencentIotLinkAppSecret;
    // 请在 [environment setEnvironment]; 之后设置 oemAppApi 以免被覆盖。
    environment.oemAppApi = @"需要替换为自建后台服务地址";
}
```

[TloTCoreAppEnvironment.m](#) 的 `setEnvironment`方法中默认 API 配置说明。

```
- (void)setEnvironment {
    //公版&开源体验版使用 当在 app-config.json 中配置 TencentIotLinkAppkey TencentIotLinkAppSecret
    后，将自动切换为 OEM 版本。
    self.studioBaseUrl = @"https://iot.cloud.tencent.com/api/studioapp";
    self.studioBaseUrlForLoggedIn = @"https://iot.cloud.tencent.com/api/studioapp/tokenapi";
    //OEM App 使用
    self.oemAppApi = @"https://iot.cloud.tencent.com/api/exploreropen/appapi"; // 需要在 TloTAppEn
    vironment.m 的 -selectEnvironmentType: 中替换为自建后台服务地址。
    self.oemTokenApi = @"https://iot.cloud.tencent.com/api/exploreropen/tokenapi"; // 可安全在设备端
    调用。
    self.wsUrl = @"wss://iot.cloud.tencent.com/ws/explorer";
    self.h5Url = @"https://iot.cloud.tencent.com/explorer-h5";
}
```

- Android 版本 可通过 [HttpRequest](#) 中设置API 。
登录前所使用的 API URL 在 `OEM_APP_API` 配置，请务必替换成自建的后台服务地址。

```
/*
接口请求文件
*/
class HttpRequest private constructor() {
    companion object {
```

```
// 公版&开源体验版使用 当在 app-config.json 中配置 TencentIotLinkAppkey TencentIotLinkAppSecret
后，将自动切换为 OEM 版本。
const val STUDIO_BASE_URL = "https://iot.cloud.tencent.com/api/studioapp"
const val STUDIO_BASE_URL_FOR_LOGINED = "https://iot.cloud.tencent.com/api/studioapp/token
api"
// OEM App 使用
const val OEM_APP_API = "https://iot.cloud.tencent.com/api/exploreropen/appapi" // 需要替换为自
建后台服务地址
const val OEM_TOKEN_API = "https://iot.cloud.tencent.com/api/exploreropen/tokenapi" // 可安全在
设备端调用。
const val APP_COS_AUTH = "https://iot.cloud.tencent.com/api/studioapp/AppCosAuth"
const val BUSI_APP = "studioapp"
const val BUSI_OPENSOURCE = "studioappOpensource"
}
}
```

App Demo、SDK Demo 和 SDK 的关系

iOS:

工程中已经包含 [App Demo](#)、[SDK Demo](#) 和 [SDK](#)，无需额外引入。

Android:

工程中已经包含 [App Demo](#)、[SDK Demo](#) 和 [SDK](#)，无需额外引入。

部署流程

最近更新时间：2021-03-12 14:42:16

本文为您详细介绍第三方定制 App 部署相关操作。

前提条件

- 注册 [腾讯云账号](#) 并完成 [实名认证](#)。
- 登录腾讯云 [物联网开发平台控制台](#)，并注册 [移动应用](#)。

App 参数写入配置文件

- iOS 平台的配置文件为 [app-config.json](#)
- Android 平台的配置文件为 [app-config.json](#)

请根据实际情况调整配置信息，配置文件内容如下：

```
{
  "TencentIotLinkAppkey": "请输入从物联网开发平台申请的 App key，正式发布前务必填写",
  "TencentIotLinkAppSecret": "请输入从物联网开发平台申请的 App Secret，App Secret 请保存在服务端，此处  
仅为演示，如有泄露概不负责",
  "WXAccessAppld": ""
}
```

1. TencentIotLinkAppkey 和 TencentIotLinkAppSecret 请使用物联网平台申请的 [App Key](#) 和 [App Secret](#)，TencentIotLinkAppSecret 存放在用户自建后台服务器中，详情可见[应用端 API 简介](#)。
2. WXAccessAppld 请使用微信开放平台申请的 [AppID](#)，微信开发平台申请的 AppSecret 需要存放在用户自建的接入服务器中，用于获取 [微信登录信息](#)。

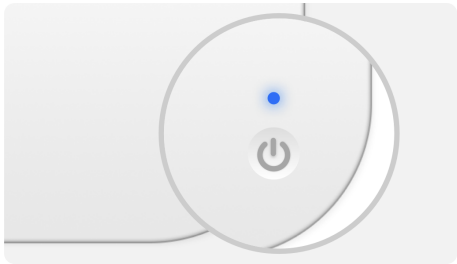
App 绑定真实设备

- 通过 [softAP 配网](#) 方式（自助配网）绑定真实设备，操作步骤如下：
 - i. 点击首页添加设备按钮，跳转到添加设备页面后，点击需要添加的硬件设备。

ii. App 自动识别设备配网为 softAP 配网方式，如图：



将设备设置为热点配网模式



1. 接通设备电源。
2. 长按复位键（开关），切换设备配网模式到热点配网（不同设备操作方式有所不同）。
3. 指示灯慢闪即进入热点配网模式。

下一步

iii. 点击底部按钮后，连接 Wi-Fi，输入 Wi-Fi 名称和密码，如图：

中国移动

下午 5:20

92%

<

热点配网

1

2

3

4

配置硬件

设置目标WiFi

连接设备

开始配网

请输入WiFi密码

WiFi

Tencent-WiFi

▼

密码

请输入密码

下一步

iv. 输入完后点击确认，连接设备热点，如图：



将手机WiFi连接设备热点



请前往手机WiFi设置界面，连接上图所示设备WiFi

下一步

v. 在【设置】中选择设备提供的热点，进行连接。

vi. 连接成功后，返回 App，如图：



vii. 点击下一步开始连接，连接成功后，如图：

10:34



热点配网



配网完成,设备添加成功

设备名称:PTXSW1_195B3F

继续添加其他设备

完成

- 通过 [SmartConfig 配网](#) 方式（智能配网）绑定真实设备，操作步骤如下：

 注意：

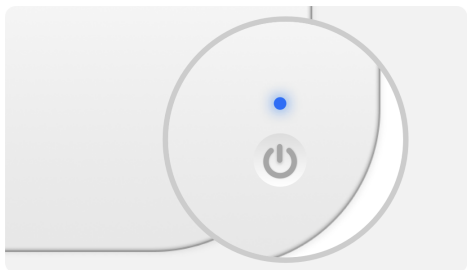
在此配网方式下，目前只支持 Wi-Fi 2.4GHZ。

1. 点击首页添加设备按钮，跳转到添加设备页面后，点击需要添加的硬件设备。

2. App 自动识别设备配网为 SmartConfig 配网方式，如图：



将设备设置为一键配网模式



1. 接通设备电源。
2. 长按复位键（开关），切换设备配网模式到一键配网（不同设备操作方式有所不同）。
3. 指示灯快闪即进入一键配网模式。

下一步

3. 点击底部按钮后，连接 Wi-Fi，输入 Wi-Fi 名称和密码，如图：

中国移动

下午 4:50

96%

<

一键配网

1

2

3

配置硬件

选择目标WiFi

开始配网

请输入WiFi密码

WiFi

Tencent-WiFi

▼

密码

请输入密码

下一步

4. 连接成功后，如图：



App 通过上述方式绑定真实设备后，设备状态会自动上报至物联网平台；手机端通过与物联网平台维持的长连接，可以实现实时显示设备状态。

第三方服务接入指南

最近更新时间：2020-12-15 15:22:14

本文档为您介绍开源版 App 可接入的第三方服务，包括腾讯移动推送、第三方登录（微信登录）和 Firebase 上报 Crash 信息。第三方服务为可选服务，可根据项目需要进行配置使用。

接入腾讯移动推送

步骤1：创建移动推送应用

1. 登录 [腾讯移动推送平台](#)。
2. 选择【产品管理】，单击【新增产品】。



3. 进入到产品新增界面，填写产品相关信息，并勾选“Android”、“iOS”平台，单击【确定】。

新增产品



腾讯移动推送提供专业技术团队配合接入测试，请微信添加技术支持：
TPNS_Push

产品名称

请输入产品名称

产品描述

请输入产品描述

产品分类

请选择产品分类

服务接入点

请选择服务接入点

平台



Android



iOS



macOS

确定

取消

- 产品名称：支持汉字、字母、数字，长度限制30个字符。
- 产品描述：支持汉字、字母、数字，长度限制60个字符。
- 产品分类：可根据您的 App 项目进行选择工具、生活等分类。
- 接入服务点：支持选择广州、上海、香港、新加坡。

4. 新增产品完成后，将出现【查看新手指引】弹窗，可忽略直接关闭，并按照以下步骤继续执行。

新增产品

✓ 新增产品成功

✓ 新增Android应用成功

✓ 新增iOS应用成功

点击下方按钮快速开始接入测试

查看新手指引

5. 在已创建的产品中：

- i. 选择 Android 平台所在行右侧的【配置管理】，进入应用信息界面，填写“应用包名”并单击【保存】。

应用信息

应用包名

请输入应用包名，例如com.tpns.demo

上方包名将作为该应用主包名，推送时默认向该包名下发。若有多个渠道包名，可[点击添加](#)或稍后在配置管理页面添加

保存

- ii. 选择 iOS 平台所在行右侧的【配置管理】，进入应用信息界面，填写 BundleID 并单击【保存】。

应用信息

BundleID

请输入应用BundleID

保存

BundleID 和应用包名保存后，即可获得到对应的 Access ID 和 SecretKey。

6. 上传 iOS 平台对应的证书，详情可参见 [推送证书获取指引](#)。

推送证书

开发环境

证书状态

暂无推送证书 [上传证书](#) 可上传.p12格式的开发环境证书或生产&开发合并环境证书

过期时间

--

生产环境

证书状态

暂无推送证书 [上传证书](#) 可上传.p12格式的生产环境证书或生产&开发合并环境证书

过期时间

--

证书获取指引

[查看指引](#)

温馨提示

若您持有合并环境证书，且需要在生产环境和开发环境同时使用推送服务，则需要上方两个「上传证书」入口分别上传一次合并证书

步骤2：推送配置

1. 登录 [物联网开发平台控制台](#)，选择 [新建项目](#) 或者进入已有项目。

2. 选择左侧导航菜单【应用开发】，单击【新建应用】并填写应用信息，单击【保存】。

新建应用



应用名称 *

支持中文、英文、数字、下划线的组合，最多不超过50个字符

备注

选填

最多不超过1024个字符

保存

取消

- 应用名称：必填，支持中文、英文、数字、下划线的组合，最多不超过50个字符。
- 备注：非必填，最多不超过1024个字符。

3. 创建完成后，单击【推送配置】进入推送配置页面，填写在 [步骤1](#) 所选择的服务接入点以及获取到 Access ID、SecretKey。

应用推送配置

📘 请配置您需要的信息，若iOS或者Android只需要配置其中之一，请保持另一端配置内容为空。

测试-iOS

AccessID

请填写腾讯移动推送平台获取的 AccessID

SecretKey

请填写腾讯移动推送平台获取的 SecretKey

选择环境

生产环境 ▼

根据您的应用的需要选择生产环境或开发环境进行使用或调试

服务器接入点

广州 ▼

接口请求地址与服务器接入点——对应，请选择您在腾讯移动推送平台接入的服务点

测试-Android

AccessID

请填写腾讯移动推送平台获取的 AccessID

SecretKey

请填写腾讯移动推送平台获取的 SecretKey

服务器接入点

广州 ▼

接口请求地址与服务器接入点——对应，请选择您在腾讯移动推送平台接入的服务点

保存

取消

⚠ 注意：

- 请认真核对 Access ID、SecretKey 以及对应的服务接入点，若信息填写有误将导致配置不成功。
- iOS 和 Android 的推送配置可以只选择其一，请在填写配置时保持另一端内容为空。

4. 填写完成后单击【保存】即可生效。

步骤3：配置工程文件

将 [步骤1](#) 获取的 **AccessID** 和 **AccessKey** 写入对应的工程文件中：

```
{
  "WXAccessAppId": "",
  "TencentlotLinkAppkey": "",
  "TencentlotLinkAppSecret": "",
  "XgAccessId": "",
  "XgAccessKey": ""
}
```

② 说明：

- TencentlotLinkAppkey 和 TencentlotLinkAppSecret 的参数获取，详情请参见 [接入指南](#)。
- 若不使用推送功能，将 **XgAccessId** 和 **XgAccessKey** 设置为长度为0的字符串即可。

接入微信登录

步骤1：登录微信开放平台

登录 [微信开放平台](#)。

步骤2：创建应用

在微信开放平台中的移动应用模块下创建应用，填写对应信息，审核通过后，即可获取 AppID 和 AppSecret；请务必保存好 AppID 和 AppSecret，在 SDK 接入配置中需要用到。微信开发平台申请的 AppSecret 需要存放在用户自建的后台服务器中，用于获取 [微信登录信息](#)。

步骤3：搭建后台服务

[自行搭建后台服务](#)，请求微信开放平台，实现微信登录。

步骤4：填入工程

将配置项 **WXAccessAppId** 设置为在微信开放平台申请获取的 **AppID** 和 **AppSecret**；同时请遵从官方建议自建微信接入服务器，保证 AppSecret 不被泄露；在 app-config.json 文件中修改 **WXAccessAppId** 配置项，示例代码如下：

```
{
  "WXAccessAppId": "",
  "TencentlotLinkAppkey": "",
  "TencentlotLinkAppSecret": "",
  "XgAccessId": "",
  "XgAccessKey": ""
}
```

```
"XgAccessKey": "",
"XgUSAAccessId": "",
"XgUSAAccessKey": ""
}
```

❓ 说明：

若不使用微信授权登录功能，将 **WXAccessAppld** 设置为**长度为0字符串**即可。

另外，实现微信授权登录还需：

- iOS工程：

将配置项 **LinkAPP_WEIXIN_APPID** 设置为在微信开放平台申请并获得的 AppID；**同时请遵从官方建议自建微信接入服务器，保证 AppSecret 不被泄露**；在工程中 Supporting Files 目录下的 LinkApp.xcconfig 文件中修改 **LinkAPP_WEIXIN_APPID** 配置项，示例代码如下：

```
{
  LinkAPP_NAME =
  LinkAPP_WEIXIN_APPID = ""
  LinkAPP_BUNDLE_TEAM =
  LinkAPP_BUNDLE_SIGN =
  LinkAPP_BUNDLE_ID =
  LinkAPP_BUNDLE_PROVISIONING =
}
```

- Android工程：

- i. 将 `opensource_keystore.jks` 文件替换成自己的签名文件并给应用签字。
- ii. 前往 [微信开放平台](#) 下载签名生成工具，使用该工具生成应用的数字签名（需要将该工具和应用同时安装到手机上，打开签名生成工具输入应用包名即可生成数字签名）。
- iii. 将该数字签名和应用包名登记到微信开放平台，否则微信授权登录将不可用。

接入 Firebase 上报 Crash 信息

接入 Firebase 上报 Crash 信息可方便用户分析和排查问题。具体操作步骤详情请参见如下指引：

- iOS：[快速开始 Firebase 部分](#)。
- Android：[快速开始 Firebase 部分](#)。

App 发布前确认事项

最近更新时间：2021-03-12 14:40:24

iOS

- 确认填写用户自己的 Bundle ID 和对应的发布证书。
- 根据实际情况调整配置文件中的内容，需要配置的内容可参考 [App 参数写入配置文件](#)，获取到相应的 ID 和 key 值替换 App 中的字符串。
- 如果用户确认接入 Firebase，用户需要使用从 Firebase 官网自建应用获得 **GoogleService-Info.plist**，替换 App 中 **GoogleService-Info.plist** 文件。

Android

1. 请根据实际情况调整 **app-config.json** 中的内容。app-config.json 需要配置的内容，如下所示。

```
{
  "TencentlotLinkAppkey": "请输入从物联网开发平台申请的 App Key，正式发布前务必填写",
  "TencentlotLinkAppSecret": "请输入从物联网开发平台申请的 App Secret，App Secret 请保存在服务端，此处仅为演示，如有泄露概不负责",
  "WXAccessAppId": ""
}
```

- **TencentlotLinkAppkey** 和 **TencentlotLinkAppSecret** 请使用在物联网平台创建应用时生成的 **App Key** 和 **App Secret**，**TencentlotLinkAppSecret** 存放在用户自建后台服务器中，详情可参见 [应用端 API 简介](#)。
- **WXAccessAppId** 请使用在微信开放平台申请并获得的 **AppID**；若确认使用自定义的微信授权登录，需要在 [微信开放平台](#) 注册开发者帐号，创建移动应用，审核通过后，即可获得相应的 **AppID** 和 **AppSecret**。

使用微信授权登录还需：

- i. 将 **opensource_keystore.jks** 文件替换成自己的签名文件并给应用签字。
- ii. 前往 [微信开放平台](#) 下载签名生成工具，使用该工具生成应用的数字签名(需要将该工具和应用同时安装到手机上，打开签名生成工具输入应用包名即可生成数字签名)
- iii. 将该数字签名和应用包名登记到微信开放平台，否则微信授权登录将不可用。
- iv. 最后将配置项 **WXAccessAppId** 设置为在微信开放平台申请并获得的 **AppID**；同时请遵从官方建议自建微信接入服务器，保证 **AppSecret** 不被泄露。

2. 项目配置 **Firebase** 插件。

- 若用户确认使用 **Firebase** 功能，需通过 **Firebase** 官网创建应用并获取 **google-services.json**，替换项目中的 **google-services.json** 文件。
- 若不使用 **Firebase** 功能，需要在以下文件中注释掉对应依赖。
 - 在项目级 **build.gradle** 中注释掉以下三个依赖项。


```
dependencies {  
    // classpath 'com.google.gms:google-services:4.3.3'  
    // classpath 'com.google.firebase:firebase-crashlytics-gradle:2.1.1'  
    // classpath 'com.google.firebase:perf-plugin:1.3.1'  
}
```

- 在应用级 **build.gradle** 中注释掉以下三个应用插件和三个依赖项。

```
//apply plugin: 'com.google.gms.google-services'  
//apply plugin: 'com.google.firebase.crashlytics'  
//apply plugin: 'com.google.firebase.firebase-perf'  
dependencies {  
    // implementation 'com.google.firebase:firebase-analytics-ktx:17.4.3'  
    // implementation 'com.google.firebase:firebase-crashlytics:17.0.1'  
    // implementation 'com.google.firebase:firebase-perf:19.0.7'  
}
```

基于 App SDK

概述

最近更新时间：2021-12-22 10:03:39

为进一步构建物联网开放生态，由腾讯云物联网平台打造的腾讯连连 App SDK，集成通用版 App 的多功能模块。设备厂商可通过 SDK 将设备接入腾讯云物联网平台进行设备管理，涵盖家用电器、运动健康、网络设备等众多设备。

目前，腾讯连连 App SDK 的核心模块包含设备配网的两种模式，分别是 SmartConfig 与 Soft AP；其他模块包括设备消息操作、账户系统、设备管理等。

业务包

App SDK 位于 [iOS 工程](#) 或 [Android 工程](#) 目录中，包含了 SDK Demo 演示工程，用户可根据 SDK Demo 快速接入 App SDK。

同时也可通过已开源 [iOS 腾讯连连工程](#) 或 [Android 腾讯连连工程](#) 快速搭建起自己的物联网智能设备管理 App。

App SDK版本更新

V1.3.0

- 发布日期：2021/01/15
- 系统平台：iOS/Android
- 开发语言：OC 语言/Java
- 开发环境：Mac、Xcode/Android studio
- 更新内容：
 - 新增扫一扫产品落地详情页，支持控制台详情页配置。
 - 优化实时音视频信令逻辑，提升用户体验。
 - 修复 App H5 自定义控制面板的已知问题。
 - App 主要页面、按钮风格优化。
 - 设备绑定优化，为用户自动填写默认设备名称。

V1.2.1

- 发布日期：2020/12/10
- 系统平台：iOS/Android
- 开发语言：OC 语言/Java
- 开发环境：Mac、Xcode/Android studio
- 更新内容：修复 V1.2.0 版本的 SDK 配置，删除 i386 架构。

V1.2.0

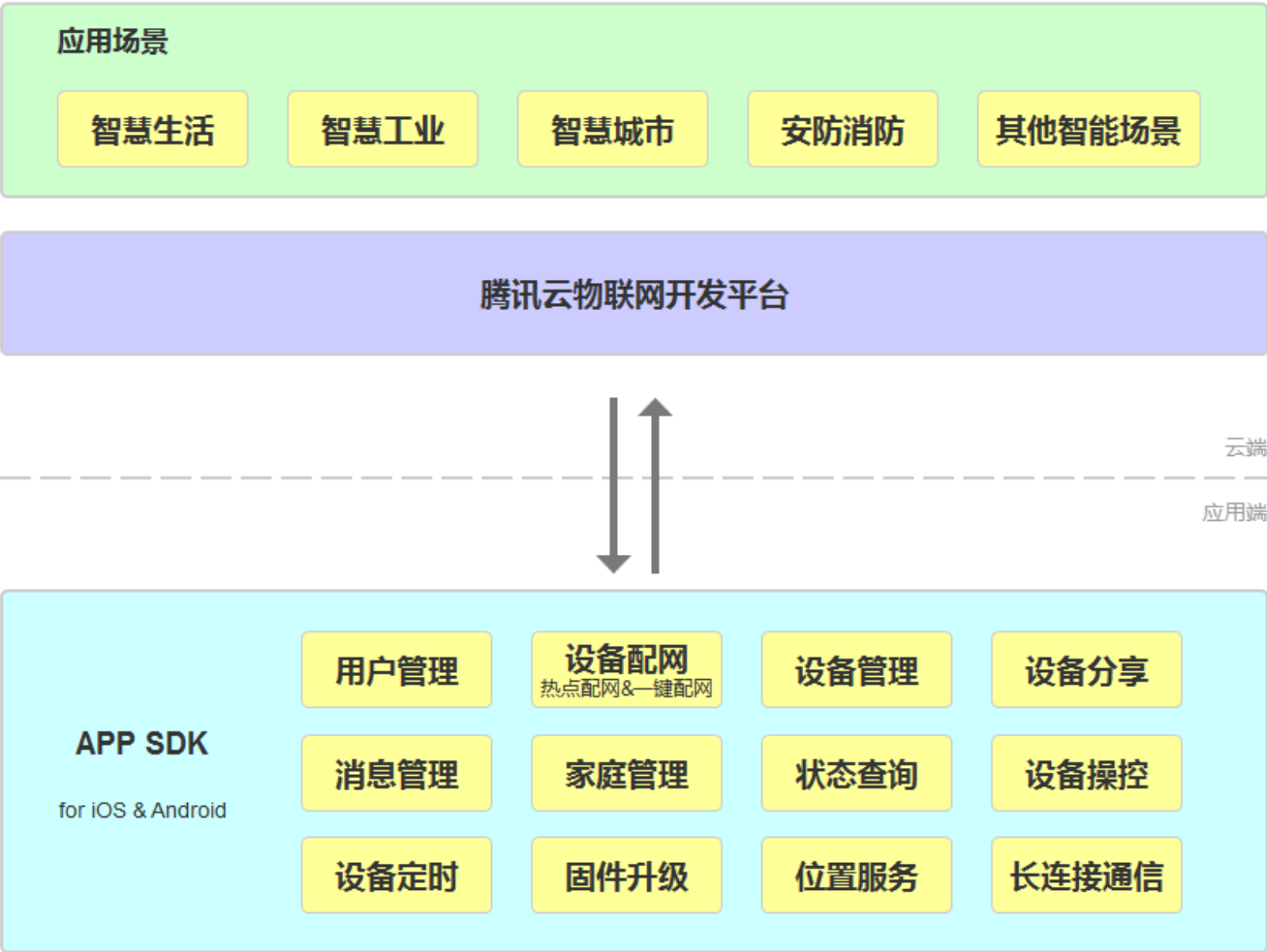
- 发布日期：2020/12/02
- 系统平台：iOS/Android
- 开发语言：OC 语言/Java
- 开发环境：Mac、Xcode/Android studio
- 更新内容：提供接入实时音视频通话场景需求能力。

V1.1.0

- 发布日期：2020/10
- 系统平台：iOS/Android
- 开发语言：OC 语言/Java
- 开发环境：Mac、Xcode/Android studio
- 更新内容：
 - 提供 SoftAP、SmartConfig 配网管理。
 - 实现设备管理、设备分享、设备定时。
 - 支持家庭管理、消息管理。
 - 实现长链接通信获取设备状态。

SDK 的依赖关系

在腾讯云物联网平台中，App SDK 扮演的角色如图所示。App 通过接入 App SDK 来实现与智能设备的配网，并通过物联网平台对智能设备进行管理。目前 App SDK 中与设备配网方式提供 [SmartConfig 配网](#) 和 [SoftAP 配网](#) 模式。



接入指南

最近更新时间：2021-03-12 14:39:32

App SDK 模块说明

iOS

子模块	实现相关功能
QCDeviceCenter	配网模块
QCAPISets	设备控制、消息相关、家庭管理、账户管理等 API
QCFoundation	工具类

Android

子模块	实现相关功能
link	配网模块
auth	设备控制、消息相关、家庭管理、账户管理等 API
utils	工具类
log	日志模块

SDK 接入详情

接入前 API 参数对照表

基础参数对照表：

参数名称	参数说明
phoneNumber	手机号
countryCode	国际区号，如中国大陆区号为86
email	邮箱地址
familyId	家庭 ID
familyName	家庭名称
familyAddress	家庭地址

参数名称	参数说明
Role	1是所有者， 0是普通成员
roomId	房间 ID
roomName	房间名
ProductId	设备产品 ID
Avatar	用户信息中头像链接
signature	使用绑定设备 API 时传入，设备签名
DeviceId	设备 ID

设备控制面板列表参数对照表：

参数名称	参数说明
ID	设备可控属性
name	设备可控属性名，如” 电源开关”、“颜色”
big	设备可控属性，面板按钮是否是大按钮
type	设备可控属性，面板按钮类型，如： btn-big、btn-col-1
value	属性值
familyAddress	家庭地址
LastUpdate	最后一次更新时间戳

SDK 使用举例

```
#iOS举例：设备配网接入
#1.创建配网对象QCSmartConfig或者QCSoftAP（视配网方式决定），注：SDK 内不持有配网对象，需使用者自己持有
self.sc = [[QCSmartConfig alloc] initWithSSID:name PWD:password BSSID:bssid];
self.sc.delegate = self;
#2.遵循TloTCoreAddDeviceDelegate协议，设置代理，并接入代理方法：
- (void)onResult:(QCResult *)result{
if (result.code == 0) { // 配网成功
}
else { // 配网失败
}
}
```

```
}  
#3.开始配网流程  
[self.sc startAddDevice];
```

更多功能

账户系统、家庭相关操作、设备控制等详细操作，请参考 SDK Demo 工程 [iOS 版本 LinkSDKDemo](#) 或 [Android 版本 sdkdemo](#)。

详细接口对照表

物联网应用开发文档	对应文档地址
应用端 API	应用端 API 文档
用户管理	用户管理文档
配网管理	配网管理文档
设备管理	设备管理文档
设备分享	设备分享文档
家庭管理	家庭管理文档
设备定时	设备定时文档
消息管理	消息管理文档
长连接通信	长连接通信文档
数据结构	数据结构文档

若接入过程中有其他问题，请参考 [基于 SDK 常见问题](#)

创建引导

最近更新时间：2021-07-15 15:06:00

准备工作

获取在物联网开发平台创建应用时生成的 [App Key](#) 和 [App Secret](#)。

⚠ 注意：

签名算法务必在服务端实现，腾讯连连 App 开源版的使用方式仅为演示，登录鉴权请参考 [自建服务](#)。

iOS 安装环境

- 安装 [Xcode 开发工具](#)。
- 集成 SDK 方式。

执行如下命令采用 Cocoapods 方式集成。

```
pod TloTLinkKit
```

- 手动集成。
- 将 [LinkCore](#) 目录拖入到工程中。

Android 安装环境

- 安装 [Android Studio](#) 开发工具
- 集成 SDK 方式
 - 依赖 maven 远程构建

```
implementation 'com.tencent.iot.explorer:explorer-link-android:1.0.0'
```

- 依赖本地 SDK 源码 构建

修改应用模块的 [build.gradle](#)，使应用模块依赖 [SDK](#) 源码，示例如下：

```
implementation project(path: ':sdk')
```

使用说明

1. 导入配置，初始化 SDK

- iOS

参考 Source/LinkSDKDemo/Supporting Files/AppDelegate.m 文件，并配置物联网开发平台申请的 [App Key](#)

和 App Secret。

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {  
    /*  
    * 此处仅供参考, 需自建服务接入物联网平台服务, 以免 App Secret 泄露  
    * 自建服务可参考此处 https://cloud.tencent.com/document/product/1081/45901#.E6.90.AD.E5.BB.BA.E5.90.8E.E5.8F.B0.E6.9C.8D.E5.8A.A1.2C-.E5.B0.86-app-api-.E8.B0.83.E7.94.A8.E7.94.B1.E8.AE.BE.E5.A4.87.E7.AB.AF.E5.8F.91.E8.B5.B7.E5.88.87.E6.8D.A2.E4.B8.BA.E7.94.B1.E8.87.AA.E5.BB.BA.E5.90.8E.E5.8F.B0.E6.9C.8D.E5.8A.A1.E5.8F.91.E8.B5.B7  
    */  
    TloTCoreAppEnvironment *environment = [TloTCoreAppEnvironment shareEnvironment];  
    [environment setEnvironment];  
    environment.appKey = @"物联网开发平台申请的 App Key";  
    environment.appSecret = @"物联网开发平台申请的 App Secret";  
    self.window.rootViewController = [[UINavigationController alloc] initWithRootViewController:[UI  
    ViewController new]];  
    return YES;  
}
```

◦ Android

选择 sdkdemo/src/main/java/com/tencent/iot/explorer/link/core/demo/App.java, 配置 App key。

```
class App : Application() {  
    companion object {  
        val data = AppData.instance  
    }  
    private val APP_KEY = "物联网开发平台申请的 App Key"  
    private val APP_SECRET = "物联网开发平台申请的 App Secret"  
    override fun onCreate() {  
        super.onCreate()  
        /*  
        * 此处仅供参考, 需自建服务接入物联网平台服务, 以免 App Secret 泄露  
        * 自建服务可参考此处 https://cloud.tencent.com/document/product/1081/45901#.E6.90.AD.E5.BB.BA.E5.90.8E.E5.8F.B0.E6.9C.8D.E5.8A.A1.2C-.E5.B0.86-app-api-.E8.B0.83.E7.94.A8.E7.94.B1.E8.AE.BE.E5.A4.87.E7.AB.AF.E5.8F.91.E8.B5.B7.E5.88.87.E6.8D.A2.E4.B8.BA.E7.94.B1.E8.87.AA.E5.BB.BA.E5.90.8E.E5.8F.B0.E6.9C.8D.E5.8A.A1.E5.8F.91.E8.B5.B7  
        */  
        IoTAuth.openLog(true)  
        IoTAuth.init(APP_KEY, APP_SECRET)  
    }  
    override fun onTerminate() {
```

```
super.onTerminate()
}
}
```

2. App SDK 功能划分说明

◦ iOS

iOS 对应模块	实现相关功能
QCDeviceCenter	配网模块
QCAPISets	设备控制、消息相关、家庭管理、账户管理等 API
QCFoundation	工具类

◦ Android

Android 子模块	实现相关功能
link	配网模块
auth	设备控制、消息相关、家庭管理、账户管理等 API
utils	工具类
log	日志模块

3. 账户相关接口，包含手机号、邮箱注册，登入登出，密码操作，用户信息操作。账户详细接口请参考 [官方文档](#)，或者 App SDK 文件中 [iOS 文件（TloTCoreAccountSet.h）](#) 或 [Android 文件（IoTAuth.kt）](#)。

⚠ 注意：

此处仅为 Demo 演示功能，请遵从官方建议自建账户后台服务后，由自建服务接入物联网平台服务，以免 App Secret 泄露。

4. 详细功能请参考 [App SDK 接入指南](#)

应用端 API

应用端 API 简介与签名介绍

最近更新时间：2021-11-26 10:03:39

简介

应用端 API 是开发平台为了满足智能家居场景，为用户开发自有品牌的小程序或 App 而提供的云端服务，用户无需实现用户管理、设备管理、设备定时、家庭管理等基础能力，可通过调用应用端 API 快速完成移动应用端的开发。

调用方式

1. 登录前所使用的 API URL 为 <https://iot.cloud.tencent.com/api/exploreropen/appapi>。

其中公共参数有：Action，RequestId，AppKey，Signature，Timestamp，Nonce。其中 Action 用于标识请求的方法名称；RequestId 用于标识一个唯一请求，推荐使用 uuid 作为参数值，定位问题时建议提供该参数值；AppKey 为应用的密钥；Signature 为本次请求的签名，具体计算方法见本文下方示例；Timestamp 为本次请求的 UNIX 秒级时间戳；Nonce 为随机正整数，用于和时间戳一起，防范 API 重放攻击。

2. 登录后所使用的 API URL 为 <https://iot.cloud.tencent.com/api/exploreropen/tokenapi>。

其中公共参数有：Action，RequestId，AccessToken。其中 Action 用于标识请求的方法名称；RequestId 用于标识一个唯一请求，推荐使用 uuid 作为参数值，定位问题时建议使用该参数值。AccessToken 用于标识一个已经登录的用户。

登录前，要通过相关用户接口换取 accesstoken 完成登录，调用 url 为 [.../appapi](#)，accesstoken 用于标识一个用户。当用户登录完毕后，使用 url 为 [.../tokenapi](#) 的相关 API 完成其他操作。

签名算法

获取应用 AppKey 和 AppSecret

如果用户不使用腾讯官方的“腾讯连连”小程序，用户也可通过平台开放能力开发自有品牌小程序。在创建应用的时候，平台会为用户生成小程序对应的安全凭证。安全凭证包括 AppKey 和 AppSecret。AppKey 是用于标识 API 调用者身份，AppSecret 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。**用户应严格保管其 AppSecret，避免泄露。**

具体获取步骤如下：

1. 登录 [物联网开发平台控制台](#)，进入开发中心。
2. 选择左侧菜单【应用开发】>【小程序开发】，新建小程序，具体新建步骤参见 [应用开发](#)。
3. 创建小程序成功后，即可获取系统自动生成的 AppKey 与 AppSecret。

生成签名串

有了安全凭证 AppKey 和 AppSecret 后，就可以生成签名串了。下面给出了一个生成签名串的详细过程。

假设用户的 AppKey 和 AppSecret 分别是：

- AppKey: ahPxdK****TGrejd
- AppSecret: NcbHqk****TCGbKnQH

❓ 说明：

本文仅为示例，请您根据自己实际的 AppKey 和 AppSecret 进行后续操作。

以通过手机号注册账号 AppCreateCellphoneUser 请求为例，当用户调用这一接口时，其请求参数可能如下：

参数名称	类型	描述	参考数值
RequestId	String	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需要提供该次请求的 RequestId	8b8d499bbba1ac28b6da21b4
Action	String	公共参数，调用的接口方法名称	AppCreateCellphoneUser
AppKey	String	公共参数，应用 AppKey，用于标识对应的 App	ahPxdK****TGrejd
Signature	String	公共参数，请求的签名	Szxai9Qs7****OXahbFbseZ+uE=
Timestamp	Int64	公共参数，当前的 UNIX 时间戳（秒级）	1546315200
Nonce	Int	公共参数，随机正整数，与时间戳一起，用于 API 防重放	71087795
CountryCode	String	国家区码	86
PhoneNumber	String	手机号码	13900000000
Password	String	密码	password
VerificationCode	String	短信验证码	123456

❓ 说明：

- 请求参数中的公共请求参数有：RequestId、Action、AppKey、Timestamp、Nonce、Signature。
- AppCreateCellphoneUser 接口特有参数：CountryCode、PhoneNumber、Password、VerificationCode。

而参数 Signature（签名串）正是由上述参数共生成的，具体步骤如下：

1. 对参数排序

对所有请求参数按参数名做字典序升序排列，所谓字典序升序排列，直观上就如同在字典中排列单词一样排序，按照字母表或数字表里递增顺序的排列次序，即先考虑第一个“字母”，在相同的情况下考虑第二个“字母”，依此类推。您可以借助编程语言中的相关排序函数来实现这一功能，例如 PHP 中的 `ksort` 函数。上述示例参数的排序结果如下：

```
{
  Action=AppCreateCellphoneUser,
  AppKey=ahPxdK****TGrejd,
  CountryCode=86,
  Nonce=71087795,
  Password=My!P@ssword,
  PhoneNumber=13900000000,
  RequestId=8b8d499bbba1ac28b6da21b4,
  Timestamp=1546315200,
  VerificationCode=123456
}
```

使用其它程序设计语言开发时，可对上面示例中的参数进行排序，得到的结果一致即可。

2. 拼接请求字符串

将把上一步排序好的请求参数格式化成“参数名称”=“参数值”的形式，如对 `Action` 参数，其参数名称为 `"Action"`，参数值为 `"AppCreateCellphoneUser"`，因此格式化后即为 `Action=AppCreateCellphoneUser`。

- “参数值”为原始值而非 URL 编码后的值。
- 若输入参数中的“键”包含下划线，则需要将其转换为“.”。对于“值”不需要额外操作。。

将格式化后的各个参数用“&”拼接在一起，最终生成的请求字符串为：

```
Action=AppCreateCellphoneUser&AppKey=ahPxdK****TGrejd&CountryCode=86&Nonce=71087795&
Password=My!P@ssword &PhoneNumber=13900000000&RequestId=8b8d499bbba1ac28b6da21b4&
Timestamp=1546315200&VerificationCode=123456
```

3. 生成签名串

使用 HMAC-SHA1 算法对上一步中获得的签名原字符串进行签名，然后将生成的签名串使用 Base64 进行编码，即可获得最终的签名串。

具体代码如下，以 PHP 语言为例：

```
$secretKey = 'NcbHqk****TCGbKnQH';
$srcStr = 'Action=AppCreateCellphoneUser&AppKey=ahPxdK****TGrejd&CountryCode=86&Nonce=71087795&Password=My!P@ssword&PhoneNumber=13900000000&RequestId=8b8d499bbba1ac28b6da21b4&Timestamp=1546315200&VerificationCode=123456';
```

```
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));  
echo $signStr
```

最终得到的签名串为：

```
Szxai9Qs7O3lBoOXahbFbseZ+uE=
```

使用其它程序设计语言开发时，可用上面示例中的原文进行签名验证，得到的签名串与例子中的一致即可。

常见问题

对于参数为字符串且为空的，应将其从签名串中省略，不需要将其参与签名。

调试工具

使用应用端 API [调试工具](#)，快速调试应用端 API。

用户管理

微信号注册登录

最近更新时间：2021-02-19 10:08:58

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppGetTokenByWeiXin）用于微信用户注册登录，获取开发平台的用户访问 Token，首次调用时，自动为该微信号注册对应账号。

2. 输入参数

名称	类型	必选	描述
Action	String	是	公共参数，本接口取值：AppGetTokenByWeiXin
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
AppKey	String	是	公共参数，应用 AppKey，用于标识对应的小程序或 App
Signature	String	是	公共参数，请求签名，需用户自行生成，用于校验请求的合法性
Timestamp	Int64	是	公共参数，请求的 UNIX 时间戳（秒级）
Nonce	Int	是	公共参数，随机正整数，与 Timestamp 联合起来，防止重放攻击
WxOpenID	String	是	微信用户的 OpenID 或 UnionID
NickName	String	是	昵称
Avatar	String	是	头像

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
ExpireAt	Int64	截止时间，UNIX 秒级时间戳
Token	String	开发平台返回的 AccessToken，通过该 Token 进行登录后的接口请求

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/appapi HTTP/1.1
content-type: application/json
{
  "WxOpenID": "0123456789",
  "NickName": "tests",
  "Avatar": "http://example.com/path/to/avatar.jpg",
  "Signature": "lY0yHX7zWCZKZ97xDonQoy16W4U=",
  "Timestamp": 1552621825,
  "Nonce": 2,
  "Action": "AppGetTokenByWeiXin",
  "AppKey": "ahPxdK*****TGrejd",
  "RequestId": "rest-client"
}
```

输出示例：成功

```
{
  "Response": {
    "Data": {
      "ExpireAt": 1556076201,
      "Token": "d2*****8514"
    },
    "RequestId": "rest-client"
  }
}
```

输出示例：失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.ErrorUserNotExists",
      "Message": "账号不存在"
    },
    "RequestId": "rest-client"
  }
}
```



```
}  
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
ErrorRequiredParamNotFound	必选参数缺失
InvalidAction	Action 非法
InvalidParameterValue	参数异常
InvalidParameterValue.InvalidJSON	请求格式不是 JSON
InvalidParameterValue.NickNameLengthInvalid	昵称长度非法

手机号注册用户

最近更新时间：2021-02-19 10:09:22

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppCreateCellphoneUser）提供手机号码方式的用户注册。

2. 输入参数

名称	类型	必选	描述
Action	String	是	公共参数，本接口取值：AppCreateCellphoneUser
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
AppKey	String	是	公共参数，应用 AppKey，用于标识对应的 App
Signature	String	是	公共参数，请求签名，需用户自行生成，用于校验请求的合法性
Timestamp	Int64	是	公共参数，请求的 UNIX 时间戳（秒级）
Nonce	Int	是	公共参数，随机正整数，与 Timestamp 联合起来，防止重放攻击
CountryCode	String	是	国家号
PhoneNumber	String	是	手机号
Password	String	是	登录密码
VerificationCode	String	是	短信验证码

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
UserID	String	用户 ID

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/appapi HTTP/1.1
content-type: application/json
{
  "CountryCode": "86",
  "PhoneNumber": "13900000000",
  "Password": "MyPassword",
  "VerificationCode": "123456",
  "Signature": "CKu55Y3ZD6RuxpjPySM6U99imbs=",
  "Timestamp": 1546315200,
  "Nonce": 71087795,
  "Action": "AppCreateCellphoneUser",
  "AppKey": "ahPxdk*****TGrejd",
  "RequestId": "8b8d499bbba1ac28b6da21b4"
}
```

输出示例：成功

```
{
  "Response": {
    "Data": { "UserID": "1234567890" },
    "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
  }
}
```

输出示例：失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.PhoneNumberInvalid",
      "Message": "号码错误"
    },
    "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
ErrorRequiredParamNotFound	必选参数缺失
InvalidAction	Action 非法
InvalidParameterValue	参数非法
InvalidParameterValue.InvalidJSON	请求格式不是 JSON
InvalidParameterValue.CheckVerifyCodeFailed	验证码错误
InvalidParameterValue.PhoneNumberUsed	电话号码已注册
InvalidParameterValue.PhoneNumberInvalid	号码错误

邮箱账号注册用户

最近更新时间：2021-02-19 10:10:06

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppCreateEmailUser）提供邮箱账号方式注册用户。

2. 输入参数

名称	类型	必选	描述
Action	String	是	公共参数，本接口取值：AppCreateEmailUser
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时需提供该次请求的 RequestId
AppKey	String	是	公共参数，应用 AppKey，用于标识对应的 App
Signature	String	是	公共参数，请求签名，需用户自行生成，用于校验请求的合法性
Timestamp	Int64	是	公共参数，请求的 UNIX 时间戳（秒级）
Nonce	Int	是	公共参数，随机正整数，与 Timestamp 联合起来，防止重放攻击
Email	String	是	用户的邮箱地址
Password	String	是	登录密码
VerificationCode	String	是	发送到邮箱的验证码

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
UserID	String	用户 ID

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/appapi HTTP/1.1
content-type: application/json
{
  "Email": "test@example.com",
  "Password": "My!P@ssword",
  "VerificationCode": "123456",
  "Signature": "CKu55Y3ZD6RuxpjPySM6U99imbs=",
  "Timestamp": 1546315200,
  "Nonce": 71087795,
  "Action": "AppCreateEmailUser",
  "AppKey": "ahPxdK*****TGrejd",
  "RequestId": "8b8d499bbba1ac28b6da21b4"
}
```

输出示例：成功

```
{
  "Response": {
    "Data": { "UserID": "1" },
    "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
  }
}
```

输出示例：失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.EmailUsed",
      "Message": "邮箱已注册"
    },
    "RequestId": "rest-client"
  }
}
```

5. 错误码

错误码	描述
InternalError	内部错误
InvalidParameterValue	参数非法
InvalidParameterValue.InvalidJSON	请求格式不是 JSON
InvalidParameterValue.EmailUsed	邮箱已注册
InvalidParameterValue.CheckVerifyCodeFailed	验证码错误

随机发送邮箱验证码

最近更新时间：2021-05-26 10:48:39

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppSendEmailVerificationCode）用于邮箱注册、绑定、重置密码和登录时，发送验证码。

2. 输入参数

名称	类型	必选	描述
Action	String	是	公共参数，本接口取值：AppSendEmailVerificationCode
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
AppKey	String	是	公共参数，应用 AppKey，用于标识对应的 App
Signature	String	是	公共参数，请求签名，需用户自行生成，用于校验请求的合法性
Timestamp	Int64	是	公共参数，请求的 Unix 时间戳（秒级）
Nonce	Int	是	公共参数，随机正整数，与 Timestamp 联合起来，防止重放攻击
Type	String	是	验证码类型。 • register：注册 • resetpass：重置密码 • login：登录
Email	String	是	邮箱

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
UserId	String	用户 ID

4. 示例

输入示例


```
POST https://iot.cloud.tencent.com/api/exploreropen/appapi HTTP/1.1
content-type: application/json
{
  "Type": "register",
  "Email": "someone@example.com",
  "Signature": "8tEbOa9wk0X3wTBn0BBoFWdgMjo=",
  "Timestamp": 1552621825,
  "Nonce": 2,
  "Action": "AppSendEmailVerificationCode",
  "AppKey": "ahPxd*****rejd",
  "RequestId": "f92406b3-5a9a-****-bc43-45e3d794bb68"
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": "OK",
    "RequestId": "f92406b3-5a9a-****-bc43-45e3d794bb68"
  }
}
```

输出示例: 失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.EmailInvalid",
      "Message": "邮箱错误"
    },
    "RequestId": "f92406b3-5a9a-****-bc43-45e3d794bb68"
  }
}
```

5. 错误码

错误码	描述

错误码	描述
InternalError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.EmailInvalid	邮箱错误
InvalidParameterValue.SendVerifyCodeTooQuick	验证码发送太频繁

随机发送手机短信

最近更新时间：2021-02-19 10:10:35

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppSendVerificationCode）用于手机号注册、绑定、重置密码时，发送验证码。

2. 输入参数

名称	类型	必选	描述
Action	String	是	公共参数，本接口取值：AppSendVerificationCode
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时需要提供该次请求的 RequestId
AppKey	String	是	公共参数，应用 AppKey，用于标识对应的 App
Signature	String	是	公共参数，请求签名，需用户自行生成，用于校验请求的合法性
Timestamp	Int64	是	公共参数，请求的 UNIX 时间戳（秒级）
Nonce	Int	是	公共参数，随机正整数，与 Timestamp 联合起来，防止重放攻击
Type	String	是	验证码类型：register 注册，resetpass 重置密码
CountryCode	String	是	国家代码
PhoneNumber	String	是	电话号码

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
UserID	String	用户 ID

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/appapi HTTP/1.1
content-type: application/json
{
  "Type": "register",
  "CountryCode": "86",
  "PhoneNumber": "13900000000",
  "Signature": "8tEbOa9wk0X3wTBn0BBoFWdgMjo=",
  "Timestamp": 1552621825,
  "Nonce": 2,
  "Action": "AppSendVerificationCode",
  "AppKey": "ahPxdK*****TGrejd",
  "RequestId": "rest-client"
}
```

输出示例：成功

```
{
  "Response": {
    "Data": "OK",
    "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
  }
}
```

输出示例：失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.PhoneNumberInvalid",
      "Message": "号码错误"
    },
    "RequestId": "rest-client"
  }
}
```

5. 错误码

错误码	描述
InternalError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.PhoneNumberInvalid	号码错误
InvalidParameterValue.SendVerifyCodeTooQuick	验证码发送太频繁

手机号、邮箱账号登录

最近更新时间：2021-02-19 10:14:57

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppGetToken）用于手机号码、邮箱账号登录，获取用户访问 Token。

2. 输入参数

名称	类型	必选	描述
Action	String	是	公共参数，本接口取值：AppGetToken
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
AppKey	String	是	公共参数，应用 AppKey，用于标识对应的 App
Signature	String	是	公共参数，请求签名，需用户自行生成，用于校验请求的合法性
Timestamp	Int64	是	公共参数，请求的 Unix 时间戳（秒级）
Nonce	Int	是	公共参数，随机正整数，与 Timestamp 联合起来，防止重放攻击
Type	String	否	账号类型，phone：手机号；email：邮箱
CountryCode	String	否	国家号
PhoneNumber	String	否	手机号
Email	String	否	邮箱
Password	String	是	登录密码

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
Data	String	响应结果

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/appapi HTTP/1.1
content-type: application/json
{
  "Email": "test@example",
  "Password": "123456",
  "Type": "email",
  "Signature": "w2kTSOU7IKc2aJ+UpIV34VHp1RI=",
  "Timestamp": 1552621825,
  "Nonce": 2,
  "Action": "AppGetToken",
  "AppKey": "ahPxdK*****TGrejd",
  "RequestId": "rest-client"
}
```

输出示例：成功

```
{
  "Response": {
    "Data": {
      "ExpireAt": 1556076201,
      "Token": "d2*****8514"
    },
    "RequestId": "rest-client"
  }
}
```

输出示例：失败

```
{
  "Response": {
    "Error": {
      "Code": "InternalServerError",
      "Message": "InvalidParameterValue.UserLoginFailed|用户登录失败"
    },
    "RequestId": "rest-client"
  }
}
```

5. 错误码

错误码	描述
InternalError	内部错误
ErrorRequiredParamNotFound	必选参数缺失
InvalidAction	Action 非法
InvalidParameterValue	参数非法
InvalidParameterValue.InvalidJSON	请求格式不是 JSON
InvalidParameterValue.UserLoginFailed	用户登录失败
InvalidParameterValue.UserLoginForbidden	账号被锁定

使用手机号重置密码

最近更新时间：2021-02-19 10:15:18

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppResetPasswordByCellphone）用于使用手机号重置密码。

2. 输入参数

名称	类型	必选	描述
Action	String	是	公共参数，本接口取值：AppResetPasswordByCellphone
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
AppKey	String	是	公共参数，应用 AppKey，用于标识对应的 App
Signature	String	是	公共参数，请求签名，需用户自行生成，用于校验请求的合法性
Timestamp	Int64	是	公共参数，请求的 Unix 时间戳（秒级）
Nonce	Int	是	公共参数，随机正整数，与 Timestamp 联合起来，防止重放攻击
CountryCode	String	是	手机号国家码
PhoneNumber	String	是	手机号码
VerificationCode	String	是	验证码
Password	String	是	新设置的密码

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
Data	String	返回数据

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/appapi HTTP/1.1
content-type: application/json
{
  "CountryCode": "86",
  "PhoneNumber": "139000000000",
  "VerificationCode": "123456",
  "Password": "password",
  "Signature": "8tEbOa9wk0X3wTBn0BBoFWdgMjo=",
  "Timestamp": 1552621825,
  "Nonce": 2,
  "Action": "AppResetPasswordByCellphone",
  "AppKey": "ahPxd*****rejd",
  "RequestId": "f92406b3-5a9a-****-****-45e3d794bb68"
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": "OK",
    "RequestId": "f92406b3-5a9a-****-****-45e3d794bb68"
  }
}
```

输出示例: 失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.PhoneNumberInvalid",
      "Message": "号码错误"
    },
    "RequestId": "rest-client"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.PhoneNumberInvalid	号码错误
InvalidParameterValue.SendVerifyCodeTooQuick	验证码发送太频繁

使用邮箱重置密码

最近更新时间：2021-05-26 10:48:34

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppResetPasswordByEmail）用于使用邮箱重置密码。

2. 输入参数

名称	类型	必选	描述
Action	String	是	公共参数，本接口取值：AppResetPasswordByEmail
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
AppKey	String	是	公共参数，应用 AppKey，用于标识对应的 App
Signature	String	是	公共参数，请求签名，需用户自行生成，用于校验请求的合法性
Timestamp	Int64	是	公共参数，请求的 Unix 时间戳（秒级）
Nonce	Int	是	公共参数，随机正整数，与 Timestamp 联合起来，防止重放攻击
Email	String	是	邮箱
VerificationCode	String	是	验证码
Password	String	是	新设置的密码

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
Data	String	返回数据

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/appapi HTTP/1.1
content-type: application/json
{
  "Email": "someone@example.com",
  "Password": "password",
  "Signature": "8tEbOa9wk0X3wTBn0BBBoFWdgMjo=",
  "Timestamp": 1552621825,
  "Nonce": 2,
  "Action": "AppResetPasswordByEmail",
  "AppKey": "ahPxd*****rejd",
  "RequestId": "f92406b3-5a9a-****-bc43-45e3d794bb68"
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": "OK",
    "RequestId": "f92406b3-5a9a-****-bc43-45e3d794bb68"
  }
}
```

输出示例: 失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.ErrorUserNotExists",
      "Message": "账号未创建或是已删除"
    },
    "RequestId": "rest-client"
  }
}
```

5. 错误码

错误码	描述

错误码	描述
InternalError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.ErrorUserNotExists	账号未创建或是已删除
InvalidParameterValue.SendVerifyCodeTooQuick	验证码发送太频繁

用户注销

最近更新时间：2021-02-19 10:15:39

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppLogoutUser）用于用户退出登录态。

2. 输入参数

名称	类型	必选	描述
Action	String	是	公共参数，本接口取值：AppLogoutUser
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
Data	String	响应结果

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "AccessToken": "bf*****d098",
  "Action": "AppLogoutUser",
  "RequestId": "rest-client"
}
```

输出示例：成功

```
{
  "Response": {
    "Data": "OK",
    "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
  }
}
```

输出示例：失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.InvalidAccessToken",
      "Message": "Token无效"
    },
    "RequestId": ""
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
ErrorRequiredParamNotFound	必选参数缺失
InvalidAction	Action 非法
InvalidParameterValue	参数非法
InvalidParameterValue.InvalidJSON	请求格式不是 JSON
InvalidParameterValue.InvalidAccessToken	Token 无效

修改用户信息

最近更新时间：2021-02-19 10:19:33

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppUpdateUser）用于修改用户信息。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppUpdateUser
NickName	String	否	昵称
Avatar	String	否	图标
CountryCode	String	否	手机号国家码
PhoneNumber	String	否	手机号码
Email	String	否	邮箱
VerificationCode	String	否	手机或邮箱验证码（绑定手机或邮箱时，该参数必填）

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
Data	String	返回数据

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "NickName": "tests",
  "Avatar": "http://example.com/path/to/avatar.jpg",
  "AccessToken": "bf*****d098",
  "Action": "AppUpdateUser",
  "RequestId": "rest-client"
}
```

输出示例：成功

```
{
  "Response": {
    "Data": "OK",
    "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
  }
}
```

输出示例：失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.InvalidAccessToken",
      "Message": "Token无效"
    },
    "RequestId": ""
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误

错误码	描述
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.NickNameLengthInvalid	昵称长度非法
InvalidParameterValue.PhoneNumberInvalid	号码错误
InvalidParameterValue.PhoneNumberUsed	号码已注册
InvalidParameterValue.EmailInvalid	邮箱错误
InvalidParameterValue.EmailUsed	邮箱已注册
InvalidParameterValue.CheckVerifyCodeFailed	验证码验证不通过

获取用户信息设置

最近更新时间：2021-02-19 10:25:11

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppGetUserSetting）用于获取用户设置的配置信息。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppGetUserSetting

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
UserSetting	UserSetting	用户设置信息

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "AccessToken": "user2",
  "Action": "AppGetUserSetting",
  "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
}
```

输出示例：成功

```
{
  "Response": {
    "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68",
    "UserSetting": {
      "EnableWechatPush": 1,
      "EnableDeviceMessagePush": 0,
      "EnableFamilyMessagePush": 0,
      "EnableNotifyMessagePush": 0
    }
  }
}
```

输出示例：失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.InvalidAccessToken",
      "Message": "Token无效"
    },
    "RequestId": "1555507435215"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数非法
InvalidParameterValue.InvalidJSON	请求格式不是 JSON
InvalidParameterValue.InvalidAccessToken	Token 无效

修改用户信息设置

最近更新时间：2021-02-19 10:25:31

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppUpdateUserSetting）用于修改用户配置信息。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppUpdateUserSetting
EnableWechatPush	Int	是	0：不允许推送微信模板消息 1：允许推送微信模板消息
EnableDeviceMessagePush	Int	是	0：不允许推送告警信息 1：允许推送告警信息
EnableFamilyMessagePush	Int	是	0：不允许推送家庭信息 1：允许推送家庭信息
EnableNotifyMessagePush	Int	是	0：不允许推送通知信息 1：允许推送通知信息

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "AccessToken": "user2",
  "Action": "AppUpdateUserSetting",
  "EnableWechatPush": 1,
  "EnableDeviceMessagePush": 1,
  "EnableFamilyMessagePush": 1,
  "EnableNotifyMessagePush": 1,
  "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
}
```

输出示例：成功

```
{"Response": {"RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68" }}
```

输出示例：失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.InvalidAccessToken",
      "Message": "Token无效"
    },
    "RequestId": "1555507435215"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数非法
InvalidParameterValue.InvalidJSON	请求格式不是 JSON
InvalidParameterValue.InvalidAccessToken	Token 无效

配网管理

生成 Wi-Fi 设备配网 Token

最近更新时间：2021-05-17 10:58:26

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi

本接口（AppCreateDeviceBindToken）用于生成 Wi-Fi 配网任务的随机 Token。

2. 输入参数

名称	类型	必选	描述
Action	String	是	公共参数，本接口取值：AppCreateDeviceBindToken
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需要提供该次请求的 RequestId
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token

3. 输出参数

名称	类型	描述
RequestId	String	RequestId
Token	String	生成的配网 Token

4. 示例

示例1 生成 Wi-Fi 设备配网 Token

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "Action": "AppCreateDeviceBindToken",
  "AccessToken": "adf*****3ks",
```



```
"RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
}
```

输出示例：成功

```
{
  "Response": {
    "Data": {
      "Token": "d7*****1b3b",
      "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
    }
  }
}
```

输出示例：失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.InvalidAccessToken",
      "Message": "Token无效"
    },
    "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
  }
}
```

5. 错误码

错误码	描述
InvalidParameter	无效参数
InvalidParameterValue	参数取值错误
InvalidParameterValue.TokenInfoSaveError	Token 保存失败

查询配网 Token 状态

最近更新时间：2021-05-17 10:57:32

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi

本接口（AppGetDeviceBindTokenState）用于查询配网 Token 的当前状态。

2. 输入参数

名称	类型	必选	描述
Action	String	是	公共参数，本接口取值：AppGetDeviceBindTokenState
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
Token	String	是	由 AppCreateDeviceBindToken 接口生成的配网 Token

3. 输出参数

名称	类型	描述
RequestId	String	RequestId
State	Uint	Token 状态，1：初始生产，2：可使用状态

4. 示例

示例1 查询配网 Token 状态

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "Action": "AppCreateDeviceBindToken",
  "AccessToken": "*****",
  "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68",
```

```
"Token": "d7a1*****a861b3b"
}
```

输出示例：成功

```
{
  "Response": {
    "Data": {
      "State": 1,
      "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
    }
  }
}
```

输出示例：失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.TokenNotExist",
      "Message": "Token不存在"
    },
    "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
  }
}
```

5. 错误码

错误码	描述
InvalidParameter	无效参数
InvalidParameterValue	参数取值错误
InvalidParameterValue.TokenNotExist	Token 不存在
InvalidParameterValue.TokenIsExpire	Token 已过期
InvalidParameterValue.ReadTokenInfoError	Token 读错误

用户绑定 Wi-Fi 设备

最近更新时间：2021-03-12 14:44:16

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi

本接口（AppTokenBindDeviceFamily）用于小程序或 App 用户绑定 Wi-Fi 类设备。

2. 输入参数

名称	类型	必选	描述
Action	String	是	公共参数，本接口取值：AppTokenBindDeviceFamily
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需要提供该次请求的 RequestId
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
Token	String	是	配网 Token
ProductId	String	是	产品 ID
DeviceName	String	是	设备名称
FamilyId	String	是	家庭 ID
RoomId	String	是	房间 ID

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId

4. 示例

示例1 用户绑定 Wi-Fi 设备

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "Action": "AppTokenBindDeviceFamily",
  "AccessToken": "tesetoken",
  "Token": "d7a12d#####861b3b",
  "ProductId": "*****",
  "DeviceName": "*****",
  "FamilyId": "f*****cc1",
  "RoomId": "0"
  "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
}
```

输出示例 成功

```
{
  "Response": {
    "Data": {
      "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
    }
  }
}
```

输出示例 失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.InvalidAccessToken",
      "Message": "Token无效"
    },
    "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
  }
}
```

5. 错误码

错误码	描述

错误码	描述
InvalidParameter	无效参数
InvalidParameterValue	参数取值错误
InvalidParameterValue.TokenNotExist	Token 不存在
InvalidParameterValue.TokenIsExpire	Token 已过期
InvalidParameterValue.ReadTokenInfoError	Token 读错误
InvalidParameterValue.TokenNotBind	Token 未绑定，即在上一步接口 AppGetDeviceBindTokenState 查询返回状态为初始状态，非可使用状态
InvalidParameterValue.FamilyDeviceCountReadError	读设备数量错误
UnauthorizedOperation	无操作权限
UnauthorizedOperation.APPNoPermissionToStudioProduct	App 对操作该产品无权限
UnauthorizedOperation.NoPermissionToFamily	操作该家庭无权限
ResourceNotFound	资源不存在
ResourceNotFound.StudioFamilyNotExist	家庭未创建或是已删除
ResourceNotFound.StudioProductNotExist	产品尚未创建或已被删除
ResourceNotFound.DeviceNotExist	设备未创建或是已删除
LimitExceeded	数量限制
LimitExceeded.FamilyDeviceExceedLimit	设备数量超出限制
InternalError	内部错误
InternalError.InternalServerError	内部错误
InternalError.InternalServerErrorDB	内部 DB 错误
UnsupportedOperation	不支持的操作
UnsupportedOperation.CannotBindSameFamily	该设备已经绑定，请勿重复绑定

设备管理

获取产品信息

最近更新时间：2021-05-14 14:14:07

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppGetProducts）获取产品信息。

2. 输入参数

名称	类型	必选	描述
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppGetProducts
ProductIds	Array of String	是	产品 ID 数组
DeviceIds	Array of String	否	设备 ID 数组
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token

3. 输出参数

名称	类型	描述
Products	Array	返回产品信息
RequestId	String	公共参数，唯一请求 ID，与入参相同

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
```

```
"RequestId": "27HPvi1ylv",
"Action": "AppGetProducts",
"ProductIds":["FVYYYEL4ON"],
"DeviceIds":["QDA1PZLBNB/aaa", "QDA1PZLBNB/bbb"],
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response":{
    "Products":[
      {
        "ProductId":"FVYYYEL4ON",
        "Name":"c1",
        "Description":"yes",
        "State": "",
        "DataTemplate":{"version":"1.0","profile":{"ProductId":"FVYYYEL4ON","CategoryId":"1"},"properties":[{"id":"mac","name":"mac地址","desc":"wifi路由器mac地址","required":true,"mode":"r","define":{"type":"string","min":"0","max":"30"}},{ "id":"signal","name":"wifi强度","desc":"wifi信号强度值","mode":"r","define":{"type":"int","unit":"dbm","step":"1","min":"-180","max":"180","start":"1"}},{ "id":"ssid","name":"wifi热点名","desc":"设备连接的wifi热点名","mode":"r","define":{"type":"string","min":"0","max":"64"}},{ "id":"severip","name":"wifi路由器接入的ip","desc":"wifi路由器接入的ip","mode":"r","define":{"type":"string","min":"0","max":"64"}},{ "id":"nmacs","name":"其他热点mac地址","desc":"可接受到的其他热点mac地址","mode":"r","define":{"type":"string","min":"0","max":"64"}}],"events":[],"actions":[]}},
      "AppTemplate": "",
      "NetType": "wifi",
      "CategoryId": 1,
      "ProductType": 0,
      "UpdateTime": 1600159203
    ],
    {
      "ProductId": "QDA1PZLBNB",
      "Name": "light5勿删",
      "Description": "",
      "State": "",
      "DataTemplate": {"version": "1.0", "profile": {"ProductId": "QDA1PZLBNB", "CategoryId": "141"}, "properties": [{"id": "power_switch", "name": "电灯开关", "desc": "控制电灯开灭", "required": true, "mode": "rw", "define": {"type": "bool", "mapping": {"0": "关", "1": "开"}}}, {"id": "color", "name": "颜色", "desc": "灯光颜色", "mod
```



```
e":"rw","define":{"type":"enum","mapping":{"0":"Red","1":"Green","2":"Blue"}}},{ "id":"brightness","name":"亮度","desc":"灯光亮度","mode":"rw","define":{"type":"int","unit":"%","step":"1","min":"0","max":"100","start":"1"}},{ "id":"name","name":"灯位置名称","desc":"灯位置名称：书房、客厅等","mode":"rw","required":false,"define":{"type":"string","min":"0","max":"64"}},{ "id":"Temperature","name":"温度","desc":"","mode":"rw","define":{"type":"float","min":"0","max":"100","start":"0","step":"1","unit":""},"required":false},{ "id":"temp_max","name":"最高温度","desc":"","mode":"rw","define":{"type":"int","min":"0","max":"100","start":"0","step":"1","unit":""},"required":false},{ "id":"LBS_BS","name":"LBS_BS","desc":"","mode":"rw","define":{"type":"string","min":"0","max":"2048"},"required":false},{ "id":"FOO","name":"FOO","desc":"","mode":"rw","define":{"type":"string","min":"0","max":"2048"},"required":false},{ "id":"GPS_Ext","name":"GPS_Ext","desc":"","mode":"rw","define":{"type":"string","min":"0","max":"2048"},"required":false}], "events":[{"id":"status_report","name":"Device Status","desc":"Report the device status","type":"info","required":false,"params":[{"id":"status","name":"running_state","desc":"Report current device running state","define":{"type":"bool","mapping":{"0":"normal","1":"fault"}}},{ "id":"message","name":"Message","desc":"Some extra message","define":{"type":"string","min":"0","max":"64"}}},{ "id":"low_voltage","name":"LowVoltage","desc":"Alert for device voltage is low","type":"alert","required":false,"params":[{"id":"voltage","name":"Voltage","desc":"Current voltage","define":{"type":"float","unit":"V","step":"1","min":"0.0","max":"24.0","start":"1"}}]},{ "id":"hardware_fault","name":"Hardware_fault","desc":"Report hardware fault","type":"fault","required":false,"params":[{"id":"name","name":"Name","desc":"Name like: memory,tf card, sensors ...","define":{"type":"string","min":"0","max":"64"}},{ "id":"error_code","name":"Error_Code","desc":"Error code for fault","define":{"type":"int","unit":"","step":"1","min":"0","max":"2000","start":"1"}}]},{ "id":"actions":[]}], "AppTemplate":"","NetType":"else", "CategoryId":141, "ProductType":0, "UpdateTime":1600265188 }, { "ProductId":"QDA1PZLBNNB", "Name":"light5勿删", "Description":"","State":"","DataTemplate":{"version":"1.0","profile":{"ProductId":"QDA1PZLBNNB","CategoryId":"141"},"properties":[{"id":"power_switch","name":"电灯开关","desc":"控制电灯开关","required":true,"mode":"rw","define":{"type":"bool","mapping":{"0":"关","1":"开"}}},{ "id":"color","name":"颜色","desc":"灯光颜色","mode":"rw","define":{"type":"enum","mapping":{"0":"Red","1":"Green","2":"Blue"}}},{ "id":"brightness","name":"亮度","desc":"灯光亮度","mode":"rw","define":{"type":"int","unit":"%","step":"1","min":"0","max":"100","start":"1"}},{ "id":"name","name":"灯位置名称","desc":"灯位置名称：书房、客厅等","mode":"rw","required":false,"define":{"type":"string","min":"0","max":"64"}},{ "id":"Temperature"
```

```
e","name":"温度","desc":"","mode":"rw","define":{"type":"float","min":"0","max":"100","start":"0","step":"1","unit":""},"required":false},{id:"temp_max","name":"最高温度","desc":"","mode":"rw","define":{"type":"int","min":"0","max":"100","start":"0","step":"1","unit":""},"required":false},{id:"LBS_BS","name":"LBS_BS","desc":"","mode":"rw","define":{"type":"string","min":"0","max":"2048"},"required":false},{id:"FOO","name":"FOO","desc":"","mode":"rw","define":{"type":"string","min":"0","max":"2048"},"required":false},{id:"GPS_Ext","name":"GPS_Ext","desc":"","mode":"rw","define":{"type":"string","min":"0","max":"2048"},"required":false}],events:[{"id":"status_report","name":"Device Status","desc":"Report the device status","type":"info","required":false,"params":[{"id":"status","name":"running_state","desc":"Report current device running state","define":{"type":"bool","mapping":{"0":"normal","1":"fault"}}},{id:"message","name":"Message","desc":"Some extra message","define":{"type":"string","min":"0","max":"64"}}},{id:"low_voltage","name":"LowVoltage","desc":"Alert for device voltage is low","type":"alert","required":false,"params":[{"id":"voltage","name":"Voltage","desc":"Current voltage","define":{"type":"float","unit":"V","step":"1","min":"0.0","max":"24.0","start":"1"}}]},{id:"hardware_fault","name":"Hardware_fault","desc":"Report hardware fault","type":"fault","required":false,"params":[{"id":"name","name":"Name","desc":"Name like: memory,tf card, sensors ...","define":{"type":"string","min":"0","max":"64"}},{id:"error_code","name":"Error_Code","desc":"Error code for fault","define":{"type":"int","unit":"","step":"1","min":"0","max":"2000","start":"1"}}]}],actions:[]},
"AppTemplate":"","
"NetType":"else",
"CategoryId":141,
"ProductType":0,
"UpdateTime":1600265188
},
"RequestId":"27HPvi1ylv"
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

用户删除设备

最近更新时间：2021-05-17 11:05:01

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppDeleteDeviceInFamily）为用户提供删除设备的功能，删除后用户需要重新配网进行绑定。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppDeleteDeviceInFamily
FamilyId	String	是	家庭 ID，成功创建家庭后，返回的 FamilyId
RoomId	String	是	房间 ID
DeviceName	String	是	设备名称

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "keyi20191119-012345",
  "Action": "AppDeleteDeviceInFamily",
  "AccessToken": "xxxv2",
```

```
"FamilyId": "1",  
"ProductId": "R32ONVL0EU",  
"DeviceName": "df2eSjyY"  
}
```

输出示例 成功

```
{"Response":{"RequestId":"keyi20191119-012345"}}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

获取用户绑定设备列表

最近更新时间：2021-05-17 11:04:38

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppGetFamilyDeviceList）用于获取用户已绑定设备列表。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。
Action	String	是	公共参数，本接口取值：AppGetFamilyDeviceList
FamilyId	String	是	家庭 ID
RoomId	String	是	房间 ID
Offset	Int	否	消息偏移量
Limit	Int	否	最大返回消息条数，最大值为50

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
Total	Int	设备总数
DeviceList	Array	设备列表，请参见 DeviceList

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
```

```
"RequestId": "req_1",
"Action": "AppGetFamilyDeviceList",
"FamilyId": "e0e23a2b33a24652b606ef9107c9a1cf",
"RoomId": "5", //如果要获取家庭所有设备 则该字段为"", "0" 则为默认房间
"Offset": 0,
"Limit": 2,
"AccessToken": "c1*****847e"
}
```

输出实例

```
{
  "Response": {
    "DeviceList": [{
      "ProductId": "R32ONVL0EU",
      "DeviceName": "df2eSjyY",
      "DeviceId": "R32ONVL0EU/df2eSjyY",
      "AliasName": "df2eSjyY",
      "UserId": "1",
      "RoomId": "r_4b27c753ef774d458e0cf8b7686486ad",
      "IconUrl": "",
      "CreateTime": 1574664969,
      "UpdateTime": 1574668779
    }],
    "RequestId": "req_1",
    "Total": 1
  }
}
```

5. 错误码

错误码	描述
InternalError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

获取设备当前状态

最近更新时间：2021-04-26 15:14:59

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppGetDeviceStatuses）用于查询设备状态。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后,获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid
Action	String	是	公共参数，本接口取值：AppGetDeviceStatuses
DeviceIds	Array of String	是	设备 ID 数组

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
Devices	Array of Devices	返回设备

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "3c5dd1032fb0****f5ba7f67cbf2f4",
  "Action": "AppGetDeviceStatuses",
  "DeviceIds": ["HY4DHFM5P6/81386276"],
  "AccessToken": "6bf4003d7f*****400f3186f"
}
```

输出示例：成功

```
{
  "Response": {
    "DeviceStatuses": [{
      "DeviceId": "22F9Y6II7O/light1",
      "ProductId": "22F9Y6II7O",
      "DeviceName": "light1",
      "Online": 0 //0 在线；1： 离线
    }],
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

修改设备名称

最近更新时间：2021-02-19 10:08:15

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppUpdateDeviceInFamily）用于修改设备名称。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid
Action	String	是	公共参数，本接口取值：AppUpdateDeviceInFamily
ProductId	String	是	产品 ID
DeviceName	String	是	设备名称
AliasName	String	是	设备别名

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
DeviceId	String	设备 ID
ProductId	String	产品 ID
DeviceName	String	设备名称
AliasName	String	设备别名
IconUrl	String	图标 URL
FamilyId	String	家庭 ID
RoomId	String	房间 ID
CreateTime	Int64	创建时间

名称	类型	描述
UpdateTime	Int64	更新时间

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "keyi20191119-012345",
  "Action": "AppUpdateDeviceInFamily",
  "AccessToken": "xxxv2",
  "ProductId": "VZS14OZSN5",
  "DeviceName": "~virtualDev",
  "AliasName": "12345"
}
```

输出示例：成功

```
{
  "Response": {
    "Data": {
      "DeviceId": "R32ONVL0EU/df2eSjyY",
      "ProductId": "R32ONVL0EU",
      "DeviceName": "df2eSjyY",
      "AliasName": "12345",
      "IconUrl": "",
      "FamilyId": "",
      "RoomId": 0,
      "CreateTime": 1574859101,
      "UpdateTime": 1574859144
    },
    "RequestId": "keyi20191119-012345"
  }
}
```

5. 错误码

错误码	描述
InternalError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

获取设备详情

最近更新时间：2020-12-03 16:51:23

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppGetDeviceInFamily）用于查询设备详情。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppGetDeviceInFamily
ProductId	String	是	产品 ID
FamilyId	String	是	家庭 ID
DeviceName	String	是	设备名称

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
DeviceId	String	设备 ID，是产品 ID/设备名称组合
ProductId	String	产品 ID
DeviceName	String	设备名称
AliasName	String	设备别名
IconUrl	String	图标 URL
FamilyId	String	家庭 ID
RoomId	String	房间 ID

名称	类型	描述
CreateTime	Int64	创建时间，UNIX 秒级时间戳
UpdateTime	Int64	最后一次更新时间，UNIX 秒级时间戳

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppGetDeviceInFamily",
  "ProductId": "R32ONVL0EU",
  "FamilyId": "xxx",
  "DeviceName": "df2eSjyY",
  "AccessToken": "xxxv2"
}
```

输出示例：成功

```
{
  "Response": {
    "Data": {
      "DeviceId": "R32ONVL0EU/df2eSjyY",
      "ProductId": "R32ONVL0EU",
      "DeviceName": "df2eSjyY",
      "AliasName": "12345",
      "IconUrl": "",
      "FamilyId": "",
      "RoomId": "",
      "DeviceType": 0, // 0 普通 1 网关设备 2 子设备
      "CreateTime": 1574931773,
      "UpdateTime": 1574931945
    },
    "RequestId": "keyi20191119-012345"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
UnauthorizedOperation	无权操作
UnauthorizedOperation.APPNoPermissionToStudioProduct	App 对操作该产品无权限
ResourceNotFound	资源不存在
ResourceNotFound.StudioMemberNotExist	家庭成员未创建或是已删除

设备更换房间

最近更新时间：2021-05-17 10:58:55

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppModifyFamilyDeviceRoom）设备更换房间。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppModifyFamilyDeviceRoom
FamilyId	String	是	家庭 ID
RoomId	String	是	房间 ID
ProductId	String	是	产品 ID
DeviceName	String	是	设备名称

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "keyi20191119-012345",
```

```
"Action": "AppModifyFamilyDeviceRoom",
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
"FamilyId": "1",
"RoomId": "r_4b27c753ef774d458e0cf8b7686486ad",
"ProductId": "R32ONVL0EU",
"DeviceName": "df2eSjyY"
}
```

输出示例: 成功

```
{"Response":{"RequestId":"keyi20191119-012345"}}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

添加子设备到用户绑定列表

最近更新时间：2021-05-18 12:46:35

1. 接口描述

接口请求域名： `iot.cloud.tencent.com/api/exploreropen/tokenapi`。

本接口（`AppBindSubDeviceInFamily`）用于将已经绑定到网关设备的子设备绑定到家庭。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppBindSubDeviceInFamily
GatewayProductId	String	是	网关产品 ID
GatewayDeviceName	String	是	网关设备名称
ProductId	String	是	子产品 ID
DeviceName	String	是	子设备名称

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
AppDeviceInfo	Object of AppDeviceInfo	返回结果

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "keyi20191119-012345",
  "Action": "AppBindSubDeviceInFamily",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
  "ProductId": "LAEG4YJE1A",
  "DeviceName": "subdev2",
  "GatewayProductId": "NJ27OVLZT4",
  "GatewayDeviceName": "gwdev"
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": {
      "AppDeviceInfo": {
        "DeviceId": "R32ONVL0EU/df2eSjyY",
        "ProductId": "R32ONVL0EU",
        "DeviceName": "df2eSjyY",
        "AliasName": "",
        "IconUrl": "",
        "FamilyId": "1",
        "RoomId": "0",
        "CreateTime": 1574152773,
        "UpdateTime": 1574152773
      }
    },
    "RequestId": "keyi20191119-012345"
  }
}
```

5. 错误码

错误码	描述
InternalError	内部错误

错误码	描述
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

获取指定网关设备的子设备列表

最近更新时间：2021-05-17 10:53:32

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppGetGatewayBindDeviceList）用于获取指定网关设备的子设备列表。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid
Action	String	是	公共参数，本接口取值：AppGetGatewayBindDeviceList
ProductId	String	是	子设备的 ProductId
GatewayProductId	String	是	网关设备的 ProductId
GatewayDeviceName	String	是	网关设备的 DeviceName
Offset	Int	是	消息偏移量
Limit	Int	是	最大返回消息条数，最大值为50

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
Total	String	总数量
DeviceList	Array of DeviceList	设备列表

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppGetGatewayBindDeviceList",
  "ProductId": "LAEG4YJE1A", //子设备的productId
  "GatewayProductId": "NJ27OVLZT4", //网关设备的productId
  "GatewayDeviceName": "gwdev", //网关设备的deviceName
  "Offset": 0,
  "Limit": 2,
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例

```
{
  "Response": {
    "DeviceList": [{
      "ProductId": "LAEG4YJE1A",
      "DeviceName": "subdev2", //子设备的deviceName
      "DeviceId": "LAEG4YJE1A/subdev2", //子设备的deviceId
      "AliasName": "",
      "IconUrl": "",
      "BindStatus": 0 //未绑定到家庭
    }, {
      "ProductId": "LAEG4YJE1A",
      "DeviceName": "subdev1",
      "DeviceId": "LAEG4YJE1A/subdev1",
      "AliasName": "",
      "IconUrl": "",
      "BindStatus": 1 //已经绑定到家庭
    }],
    "RequestId": "8b4a70dd16105f*****18edd4e78a3bb8ec",
    "Total": 2
  }
}
```

5. 错误码

错误码	描述
InternalError	内部错误
InternalError.InternalRPCError	调用超时
InternalError.InternalRPCError	调用返回失败
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
UnsupportedOperation	不支持的操作
UnsupportedOperation.NotSupportVirtualDevice	不支持虚拟设备
UnsupportedOperation.DeviceNotGateway	不是网关设备
ResourceNotFound	资源不存在
ResourceNotFound.DeviceNotExist	设备未创建或是已删除
ResourceNotFound.StudioMemberNotExist	家庭成员未创建或是已删除

获取家庭下网关的子设备列表

最近更新时间：2021-05-17 10:53:08

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppGetFamilySubDeviceList）用于获取家庭下网关的子设备列表。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid
Action	String	是	公共参数，本接口取值：AppGetFamilySubDeviceList
GatewayProductId	String	是	网关设备产品 ID
GatewayDeviceName	String	是	网关设备 Name
Offset	Int	是	消息偏移量
Limit	Int	是	最大返回消息条数，最大值为50

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
Total	String	总数量
DeviceList	Array of DeviceList	设备列表

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
```

```
"RequestId": "req_1",
"Action": "AppGetFamilySubDeviceList",
"GatewayProductId": "NJ27OVLZT4",
"GatewayDeviceName": "gwdev",
"Offset": 0,
"Limit": 2,
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例

```
{
  "Response": {
    "DeviceList": [{
      "FamilyId": "f_9b309d84c962****0a11b4c3d9588fcc1",
      "ProductId": "LAEG4YJE1A",
      "DeviceName": "subdev1",
      "DeviceId": "LAEG4YJE1A/subdev1",
      "AliasName": "",
      "RoomId": "0",
      "IconUrl": "",
      "DeviceType": 2,
      "CreateTime": 1583492992,
      "UpdateTime": 1583492992
    }],
    "RequestId": "8b4a70dd16105f*****18edd4e78a3bb8ec",
    "Total": 1
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InternalServerError.InternalRPCError	调用超时
InternalServerError.InternalRPCError	调用返回失败
InvalidParameterValue	参数取值错误

错误码	描述
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.NoPermission	用户对该设备无权限
UnsupportedOperation	不支持的操作
UnsupportedOperation.ERR_APP_INVALID_IOTAPPID	App 无效 lotAppID
ResourceNotFound	资源不存在
ResourceNotFound.DeviceNotExist	设备未创建或是已删除

蓝牙上报数据

最近更新时间：2021-06-07 17:30:03

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppReportDataAsDevice）用于小程序或 App 进行数据上报。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppReportDataAsDevice
DeviceId	String	是	设备 ID
Data	String	是	修改的物模型数据
TimeStamp	Int64	是	Unix 毫秒级时间戳

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
Data	String	返回的数据信息

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "123e4567-e89b-12d3-a456-426614174000",
  "Action": "AppReportDataAsDevice",
```

```
"DeviceId": "22*****O/light1",
"Data": "{\"light_switch\":0}",
"TimeStamp": 1581422682000,
"AccessToken": "8b4a70dd16*****2826868b18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": "",
    "RequestId": "123e4567-e89b-12d3-a456-426614174000"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.BindDeviceNotConnected	设备近期没有连接到云
InvalidParameterValue.InvalidAccessToken	Token 无效

蓝牙设备发送设备行为的回复消息

最近更新时间：2021-06-07 17:29:41

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppPublishMsgAsDevice）提供小程序或 App 模拟设备行为的回复消息功能。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppPublishMsgAsDevice
ProductId	String	是	产品 ID
DeviceName	String	是	设备名称
Topic	String	是	上报的 Topic
Payload	String	是	消息内容

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "123e4567-e89b-12d3-a456-426614174000",
  "Action": "AppPublishMsgAsDevice",
```

```
"ProductId": "22*****O",
"DeviceName": "light1",
"Topic": "$thing/up/action/22*****O/light1"
"Payload": "e1wibGlnaHRfc3dpdGNoXCI6MH0="
"AccessToken": "8b4a70dd16*****2826868b18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "RequestId": "123e4567-e89b-12d3-a456-426614174000"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.BindDeviceNotConnected	设备近期没有连接到云
InvalidParameterValue.InvalidAccessToken	Token 无效

蓝牙设备上报事件

最近更新时间：2021-06-07 17:29:19

1. 接口描述

接口请求域名： `iot.cloud.tencent.com/api/exploreropen/tokenapi` 。

本接口（AppReportDeviceEvent）用于蓝牙设备通过 App 或小程序上报事件。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppReportDeviceEvent
DeviceId	String	是	设备 ID，由 ProductId/DeviceName 拼接组成
EventId	String	是	事件 ID
Params	String	是	JSON 格式的 Event 参数
Method	String	是	上报方法，可以是 ReportEventAsDevice、reported（默认）
EventTimeStamp	Int64	是	Unix 毫秒级时间戳

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
```

```
"RequestId": "123e4567-e89b-12d3-a456-426614174000",
"Action": "AppReportDeviceEvent",
"DeviceId": "QD****BNB/dev01",
"EventId": "low_voltage",
"Params": "{\"voltage\":3.1}",
"Method": "ReportEventAsDevice",
"AccessToken": "ningtoken*****"
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": "",
    "RequestId": "123e4567-e89b-12d3-a456-426614174000"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.BindDeviceNotConnected	设备近期没有连接到云
InvalidParameterValue.InvalidAccessToken	Token 无效

配网绑定设备

最近更新时间：2021-06-07 17:34:52

1. 接口描述

接口请求域名： `iot.cloud.tencent.com/api/exploreropen/tokenapi`。

本接口（ `AppSigBindDeviceInFamily` ）用于小程序或 App 进行 Wi-Fi 设备配网绑定操作。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值： <code>AppSigBindDeviceInFamily</code>
FamilyId	String	是	家庭 ID
ProductId	String	是	产品 ID
DeviceName	String	是	设备名称
RoomId	String	否	房间 ID
DeviceTimestamp	Int64	是	设备时间戳，Unix 秒级时间戳
ConnId	String	否	随机字符串，建议5个字节长度
Signature	String	是	动态签名，由设备根据配网协议传输到小程序、App 端的签名
BindType	String	是	绑定类型。 - Wi-Fi 绑定 <code>wifi_sign</code>（默认） - 蓝牙绑定选择 <code>bluetooth_sigh</code>

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
AppDeviceInfo	Object of AppDeviceInfo	设备信息

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "keyi20191119-012345",
  "Action": "AppSigBindDeviceInFamily",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
  "FamilyId": "1",
  "ProductId": "R32ON***0EU",
  "DeviceName": "df2**JyY",
  "RoomId": "0",
  "DeviceTimestamp": 1574170043,
  "ConnId": "",
  "Signature": "123213213",
  "BindType": "bluetooth_sign"
}
```

输出示例: 成功

```
{
  "Response": {
    "RequestId": "req_1",
    "data": {
      "AppDeviceInfo": {
        "DeviceId": "45454/d2",
        "ProductId": "45454",
        "DeviceName": "d2",
        "AliasName": "",
        "CreateTime": 1552910676,
        "UpdateTime": 1552910676,
        "FamilyId": "1",
        "RoomId": "0",
        "IconUrl": ""
      }
    }
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.BindDeviceNotConnected	设备近期没有连接到云
InvalidParameterValue.InvalidAccessToken	Token 无效

设备控制

用户控制设备

最近更新时间：2021-05-17 11:04:00

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppControlDeviceData）用于用户对绑定的设备发起控制操作。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppControlDeviceData
ProductId	String	是	产品 ID
DeviceName	String	是	设备名
Data	String	是	控制设备报文

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
Data	String	返回数据

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
```

```
"RequestId": "req_1",
"AppId": 1111,
"Action": "AppControlDeviceData",
"ProductId": "22F9Y6II7O",
"DeviceName": "light1",
"Data": {"light_switch": 0},
"AccessToken": "8b*****b8ec"
}
```

输出示例：成功

```
{"Response": {"Data": "", "RequestId": "req_1"}}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

同步调用设备行为

最近更新时间：2021-09-09 14:41:47

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口(AppCallDeviceActionSync)用于用户向已成功绑定的设备发起同步行为调用。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后,获取的访问Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	AppCallDeviceActionSync
ProductId	String	是	产品 ID
DeviceName	String	是	设备名称
ActionId	String	是	行为功能的标识符，ActionId 在产品物模型的行为中定义
InputParams	String	否	输入参数

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
ClientToken	String	调用 ID
Status	String	返回状态
OutputParqams	String	输出参数,注意：此字段可能返回 null，表示取不到有效值

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "AccessToken": "accesstoken*****acc",
  "RequestId": "026fe48b-0c31-4a7c-aaf6-83ba3e9s4bcf",
  "ProductId": "AB*****2345",
  "DeviceName": "DEV",
  "ActionId": "actionid",
  "InputParams": "{\"color\":6}",
  "Action": "AppCallDeviceActionSync"
}
```

输出示例: 成功

```
{
  "Response": {
    "ClientToken": "1ee12*****7df703e0d",
    "OutputParams": "",
    "RequestId": "026fe48b-0c31-4a7c-aaf6-83ba3e9s4bcf",
    "Status": "succ"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

异步调用设备行为

最近更新时间：2021-09-09 14:41:50

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口(AppCallDeviceActionAsync)用于向用户所绑定的设备发起异步行为调用请求，异步方式调用设备行为返回结果可通过长连接通信实时获取。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后,获取的访问Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	AppCallDeviceActionAsync
ProductId	String	是	产品 ID
DeviceName	String	是	设备名称
ActionId	String	是	行为功能的标识符
InputParams	String	否	输入参数

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
ClientToken	String	调用 ID
Status	String	返回状态

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "AccessToken": "accesstoken*****acc",
  "RequestId": "026fe48b-0c31-4a7c-aaf6-83ba3e9s4bcf",
  "ProductId": "ABCDE12345",
  "DeviceName": "DEV",
  "ActionId": "actionid",
  "InputParams": "{\"color\":6}",
  "Action": "AppCallDeviceActionAsync"
}
```

输出示例: 成功

```
{
  "Response": {
    "ClientToken": "1ee12*****7df703e0d",
    "RequestId": "026fe48b-0c31-4a7c-aaf6-83ba3e9s4bcf",
    "Status": "Sent"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

设备分享

App 端发送设备分享

最近更新时间：2021-04-26 15:14:28

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppSendShareDeviceInvite）用于发送设备分享邀请。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权。
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId。
Action	String	是	公共参数，本接口取值：AppSendShareDeviceInvite。
ProductId	String	是	产品 ID。
DeviceName	String	是	设备名称。
ToUserID	String	是	被分享用户 ID。

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
Status	String	请求结果。

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "Action": "AppSendShareDeviceInvite",
```

```
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
"RequestId": "1555507435215",
"ProductId": "22F9Y6II7O",
"DeviceName": "light1",
"ToUserID": "123456"
}
```

输出示例: 成功

```
{
  "Response": {
    "RequestId": "1555507435215",
    "Status": "OK"
  }
}
```

输出示例: 失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.InvalidAccessToken",
      "Message": "Token无效"
    },
    "RequestId": "1555507435215"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.NoPermission	用户对该设备无权限

获取设备分享 Token

最近更新时间：2021-05-17 10:52:41

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppCreateShareDeviceToken）用于获取设备分享 Token。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权。
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId。
Action	String	是	公共参数，本接口取值：AppCreateShareDeviceToken。
FamilyId	String	是	家庭 ID，成功创建家庭后，返回的 FamilyId
ProductId	String	是	产品 ID。
DeviceName	String	是	设备名称。
TokenContext	String	否	分享的上下文。

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
ShareDeviceToken	String	设备分享 Token。

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
```

```
"Action": "AppCreateShareDeviceToken",
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
"RequestId": "155550****15",
"ProductId": "22F9****70",
"DeviceName": "light1"
}
```

输出示例: 成功

```
{
  "Response": {
    "RequestId": "1555507****15",
    "ShareDeviceToken": "49d2b71dc5814****7c68a4c756d3ce9"
  }
}
```

输出示例: 失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.InvalidAccessToken",
      "Message": "Token无效"
    },
    "RequestId": "1555507****15"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.InvalidAccessToken	Token 无效。
InvalidParameterValue.NoPermission	用户对该设备无权限。

获取设备分享 Token 信息

最近更新时间：2021-05-17 10:52:17

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppDescribeShareDeviceToken）用于获取设备分享 Token 信息。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权。
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId。
Action	String	是	公共参数，本接口取值：AppDescribeShareDeviceToken。
ShareDeviceToken	String	是	设备分享 Token。

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
ShareDeviceTokenInfo	Object of ShareDeviceTokenInfo	响应的结果。
AliasName	String	设备别名
IconUrl	String	图标 URL

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "Action": "AppDescribeShareDeviceToken",
```

```
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",  
"RequestId": "1555507****15",  
"ShareDeviceToken": "49d2b71dc5814354****8a4c756d3ce9"  
}
```

输出示例: 成功

```
// UserId、UserNick 在被分享者接受前为空  
// 被分享者接受分享后，UserId 保存的是被分享者的信息  
// 前端可以通过UserId是否为空，来判断绑定情况。  
{  
  "Response": {  
    "RequestId": "1555507****15",  
    "ShareDeviceTokenInfo": {  
      "FromUserId": "1",  
      "FromUserNick": "tests",  
      "UserId": "2",  
      "UserNick": "test2",  
      "ProductId": "22F9Y6II7O",  
      "DeviceName": "light1",  
      "AliasName": "my_light",  
      "IconUrl": "",  
      "BindTime": 0,  
      "ExpireTime": 1574757262,  
      "CreateTime": 1574152462  
    }  
  }  
}
```

输出示例: 失败

```
{  
  "Response": {  
    "Error": {  
      "Code": "InvalidParameterValue.InvalidShareDeviceToken",  
      "Message": "设备分享Token无效"  
    },  
    "RequestId": "1555507****15"  
  }  
}
```

```
}  
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.InvalidShareDeviceToken	设备分享 Token 无效。

绑定用户分享的设备

最近更新时间：2021-03-12 14:37:21

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppBindUserShareDevice）用于绑定用户分享的设备。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权。
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId。
Action	String	是	公共参数，本接口取值：AppBindUserShareDevice。
ShareDeviceToken	String	是	设备分享的 Token。
ProductId	String	是	产品 ID。
DeviceName	String	是	设备名称。

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
Status	String	响应结果。

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "Action": "AppBindUserShareDevice",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
  "RequestId": "1555507****15",
```



```
"ProductId": "22F9Y6II7O",
"DeviceName": "light1",
"ShareDeviceToken": "0820728466424d8f****422ec06bd398"
}
```

输出示例: 成功

```
{
  "Response": {
    "RequestId": "1555507****15",
    "Status": "OK"
  }
}
```

输出示例: 失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.DeviceBinded",
      "Message": "设备已经和当前用户绑定"
    },
    "RequestId": "1555507****15"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.DeviecBinded	设备已经和当前用户绑定。
InvalidParameterValue.UserBindExceedLimit	用户绑定分享设备数已达限制。
InvalidParameterValue.DeviceShareExceedLimit	设备已达最大分享用户数限制。
InvalidParameterValue.CannotBindOwnDevice	不能绑定自己分享的设备。

错误码	描述
InvalidParameterValue.ShareDeviceTokenUsed	设备分享 Token 已使用。
InvalidParameterValue.InvalidShareDeviceToken	设备分享 Token 无效。

查询用户分享设备列表

最近更新时间：2021-05-17 10:51:47

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppListUserShareDevices）用于查询用户分享的设备列表。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权。
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId。
Action	String	是	公共参数，本接口取值：AppListUserShareDevices。
Offset	Int	是	分页偏移，0起始，最大不超过500。
Limit	Int	是	单次拉取数量，默认为10，最大不超过100。

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
Total	Int	返回的消息总条数。
ShareDevices	Array of ShareDevices	响应设备信息。

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "Action": "AppListUserShareDevices",
```

```
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
"RequestId": "1555507****15",
"Offset": 0,
"Limit": 10
}
```

输出示例: 成功

```
{
  "Response": {
    "RequestId": "1555507****15",
    "ShareDevices": [
      {
        "ProductId": "WTXXX0MF2",
        "DeviceName": "device_XXXX001",
        "DeviceId": "WTXXX0MF2/device_XXXX001",
        "AliasName": "test_device",
        "IconUrl": "https://yyy-xxxx-xxx.cos.ap-guangzhou.myqcloud.com/113xxxx7a0b273bxxxxc0b9040feb/1xxxxxxxxxx0.png",
        "CreateTime": "2019-12-17T19:39:21+08:00"
      }
    ],
    "Total": 1
  }
}
```

输出示例: 失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.InvalidAccessToken",
      "Message": "Token无效"
    },
    "RequestId": "1555507****15"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.InvalidAccessToken	Token 无效。

删除用户分享的设备

最近更新时间：2021-05-14 14:23:38

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppRemoveUserShareDevice）用于删除用户分享的设备。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权。
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId。
Action	String	是	公共参数，本接口取值：AppRemoveUserShareDevice。
ProductId	String	是	产品 ID。
DeviceName	String	是	设备名称。

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
Status	String	响应结果。

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "Action": "AppRemoveUserShareDevice",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
  "RequestId": "1555507****15",
  "ProductId": "22F9Y6II7O",
```

```
"DeviceName": "light1"
}
```

输出示例: 成功

```
{
  "Response": {
    "RequestId": "1555507****15",
    "Status": "OK"
  }
}
```

输出示例: 失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.AppDeviceNotExist",
      "Message": "没有找到绑定设备"
    },
    "RequestId": "1555507****15"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.AppDeviceNotExist	没有找到绑定设备。

查询设备的用户列表

最近更新时间：2021-05-14 14:21:16

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppListShareDeviceUsers）用于查询设备用户列表。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权。
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId。
Action	String	是	公共参数，本接口取值：AppListShareDeviceUsers。
DeviceName	String	是	设备名称。
Offset	Int	是	分页偏移，0起始，最大不超过500。
Limit	Int	是	单次拉取数量，默认为10，最大不超过100。

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
Total	Int	返回的数据总数。
Users	Array of ShareUserInfo	用户列表。

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
```



```
{
  "Action": "AppListShareDeviceUsers",
  "AccessToken": "user2",
  "RequestId": "1555507****15",
  "ProductId": "22F9Y6II7O",
  "DeviceName": "light1",
  "Offset": 0,
  "Limit": 10
}
```

输出示例: 成功

```
{
  "Response": {
    "RequestId": "1555507****15",
    "Users": [
      {
        "UserId": "1",
        "CountryCode": "86",
        "PhoneNumber": "139****5678",
        "NickName": "tests",
        "Avatar": "",
        "BindTime": 1574153536
      }
    ],
    "Total": 1
  }
}
```

输出示例: 失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.InvalidAccessToken",
      "Message": "Token无效"
    },
    "RequestId": "1555507****15"
  }
}
```

```
}  
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.InvalidAccessToken	Token 无效。

删除设备分享的用户

最近更新时间：2021-03-12 14:34:09

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。
本接口（AppRemoveShareDeviceUser）用于删除设备分享的用户。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权。
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId。
Action	String	是	公共参数，本接口取值：AppRemoveShareDeviceUser。
RemoveUserId	String	是	待删除用户 ID。
ProductId	String	是	产品 ID。
DeviceName	String	是	设备名称。

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
Status	String	响应结果。

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "Action": "AppRemoveShareDeviceUser",
  "AccessToken": "user2",
  "RequestId": "15555*****15",
```

```
"RemoveUserId": "1234555",
"ProductId": "22F9Y6II7O",
"DeviceName": "light1"
}
```

输出示例: 成功

```
{
  "Response": {
    "RequestId": "155550*****15",
    "Status": "OK"
  }
}
```

输出示例: 失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.AppDeviceNotExist",
      "Message": "没有找到绑定设备"
    },
    "RequestId": "15555*****15"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.AppDeviceNotExist	没有找到绑定设备。
InvalidParameterValue.NoPermission	用户对该设备无权限。

家庭管理

创建家庭

最近更新时间：2021-05-17 11:03:23

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppCreateFamily）用于创建家庭。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppCreateFamily
Name	String	是	家庭名称
Address	String	是	家庭地址

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
FamilyId	String	成功创建的家庭 ID

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppCreateFamily",
```

```
"Name": "family_name",
"Address": "family_address",
"AccessToken": "c1*****847e"
}
```

输出示例：成功

```
{
  "Response": {
    "Data": {
      "FamilyId": "e0e23a2b33a24652b606ef9107c9a1cf"
    },
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

删除家庭

最近更新时间：2021-05-17 11:03:08

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppDeleteFamily）用于删除家庭。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppDeleteFamily
Name	String	是	需要删除的家庭名称
FamilyId	String	是	需要删除的家庭 ID

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppDeleteFamily",
  "FamilyId": "e0e23a2b33a24652b606ef9107c9a1cf",
  "Name": "family_name",
```

```
"AccessToken": "c1*****847e"
}
```

输出示例：成功

```
{
  "Response": {
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

修改家庭

最近更新时间：2021-05-17 11:02:50

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppModifyFamily）用于修改家庭。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid
Action	String	是	公共参数，本接口取值：AppModifyFamily
Name	String	是	修改后的家庭名称
FamilyId	String	是	需要修改的家庭 ID
Address	String	是	修改后的家庭地址

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppModifyFamily",
  "Name": "family_name",
  "Address": "family_address",
```

```
"FamilyId":"e0e23a2b33a24652b606ef9107c9a1cf",
"AccessToken": "c1*****847e"
}
```

输出示例：成功

```
{
  "Response": {
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

获取家庭列表

最近更新时间：2021-05-17 11:02:33

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppGetFamilyList）用于获取家庭列表。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppGetFamilyList
Offset	Int	否	所需要查询数据的偏移量
Limit	Int	否	返回条数的限制量，最多为50条

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
Total	Int	返回的家庭数量总数
FamilyList	Array	返回的家庭列表信息数组，请参见 FamilyList

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppGetFamilyList",
```

```
"Offset":0,
"Limit":2,
"AccessToken": "c1*****847e"
}
```

输出示例：成功

```
{
  "Response": {
    "RequestId": "req_1",
    "FamilyList": [{
      "FamilyId": "a1c6939b39d345b897b168313f8ca12c",
      "Name": "family_name",
      "Role": 0, // 1:管理员 0: 普通成员
      "CreateTime": 1570786578,
      "UpdateTime": 1570790807
    }],
    "Total":1
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

获取家庭详情

最近更新时间：2021-05-17 11:02:17

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppDescribeFamily）用于获取家庭详情。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppDescribeFamily
FamilyId	String	是	所需查询详情的家庭 ID

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
FamilyId	String	家庭 ID
FamilyName	String	家庭名称
Address	String	家庭地址
CreateTime	Int64	创建时间，UNIX 时间戳（秒级）
UpdateTime	Int64	最后一次更新时间，UNIX 时间戳（秒级）

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
```

```
{
  "RequestId": "req_1",
  "Action": "AppDescribeFamily",
  "FamilyId": "a1c6939b39d345b897b168313f8ca12c",
  "AccessToken": "c1*****847e"
}
```

输出示例：成功

```
{
  "Response": {
    "RequestId": "req_1",
    "Data": {
      "FamilyId": "a1c6939b39d345b897b168313f8ca12c",
      "FamilyName": "family_name",
      "Address": "family_address",
      "CreateTime": 1570786578,
      "UpdateTime": 1570790807
    }
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

新建房间

最近更新时间：2021-05-17 11:01:59

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppCreateRoom）用于新建房间。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppCreateRoom
FamilyId	String	是	新建房间的所属家庭 ID
Name	String	是	新建的房间名称

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
RoomId	String	成功创建的房间 ID

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppCreateRoom",
  "Name": "name",
```

```
"FamilyId": "family_address",
"AccessToken": "c1*****847e"
}
```

输出示例：成功

```
{
  "Response": {
    "Data": {
      "RoomId": 123456
    },
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

修改房间

最近更新时间：2021-05-17 11:01:44

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppModifyRoom）用于修改房间。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppModifyRoom
FamilyId	String	是	需要修改的房间所属家庭 ID
Name	String	否	房间名称
RoomId	String	是	需要修改的房间 ID

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppModifyRoom",
  "FamilyId": "fa070f59a2fe4****43815310c794fdf",
```

```
"RoomId": "5",
"Name": "TestName",
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例：成功

```
{
  "Response": {
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

删除房间

最近更新时间：2021-05-17 11:01:29

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppDeleteRoom）用于删除房间。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppDeleteRoom
FamilyId	String	是	需要删除的房间所属家庭 ID
RoomId	String	是	需要删除的房间 ID

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppDeleteRoom",
  "FamilyId": "e0e23a2b33a24652b606ef9107c9a1cf",
  "RoomId": "123456",
```

```
"AccessToken": "c1*****847e"
}
```

输出示例：成功

```
{
  "Response": {
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

获取房间列表

最近更新时间：2021-05-17 11:01:13

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppGetRoomList）用于获取房间列表。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权。
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppGetRoomList
Offset	Int	否	所需要查询的数据的偏移量
Limit	Int	否	所需要查询的总限制量，最大返回50条
FamilyId	String	否	需要获取的家庭 ID

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
Total	Int	返回的设备总数
FamilyList	Array	返回的家庭列表信息数组，请参见 Roomlist

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
```

```
"Action": "AppGetRoomList",
"FamilyId": "fa070f59a2fe4f1da43815310c794fdf",
"Offset": 0,
"Limit": 2,
"AccessToken": "xxx"
}
```

输出示例：成功

```
{
  "Response": {
    "RequestId": "req_1",
    "FamilyList": [{
      "RoomId": "123456",
      "RoomName": "name",
      "DeviceNum": 2,
      "CreateTime": 1570786578,
      "UpdateTime": 1570790807
    }],
    "Total": 1
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

邀请家庭成员

最近更新时间：2021-05-17 11:00:56

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppInviteMember）用于向微信好友发送邀请加入家庭请求。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppInviteMember
Name	String	是	家庭名称
FamilyId	String	是	家庭 ID

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
ShareToken	String	分享 Token，用于邀请成员加入家庭

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppInviteMember",
  "Name": "family_name",
```

```
"FamilyId": "e0e23a2b33a24652b606ef9107c9a1cf",  
"AccessToken": "c1*****847e"  
}
```

输出示例：成功

```
{  
  "Response": {  
    "ShareToken": "e0*****a1cf",  
    "RequestId": "req_1"  
  }  
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

成员加入家庭

最近更新时间：2021-05-17 11:00:39

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppJoinFamily）用于成员加入家庭。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppJoinFamily
ShareToken	String	是	分享出去的 Token，用于标识一个分享

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "550e8400-e29b-41d4-a716-446655440000",
  "Action": "AppJoinFamily",
  "ShareToken": "e0*****a1cf",
  "AccessToken": "c1*****847e"
}
```

输出示例：成功

```
{
  "Response": {
    "RequestId": "550e8400-e29b-41d4-a716-446655440000"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

删除家庭成员

最近更新时间：2021-05-17 11:00:23

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppDeleteFamilyMember）用于管理员删除家庭成员。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppDeleteFamilyMember
MemberID	String	是	需要移除的成员 ID

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "550e8400-e29b-41d4-a716-446655440000",
  "Action": "AppDeleteFamilyMember",
  "MemberID": "12345678",
  "AccessToken": "c1*****847e"
}
```

输出示例：成功

```
{
  "Response": {
    "RequestId": "550e8400-e29b-41d4-a716-446655440000"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

成员退出家庭

最近更新时间：2021-05-17 11:00:05

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppExitFamily）用于成员主动退出某个家庭。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问Token
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppExitFamily

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "550e8400-e29b-41d4-a716-446655440000",
  "Action": "AppExitFamily",
  "FamilyId": "e0e23a2b33a24652b606ef9107c9a1cf",
  "AccessToken": "c1*****847e"
}
```

输出示例：成功

```
{
  "Response": {
    "RequestId": "550e8400-e29b-41d4-a716-446655440000"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

获取家庭成员列表

最近更新时间：2021-05-17 10:59:36

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppGetFamilyMemberList）用于获取家庭成员列表。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppGetFamilyMemberList
Offset	Int	否	所需要查询的数据的偏移量
Limit	Int	否	所需要查询的总限制量，最大返回50条
FamilyId	String	是	需要获取的家庭 ID

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
Total	Int	返回的家庭成员总数
MemberList	Array	返回的家庭成员数组，请参见 MemberList

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
```

```
"FamilyId": "f_b3ac7e6b1cd24392b41e65d91766215a",
"Offset": 0,
"Limit": 100,
"Action": "AppGetFamilyMemberList",
"RequestId": "hojkE0Hx88",
"AccessToken": "3c*****40c5"
}
```

输出示例：成功

```
{
  "MemberList": [{
    "UserID": "7267",
    "NickName": "deny",
    "Avatar": "https://wx.qlogo.cn/mmopen/vi_32/V2kIBBE35phMrFKu5sra16bzpUuPtokyWTbMRPt7bRGBjx4fEqQrZxoDYyxNVbtkXMosuxQelqiafIZJ8ADr8cw/132",
    "Role": 1
  }],
  "Total": 1,
  "RequestId": "hojkE0Hx88"
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

App 端邀请家庭成员

最近更新时间：2021-05-14 14:24:15

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppSendShareFamilyInvite）用于发送家庭分享邀请。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权。
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId。
Action	String	是	公共参数，本接口取值：AppSendShareFamilyInvite
FamilyId	String	是	家庭 ID
ToUserID	String	是	被分享用户 ID

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId。

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "Action": "AppSendShareFamilyInvite",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
  "RequestId": "1555507435215",
  "FamilyId": "e0e23a2b33a24652b606ef9107c9a1cf",
```

```
"ToUserId":"123456"
}
```

输出示例: 成功

```
{
  "Response": {
    "RequestId": "1555507435215",
  }
}
```

输出示例: 失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.InvalidAccessToken",
      "Message": "Token无效"
    },
    "RequestId": "1555507435215"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.NoPermission	用户对该家庭无权限

设备定时 新建定时任务

最近更新时间：2021-05-17 11:06:52

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppCreateTimer）用于用户创建设备的定时任务。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值 AppCreateTimer
ProductId	String	是	新建定时任务所属产品 ID
DeviceName	String	是	新建定时任务控制的设备名称
TimerName	String	是	定时器的名称
Days	String	是	定时器开启时间，每一位，0：关闭，1：开启，从左至右依次表示：周日、周一、周二、周三、周四、周五、周六
TimePoint	String	是	定时器开启时间点
Repeat	int	是	是否循环，0：不需要，1：需要
Data	String	是	定时器启动时下发的控制报文

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
TimerId	String	定时器 ID
TimerName	String	定时器名称

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppCreateTimer",
  "ProductId": "US4CJ11DIK",
  "DeviceName": "411_3",
  "AccessToken": "c1*****847e",
  "TimerName": "timer_test",
  "Days": "1000000", // 每一位 0:关闭,1:开启, 从左至右依次表示: 周日 周一 周二 周三 周四 周五 周六
  "TimePoint": "06:30",
  "Repeat": 1, //是否需要循环 0:不需要 1 需要
  "Data": "{\"brightness\": 25}"
}
```

输出示例：成功

```
{
  "Response": {
    "Data": {
      "TimerId": "e0e23a2b33a24652b606ef9107c9a1cf",
      "TimerName": "timer_test"
    },
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

修改定时任务

最近更新时间：2021-05-17 11:06:36

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppModifyTimer）用于修改设备的定时任务。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值 AppModifyTimer
ProductId	String	是	修改定时器对应的产品 ID
DeviceName	String	是	定时器对应的设备名称
TimerName	String	是	定时器的名称
TimerId	String	是	定时器 ID
TimePoint	String	是	定时器开启时间点
Days	String	是	定时器开启时间，每一位，0：关闭，1：开启，从左至右依次表示：周日、周一、周二、周三、周四、周五、周六
Repeat	Int	是	是否循环，0：不需要，1：需要
Data	String	是	定时器启动时，下发的控制报文

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppModifyTimer",
  "ProductId": "US4CJ11DIK",
  "DeviceName": "411_3",
  "AccessToken": "c1*****847e",
  "TimerId": "a1c6939b39d345b897b168313f8ca12c",
  "TimerName": "timer_test_modify",
  "Days": "1100000",
  "TimePoint": "9:30",
  "Repeat": 0,
  "Data": "{\"brightness\": 28}"
}
```

输出示例：成功

```
{"Response": {"RequestId": "req_1"}}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

删除定时任务

最近更新时间：2021-05-17 11:06:12

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppDeleteTimer）用于删除设备定时任务。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需要提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值 AppDeleteTimer
ProductId	String	是	修改定时器对应的产品 ID
DeviceName	String	是	定时器对应的设备名称
TimerId	String	是	定时器 ID

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppDeleteTimer",
  "ProductId": "US4CJ11DIK",
```

```
"DeviceName": "411_3",  
"TimerId": "e0e23a2b33a24652b606ef9107c9a1cf",  
"AccessToken": "c1*****847e"  
}
```

输出示例：成功

```
{"Response": {"RequestId": "req_1"}}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

修改定时任务状态

最近更新时间：2021-05-17 11:05:49

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppModifyTimerStatus）用于修改定时任务的启停状态。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值 AppModifyTimerStatus
ProductId	String	是	需要修改定时任务列表的产品 ID
DeviceName	String	是	需修改定时器状态设备名称
TimerId	String	是	定时器 ID
Status	Int	否	需要修改的定时器状态，0：关闭，1：开启

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppModifyTimerStatus",
```

```
"ProductId": "US4CJ11DIK",
"DeviceName": "411_3",
"TimerId": "a1c6939b39d345b897b168313f8ca12c",
"Status": 0,
"AccessToken": "c1*****847e"
}
```

输出示例：成功

```
{"Response":{"RequestId":"req_1"}}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

获取定时任务列表

最近更新时间：2021-05-17 11:05:26

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppGetTimerList）用于获取定时器列表。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值 AppGetTimerList
ProductId	String	是	需要获取定时任务列表的产品 ID
DeviceName	String	是	需要获取定时任务列表的设备名称
Offset	Int	否	获取的列表偏移量
Limit	Int	否	获取到的总消息条数，最大为50

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
TimerList	Array	获取到的产品列表，请参见 TimerList
Total	Int	总共拥有的条数

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
```

```
{
  "RequestId": "req_1",
  "Action": "AppGetTimerList",
  "ProductId": "US4CJ11DIK",
  "DeviceName": "411_3",
  "Offset": 0,
  "Limit": 2,
  "AccessToken": "c1*****847e"
}
```

输出示例：成功

```
{
  "Response": {
    "RequestId": "req_1",
    "Total": 2,
    "TimerList": [{
      "TimerId": "a1c6939b39d345b897b168313f8ca12c",
      "TimerName": "timer_test_modify",
      "ProductId": "US4CJ11DIK",
      "DeviceName": "411_3",
      "Days": "1100000",
      "TimePoint": "09:30",
      "Repeat": 0,
      "Data": "{\"brightness\": 28}",
      "Status": 0,
      "CreateTime": 1570786578,
      "UpdateTime": 1570790807
    }, {
      "TimerId": "f888dd20c70744939a5e9bdd631b1c3a",
      "TimerName": "timer_test",
      "ProductId": "US4CJ11DIK",
      "DeviceName": "411_3",
      "Days": "1000000",
      "TimePoint": "07:00",
      "Repeat": 1,
      "Data": "{\"brightness\": 25}",
      "Status": 1,
      "CreateTime": 1570786483,
      "UpdateTime": 1570786483
    }
  ]
}
```

```
}|
}
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

固件升级

查询设备固件是否升级

最近更新时间：2021-02-19 10:05:39

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppCheckFirmwareUpdate）用于查询设备固件升级信息。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppCheckFirmwareUpdate
ProductId	String	是	产品 ID
DeviceName	String	是	设备名

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
CurrentVersion	String	设备当前固件版本
DstVersion	String	固件可升级版本

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
```

```
{
  "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68",
  "Action": "AppCheckFirmwareUpdate",
  "ProductId": "22*****O",
  "DeviceName": "li****1",
  "AccessToken": "8b4d16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "DstVersion": "2.0",
    "CurrentVersion": "1.0",
    "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

确认固件升级任务

最近更新时间：2021-05-17 10:54:28

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppPublishFirmwareUpdateMessage）用于用户确认升级后，云端向设备发起固件升级请求。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppPublishFirmwareUpdateMessage
ProductId	String	是	产品 ID
DeviceName	String	是	设备名

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68",
  "Action": "AppPublishFirmwareUpdateMessage",
  "ProductId": "22*****O",
  "DeviceName": "li****1",
```



```
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

查询设备固件升级状态

最近更新时间：2021-04-26 15:13:00

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppDescribeFirmwareUpdateStatus）用于查询设备固件升级状态及进度。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppDescribeFirmwareUpdateStatus
ProductId	String	是	产品 ID
DeviceName	String	是	设备名

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
OriVersion	String	升级任务源版本
DstVersion	String	升级任务目标版本
ErrMsg	String	错误信息
Percent	Int64	升级进展百分比

名称	类型	描述
Status	Int64	升级状态： • 0 设备离线 • 1 待处理 • 2 消息下发成功 • 3 下载中 • 4 烧录中 • 5 失败 • 6 升级完成 • 7 正在处理中 • 8 等待用户确认 • 20 下载完成

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68",
  "Action": "AppDescribeFirmwareUpdateStatus",
  "ProductId": "22*****O",
  "DeviceName": "li****1",
  "AccessToken": "8b4d16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "DstVersion": "2.0",
    "ErrMsg": "",
    "OriVersion": "1.0",
    "Percent": 0,
    "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

消息管理

获取消息列表

最近更新时间：2021-05-17 11:07:57

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppGetMessages）用于查询消息列表，消息包括设备告警消息、操作通知消息。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问Token
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用uuid。定位问题时需要提供该次请求的RequestId
Action	String	是	公共参数，本接口取值：AppGetMessages
Category	Int64	是	主类型，1：设备，2：家庭，3：通知
MsgID	String	否	消息ID，首次可不传
MsgTimestamp	Int64	否	消息的时间戳，首次可不传或传0
Limit	Int64	是	最大返回条数，最大不超过100

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求ID，与入参相同
Listover	String	用于翻页

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
```

```
{
  "Action": "AppGetMessages",
  "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68",
  "AccessToken": "8b*****b8ec",
  "Category": 2,
  "MsgID": "",
  "MsgTimestamp": 0,
  "Limit": 5
}
```

输出示例：成功

```
{
  "Response": {
    "Data": {
      "Listover": true,
      "Msgs": [{
        "UserID": "1",
        "FromUserID": "",
        "MsgID": "65167410520395776",
        "Category": 1,
        "MsgType": 100,
        "MsgTitle": "Device Warning",
        "MsgContent": "Some message Wed Nov 27 2019 11:52:01 GMT+0000.",
        "MsgTimestamp": 1574855521420,
        "ProductId": "",
        "DeviceName": "",
        "DeviceAlias": "",
        "FamilyId": "",
        "FamilyName": "",
        "RelateUserID": "",
        "CreateAt": "2019-11-27T11:52:01.42Z"
      }, {
        "UserID": "1",
        "FromUserID": "1",
        "MsgID": "65167395785805824",
        "Category": 2,
        "MsgType": 200,
        "MsgTitle": "成员添加结果通知",
        "MsgContent": ""jordan"将"dylan"添加为"深圳的家"的成员，成员可以查看和控制家里的智能设备。 Wed Nov
```

```
27 2019 11:51:57 GMT+0000",
"MsgTimestamp": 1574855517907,
"ProductId": "",
"DeviceName": "",
"DeviceAlias": "",
"FamilyId": "1",
"FamilyName": "test",
"RelateUserID": "2",
"CreateAt": "2019-11-27T11:51:57.907Z"
}, {
"UserId": "1",
"FromUserId": "1",
"MsgId": "65167386239569920",
"Category": 2,
"MsgType": 201,
"MsgTitle": "成员删除结果通知",
"MsgContent": "\"jordan\"将\"dylan\"从\"深圳的家\"中删除，删除后将不能查看和控制家中的智能设备。 Wed Nov
27 2019 11:51:54 GMT+0000",
"MsgTimestamp": 1574855515631,
"ProductId": "",
"DeviceName": "",
"DeviceAlias": "",
"FamilyId": "1",
"FamilyName": "test",
"RelateUserID": "2",
"CreateAt": "2019-11-27T11:51:55.631Z"
}, {
"UserId": "1",
"FromUserId": "1",
"MsgId": "65167339519217664",
"Category": 3,
"MsgType": 300,
"MsgTitle": "设备分享结果通知",
"MsgContent": "\"dylan\"已成功接收\"jordan\"分享的设备\"智能灯\"，现在可以查看和控制设备了。 Wed Nov 27 2
019 11:51:44 GMT+0000",
"MsgTimestamp": 1574855504492,
"ProductId": "22F9Y6II7O",
"DeviceName": "light1",
"DeviceAlias": "light1",
"FamilyId": "",
```

```
"FamilyName": "",
"RelateUserID": "2",
"CreateAt": "2019-11-27T11:51:44.492Z"
}]
},
"RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
}
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

删除消息

最近更新时间：2021-05-17 11:07:31

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppDeleteMessage）用于删除消息。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppDeleteMessage
MsgID	String	是	消息 ID

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
Data	String	返回数据

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "Action": "AppDeleteMessage",
  "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68",
  "AccessToken": "8b*****b8ec",
  "MsgID": 1633328834809856
}
```

输出示例：成功

```
{
  "Response": {
    "Data": "OK",
    "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
  }
}
```

输出示例：失败

```
{
  "Response": {
    "Error": {
      "Code": "InvalidParameterValue.InvalidAccessToken",
      "Message": "Token无效"
    },
    "RequestId": ""
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.MsgIDInvalid	消息 ID 非法

发布 RRPC 消息

最近更新时间：2021-09-15 17:04:00

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口(AppPublishRRPCMessage)用于向设备发布 RRPC 消息进行远程配置。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后,获取的访问 Token
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppPublishRRPCMessage
ProductId	String	是	产品 ID
DeviceName	String	是	设备名
Payload	String	是	消息内容，utf8 编码

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
MessageId	Int64	RRPC 消息 ID
PayloadBase64	String	设备回复的消息内容，采用 base64 编码

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
```

```
{
  "Action": "AppPublishRRPCMessage",
  "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
  "ProductId": "ASBHKN",
  "DeviceName": "dev",
  "Payload": "1234561"
}
```

输出示例: 成功

```
{
  "Response": {
    "MessageId": 74,
    "PayloadBase64": "QUJDRA==",
    "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

手动智能

创建手动智能联动

最近更新时间：2021-06-07 17:28:23

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppCreateScene）用于创建手动智能联动。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppCreateScene
FamilyId	String	是	手动智能联动所属家庭 ID
SceneName	String	是	手动智能联动的名称
SceneIcon	String	是	手动智能联动的背景图片地址
Actions	Array of Action	是	手动智能联动动作列表

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
SceneId	String	手动智能联动 ID

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "Actions": [
    {
      "ActionType": 0,
      "ProductId": "R32O***0EU",
      "DeviceName": "df****yY",
      "Data": "{\"brightness\": 25}"
    }
  ],
  "AccessToken": "x***v2",
  "RequestId": "re***_1",
  "Action": "AppCreateScene",
  "FamilyId": "f_9b30*****c3d9588fcc1",
  "SceneName": "name",
  "SceneIcon": "icon"
}
```

输出示例: 成功

```
{
  "Response": {
    "SceneId": "a_cd61ddbe*****47d21d6aeffa",
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InternalServerError.InternalServerErrorDB	DB 错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

错误码	描述
InvalidParameterValue.NoPermission	用户对该设备无权限

获取手动智能联动列表

最近更新时间：2021-06-07 17:28:06

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppGetSceneList）用于获取手动智能联动列表。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppGetSceneList
FamilyId	String	是	家庭 ID
Offset	Int	是	消息偏移量
Limit	Int	是	最大返回消息条数，最大值为 50

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
SceneList	Array of Scene	手动智能联动列表
Total	Int	手动智能联动总数

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
```



```
{
  "AccessToken": "***v2",
  "RequestId": "req_1",
  "Action": "AppGetSceneList",
  "FamilyId": "f_9b309*****4c3d9588fcc1",
  "Offset": 0,
  "Limit": 10
}
```

输出示例: 成功

```
{
  "Response": {
    "RequestId": "req_1",
    "SceneList": [
      {
        "SceneId": "s_f7feb440d*****28f3a598d00c",
        "FamilyId": "f_9b309d8*****a11b4c3d9588fcc1",
        "SceneName": "f***na",
        "SceneIcon": "",
        "Actions": [ // Actions数组中的每项的字段，为空则不出现
          {
            "ActionType": 0,
            "ProductId": "R32*****0EU",
            "DeviceName": "df***yY",
            "Data": "{\"brightness\": 25}",
            "AliasName": "1***45" // 说明：AliasName为设备别名。
          },
          {
            "ActionType": 1,
            "Data": "100"
          }
        ],
        "UserId": "1",
        "CreateTime": 1578557536,
        "UpdateTime": 1578560598,
        "Flag": 0, // Flag : 0 为正常的场景，1：场景中的设备异常，不在家庭中
        "Status": 0 // 场景运行状态 0未运行，1运行中
      }
    ],
  }
}
```

```
"Total": 1
}
}

// IconUrl为具体设备的url
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InternalServerError.InternalServerErrorDB	DB 错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.NoPermission	用户对该设备无权限
UnsupportedOperation.SceneRunInProgress	场景执行中

删除手动智能联动

最近更新时间：2021-06-07 17:27:51

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppDeleteScene）用于删除手动智能联动。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppDeleteScene
ScenelId	String	是	手动智能联动 ID

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "AccessToken": "***v2",
  "RequestId": "req_1",
  "ScenelId": "s_527cb52*****b453cb6a46f653",
  "Action": "AppDeleteScene"
}
```

输出示例: 成功

```
{"Response":{"RequestId":"req_1"}}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InternalServerError.InternalServerErrorDB	DB 错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.NoPermission	用户对该设备无权限
UnsupportedOperation.SceneRunInProgress	场景执行中

修改手动智能联动

最近更新时间：2021-06-07 17:27:35

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppModifyScene）用于修改手动智能联动。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppModifyScene
ScenelId	String	是	手动智能联动 ID
SceneName	String	是	手动智能联动的名称
Scenelcon	String	是	手动智能联动的背景图片地址
Actions	Array of Action	是	手动智能联动动作列表

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "Actions": [
```

```
{
  "ActionType": 0,
  "ProductId": "R32****0EU",
  "DeviceName": "df****yY",
  "Data": "{\"brightness\": 25}"
},
{
  "AccessToken": "***v2",
  "RequestId": "req_1",
  "Action": "AppModifyScene",
  "SceneId": "s_527cb5*****3cb6a46f653",
  "SceneName": "name",
  "SceneIcon": "****icon"
}
```

输出示例: 成功

```
{"Response":{"RequestId":"req_1"}}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InternalServerError.InternalServerErrorDB	DB 错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.NoPermission	用户对该设备无权限
UnsupportedOperation.SceneRunInProgress	场景执行中

执行手动智能联动

最近更新时间：2021-06-07 17:27:18

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppRunScene）用于执行手动智能联动。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppRunScene
ScenelId	String	是	手动智能联动 ID

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "AccessToken": "***v2",
  "RequestId": "req_1",
  "ScenelId": "s_527cb5*****53cb6a46f653",
  "Action": "AppRunScene"
}
```

输出示例: 成功

```
{"Response":{"RequestId":"req_1"}}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InternalServerError.InternalServerErrorDB	DB 错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.NoPermission	用户对该设备无权限
UnsupportedOperation.SceneRunInProgress	场景执行中

自动智能

创建自动智能联动

最近更新时间：2021-06-07 17:26:46

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppCreateAutomation）用于创建自动智能联动。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppCreateAutomation
FamilyId	String	是	自动智能联动所属家庭 ID
Icon	String	是	自动智能联动的背景图片地址
Name	String	是	自动智能联动的名称
Status	int	是	自动智能联动开关状态。0：关闭，1：启用
MatchType	int	是	条件匹配类型。0：全部条件满足 1：其中一个条件满足
Conditions	Array of Condition	是	自动智能联动触发条件
Actions	Array of Action	是	自动智能联动动作列表
EffectiveBeginTime	String	是	生效开始时间
EffectiveEndTime	String	是	生效结束时间
EffectiveDays	String	是	生效日期由0和1组成的7位数字，0表示不执行，1表示执行，第1位为周日，依次表示周一至周六

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
AutomationId	String	自动智能联动 ID

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppCreateAutomation",
  "FamilyId": "f_9***c1",
  "Icon": "https://www.icon.com/",
  "Name": "autoName",
  "Status": 0, //联动的开关 0：关闭 1：启用
  "MatchType": 1, //条件匹配类型 0：条件全部与 1：条件全部或
  "EffectiveBeginTime": "00:00", // 【两个新增参数，用来表示开始和结束时间】
  "EffectiveEndTime": "23:59",
  "EffectiveDays": "1111111", // 由0和1组成的7位数字，0表示不执行，1表示执行，第1位为周日，依次表示周一至周六
  "Conditions": [
    {
      "CondId": "abc", //条件ID，保证在一个联动下唯一即可
      "CondType": 0, //条件类型 0：设备属性值条件 1：定时条件
      "Property": {
        "ProductId": "N***4",
        "DeviceName": "g***ev",
        "AliasName": "devName",
        "IconUrl": "productIcon",
        "PropertyId": "switch", //设备的属性Id
        "Op": "eq", //条件操作符 eq 等于 ne 不等于 gt 大于 lt 小于 ge 大等于 le 小等于
        "Value": 1 //比较的值
      }
    },
    {
      "CondId": "efg",
      "CondType": 1,
```

```
"Timer": {
  "Days": "1000000", // 由0和1组成的7位数字，0表示不执行，1表示执行，第1位为周日，依次表示周一至周六
  "TimePoint": "09:30" // 触发时间，24小时制，例如"14:00"
}
},
"Actions": [
  {
    "ActionType": 0, // 0: 则为设备动作，具体参数设置为Data 0：设备动作 1：延时 2:场景 3：通知
    "ProductId": "R32***0EU",
    "DeviceName": "df***yY",
    "Data": "{\"brightness\": 25}", // ActionType为 0：设备动作参数 为1：延时时间单位秒 2：执行场景Id 3:通知内容
    "AliasName": "12345",
    "IconUrl": "url"
  },
  {
    "ActionType": 1, //1: 则为延时动作，具体延时时间为Data，值为秒
    "Data": "100"
  }
],
"AccessToken": "***v2"
}
```

输出示例: 成功

```
{
  "Response": {
    "AutomationId": "a_cd61dd*****d21d6aeffea",
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InternalServerError.InternalServerErrorDB	DB 错误

错误码	描述
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.NoPermission	用户对该设备无权限
LimitExceeded	数量超过限制

获取自动智能联动列表

最近更新时间：2021-06-07 17:26:31

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppGetAutomationList）用于获取自动智能联动列表。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppGetAutomationList
FamilyId	String	是	家庭 ID

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
List	Array of AutomationListItem	自动智能联动列表

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppGetAutomationList",
  "FamilyId": "f_9****c1",
```

```
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "List": [
      {
        "AutomationId": "a_cd61*****d21d6aeffea",
        "Icon": "https://www.***.com/",
        "Name": "autoName",
        "Status": 0
      }
    ],
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InternalServerError.InternalServerErrorDB	DB 错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.NoPermission	用户对该设备无权限

获取自动智能联动详情

最近更新时间：2021-06-07 17:26:17

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppDescribeAutomation）用于获取自动智能联动详情。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppDescribeAutomation
AutomationId	String	是	自动智能联动 ID

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
Data	Object of Automation	自动智能联动实体

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppDescribeAutomation",
  "AutomationId": "a_cd61dd*****7d21d6aeffa",
```

```
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": {
      "AutomationId": "a_cd61d*****d6aeffea",
      "Icon": "https://www.****.com/",
      "Name": "autoName",
      "Status": 0,
      "MatchType": 1,
      "EffectiveBeginTime": "00:00", // 【两个新增参数，用来表示开始和结束时间】
      "EffectiveEndTime": "23:59",
      "EffectiveDays": "1111111", // 由0和1组成的7位数字，0表示不执行，1表示执行，第1位为周日，依次表示周一至周六
      "Conditions": [
        {
          "CondId": "abc",
          "CondType": 0,
          "Property": {
            "ProductId": "NJ****LZT4",
            "DeviceName": "gw***v",
            "AliasName": "d****me",
            "IconUrl": "productIcon",
            "PropertyId": "switch",
            "Op": "eq",
            "Value": 1
          }
        },
        {
          "CondId": "efg",
          "CondType": 1,
          "Timer": {
            "Days": "1000000",
            "TimePoint": "09:30"
          }
        }
      ],
    }
  }
}
```



```
"Actions": [
  {
    "ActionType": 0,
    "ProductId": "R3****0EU",
    "DeviceName": "df***yY",
    "Data": "{\"brightness\": 25}",
    "AliasName": "12345",
    "IconUrl": "url"
  },
  {
    "ActionType": 1,
    "Data": "100"
  }
],
"RequestId": "req_1"
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InternalServerError.InternalServerErrorDB	DB 错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.NoPermission	用户对该设备无权限

删除自动智能联动

最近更新时间：2021-06-07 17:25:56

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppDeleteAutomation）用于获取自动智能联动详情。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppDeleteAutomation
AutomationId	String	是	自动智能联动 ID

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId。

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppDeleteAutomation",
  "AutomationId": "a_cd61d*****1d6aeffea",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{"Response":{"RequestId":"req_1"}}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InternalServerError.InternalServerErrorDB	DB 错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.NoPermission	用户对该设备无权限

修改自动智能联动

最近更新时间：2021-06-07 17:25:34

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppModifyAutomation）用于修改自动智能联动。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppModifyAutomation
AutomationId	String	是	自动智能联动 ID
Icon	String	是	新建自动智能联动的背景图片地址
Name	String	是	自动智能联动的名称
Status	int	是	自动智能联动开关状态，0：关闭，1：启用
MatchType	int	是	条件匹配类型，0：全部条件满足 1：其中一个条件满足
Conditions	Array of Condition	是	自动智能联动触发条件
Actions	Array of Action	是	自动智能联动动作列表
EffectiveBeginTime	String	是	生效开始时间
EffectiveEndTime	String	是	生效结束时间
EffectiveDays	String	是	生效日期 由0和1组成的7位数字，0表示不执行，1表示执行，第1位为周日，依次表示周一至周六

3. 输出参数

名称	类型	描述

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppCreateAutomation",
  "AutomationId": "a_cd61ddbe*****47d21d6aeffa",
  "Icon": "https://www.***.com/",
  "Name": "autoName",
  "Status": 0, //联动的开关 0：关闭 1：启用
  "MatchType": 1, //条件匹配类型 0：条件全部与 1：条件全部或
  "EffectiveBeginTime": "00:00", // 【两个新增参数，用来表示开始和结束时间】
  "EffectiveEndTime": "23:59",
  "EffectiveDays": "1111111", // 由0和1组成的7位数字，0表示不执行，1表示执行，第1位为周日，依次表示周一至周六
  "Conditions": [
    {
      "CondId": "abc", //条件ID，保证在一个联动下唯一即可
      "CondType": 0, //条件类型 0：设备属性值条件 1：定时条件
      "Property": {
        "ProductId": "N****4",
        "DeviceName": "gw**v",
        "AliasName": "devName",
        "IconUrl": "productIcon",
        "PropertyId": "switch", //设备的属性Id
        "Op": "eq", //条件操作符 eq 等于 ne 不等于 gt 大于 lt 小于 ge 大等于 le 小等于
        "Value": 1 //比较的值
      },
      {
        "CondId": "efg",
        "CondType": 1,
        "Timer": {
```

```
"Days": "1000000", // 由0和1组成的7位数字, 0表示不执行, 1表示执行, 第1位为周日, 依次表示周一至周六
"TimePoint": "09:30" // 触发时间, 24小时制, 比如"14:00"
}
},
{
  "Actions": [
    {
      "ActionType": 0, // 0: 则为设备动作, 具体参数设置为Data 0: 设备动作 1: 延时 2:场景 3: 通知
      "ProductId": "R3****0EU",
      "DeviceName": "df****yY",
      "Data": "{\"brightness\": 25}", // ActionType为 0: 设备动作参数 为1: 延时时间单位秒 2: 执行场景Id 3:通知内容
      "AliasName": "12345",
      "IconUrl": "url"
    },
    {
      "ActionType": 1, //1: 则为延时动作, 具体延时时间为Data, 值为秒
      "Data": "100"
    }
  ],
  "AccessToken": "***v2"
}
```

输出示例: 成功

```
{"Response":{"RequestId":"req_1"}}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InternalServerError.InternalServerErrorDB	DB 错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.NoPermission	用户对该设备无权限
LimitExceeded	数量超过限制

修改自动智能联动状态

最近更新时间：2021-06-07 17:24:08

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/appapi。

本接口（AppModifyAutomationStatus）用于修改自动智能联动状态。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppModifyAutomationStatus
AutomationId	String	是	自动智能联动 ID
Status	int	是	自动智能联动开关状态，0：关闭，1：启用

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId。

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppGetAutomationList",
  "AutomationId": "a_cd61dd*****1d6aeffea",
  "Status": 1, //联动的开关 0：关闭 1：启用
```

```
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{"Response":{"RequestId":"req_1"}}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InternalServerError.InternalServerErrorDB	DB 错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.NoPermission	用户对该设备无权限

数据查询

获取设备物模型数据

最近更新时间：2021-04-08 17:00:24

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppGetDeviceData）用于获取设备物模型数据。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppGetDeviceData
ProductId	String	是	产品 ID
DeviceName	String	是	设备名

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
Data	String	返回数据

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppGetDeviceData",
```

```
"ProductId":"22F9Y****O",
"DeviceName":"light1",
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": "{ \"brightness\": { \"Value\": 8, \"LastUpdate\": 1552904784552 }, \"color\": { \"Value\": 1, \"LastUpdate\": 1552904784552 }, \"light_switch\": { \"Value\": 0, \"LastUpdate\": 1552904784552 }, \"name\": { \"Value\": \"light1\", \"LastUpdate\": 1552904784552 } }",
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

6. 错误信息

名称	类型	描述
Code	String	错误码
Message	String	错误信息

获取设备物模型历史数据

最近更新时间：2021-05-14 14:32:57

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppGetDeviceDataHistory）用于获取设备物模型历史数据。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权。
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId。
Action	String	是	公共参数，本接口取值：AppGetDeviceDataHistory
ProductId	String	是	产品 ID
DeviceName	String	是	设备名
FieldName	String	是	查询的属性名称
Context	String	否	翻页游标，首次查询时，可不带
MinTime	Int	是	开始时间，毫秒时间戳
MaxTime	Int	是	结束时间，毫秒时间戳
Limit	Int	是	单页数据量

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
Context	String	翻页游标
FieldName	String	查询的属性名称
Listover	Bool	数据是否查询完，True 表示未完可继续读取，False 表示已全部读取完

名称	类型	描述
Results	Array of DataHistoryItem	返回数据

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68",
  "Action": "AppGetDeviceDataHistory",
  "ProductId": "22F9Y6II7O",
  "DeviceName": "light1",
  "MinTime": 1560949017000,
  "MaxTime": 1561035417000,
  "Limit": 20,
  "FieldName": "light_switch",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "Context": "abc",
    "FieldName": "brightness",
    "Listover": false,
    "RequestId": "f92406b3-5a9a-4fe8-bc43-45e3d794bb68",
    "Results": [
      {
        "Time": "1560949025494",
        "Value": "77"
      }
    ]
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

获取设备的历史事件

最近更新时间：2021-08-27 10:18:23

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口(AppListEventHistory)用于获取设备的历史事件。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后,获取的访问Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	AppListEventHistory
ProductId	String	是	产品 ID
DeviceName	String	是	设备名称
Type	String	否	搜索的事件类型：alert 表示告警，fault 表示故障，info 表示信息，为空则表示查询上述所有类型事件
StartTime	Int64	否	起始时间（Unix 时间戳，秒级），为0表示当前时间
EndTime	Int64	否	结束时间（Unix 时间戳，秒级），为0表示当前时间
Context	String	否	搜索上下文, 用作查询游标
Size	Int64	否	单次获取的历史数据项目的最大数量
EventId	String	否	事件标识符，可以用来指定查询特定的事件，如果不指定，则查询所有事件。

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
Context	String	搜索上下文, 用作查询游标

名称	类型	描述
Total	Int64	搜索结果数量
Listover	Bool	搜索结果是否已经结束
EventHistory	Array	搜集结果集

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "AccessToken": "accesstoken*****acc",
  "RequestId": "026fe48b-0c31-4a7c-aaf6-83ba3e9s4bcf",
  "ProductID": "ABCDE12345",
  "DeviceName": "DEV",
  "StartTime": 0,
  "EndTime": 0,
  "Type": "info",
  "Size": 5,
  "Context": "",
  "Action": "AppListEventHistory"
}
```

输出示例: 成功

```
{
  "Response": {
    "ClientToken": "1ee12*****7df703e0d",
    "OutputParams": "",
    "RequestId": "026fe48b-0c31-4a7c-aaf6-83ba3e9s4bcf",
    "Context": "DnF1ZXJ5VGhbkZld*****SUQ0UWVPZUIMLXI1QnBkbGc=",
    "EventHistory": [{
      "TimeStamp": 1553484903,
      "ProductId": "product1",
      "DeviceName": "light3",
      "EventId": "status_report",
    }
  ]
}
```

```
"Type": "info",
>Data": "{\"status\": 0,\"message\": \"test message:11:35:03.288277\"}"
}]
"Total": 1
}
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token无效

设备聚合数据查询

最近更新时间：2021-09-09 11:36:42

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi

本接口（AppGetDeviceAggsData）用于查询某段时间粒度内用户所绑定设备数值型的累积值、平均值等聚合数值。

2. 输入参数

名称	类型	必选	描述
Action	String	是	AppGetDeviceAggsData
DeviceId	String	是	设备 ID，由"产品 ID/DeviceName"组成
AggMethod	String	是	聚合类型（sum/avg/min/max 之一）
Interval	String	是	聚合时间（10m/30m/1h/1d 等）
FieldName	String	是	聚合的属性名，必须是整数或者浮点数
MinTime	UINT	否	聚合的开始时间，毫秒精度，缺省是 EndTime 24小时前
MaxTime	UINT	否	聚合的结束时间，毫秒精度，缺省是当前时间

3. 输出参数

名称	类型	描述
Date	String	文本格式的时间段开始时间
TimestampMs	UINT	毫秒时间戳（时间段开始时间）
Data	Float	聚合类型（avg/min/max 之一）

4. 示例

输入示例：

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
```

```
"Action":"AppGetDeviceAggsData","RequestId":"YZb*****G22","AccessToken":"nin*****01",
"DeviceId":"QD*****NB/dev***",
"AggMethod":"avg",
"Interval":"10m",
"FieldName":"brightness",
"MinTime":0,
"MaxTime":0
}
```

输出示例：成功

```
{
  "Response":{
    "RequestId":"YZ*****22",
    "Results":[
      {
        "Date":"2020-07-21T19:50:00.000+08:00",
        "TimestampMs":1595332200000,
        "Data":42.41379310344828
      },
      {
        "Date":"2020-07-21T20:10:00.000+08:00",
        "TimestampMs":1595333400000,
        "Data":55.06896551724138
      },
      {
        "Date":"2020-07-21T20:20:00.000+08:00",
        "TimestampMs":1595334000000,
        "Data":44.48275862068966
      },
      {
        "Date":"2020-07-22T15:20:00.000+08:00",
        "TimestampMs":1595402400000,
        "Data":44.6551724137931
      }
    ]
  }
}
```

5. 错误码

错误码	描述
UnauthorizedOperation.APPNoPermissionToStudioProduct	小程序或 App 无权限操作该产品
InvalidParameterValue.NoPermission	用户对该设备无权限
ResourceNotFound.DeviceNotExist	设备未创建或是已删除

位置服务

获取设备当前位置

最近更新时间：2021-05-14 14:17:47

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppGetDeviceLocation）用于获取蜂窝网络设备的当前位置。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppGetDeviceLocation
DeviceId	String	是	设备 ID，由 ProductId/DeviceName 拼接组成
CoordType	int	否	坐标系： 1：代表百度地图坐标系 2：代表高德地图或腾讯地图坐标系 3：代表 wgs84 坐标系

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
Data	Object of Position	位置详情

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
```

```
{
  "RequestId": "req_1",
  "Action": "AppGetDeviceLocation",
  "DeviceId": "QDA*****B/de*****01",
  "CoordType": 3
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": {
      "Ret": 0,
      "ErrMsg": "",
      "CreateTime": 1599013684000****0,
      "Longitude": 102.83191,
      "Latitude": 24.325504
    },
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

获取设备历史位置

最近更新时间：2021-05-14 14:14:35

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppGetDeviceLocationHistory）用于获取蜂窝网络设备的历史位置。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后，获取的访问 Token
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId。
Action	String	是	公共参数，本接口取值：AppGetDeviceLocationHistory
DeviceId	String	是	设备 ID，由 ProductId/DeviceName 拼接组成
MinTime	int	是	查询开始时间（毫秒级 UNIX 时间戳），支持最近1个月范围
MaxTime	int	是	查询结束时间（毫秒级 UNIX 时间戳），支持最近1个月范围

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同
Data	Array of Position	位置列表

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppGetDeviceLocationHistory",
```

```
"DeviceId": "QDA*****B/de*****01",
"MaxTime": 0,
"MinTime": 0
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": {
      "Ret": 0,
      "ErrMsg": "",
      "Pos": [{
        "Location": {
          "Longitude": 102.83191,
          "Latitude": 24.325504
        },
        "CreateTime": 1599013554000000000
      }, {
        "Location": {
          "Longitude": 102.83191,
          "Latitude": 24.325504
        },
        "CreateTime": 1599013569000000000
      }, {
        "Location": {
          "Longitude": 102.83191,
          "Latitude": 24.325504
        },
        "CreateTime": 1599013574000000000
      }, {
        "Location": {
          "Longitude": 102.83191,
          "Latitude": 24.325504
        },
        "CreateTime": 1599013579000000000
      }, {
        "Location": {
          "Longitude": 102.83191,
```

```
"Latitude": 24.325504
},
"CreateTime": 1599013604000000000
}, {
"Location": {
"Longitude": 102.83191,
"Latitude": 24.325504
},
"CreateTime": 1599013609000000000
}, {
"Location": {
"Longitude": 102.83191,
"Latitude": 24.325504
},
"CreateTime": 1599013684000000000
}]
},
"RequestId": "aaabbbb2"
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

创建设备地理围栏

最近更新时间：2021-03-30 10:10:23

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppCreateFence）用于创建设备地理围栏。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppCreateFence
ProductId	String	是	产品 ID
DeviceName	String	是	设备名称
FenceName	String	是	设备地理围栏名称
FenceArea	String	是	设备地理围栏区域范围
FenceDesc	String	是	设备地理围栏描述
AlertCondition	String	是	触发条件
FenceEnable	Boolean	是	启用状态
Method	String	是	通知方式

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
FenceId	string	设备地理围栏 ID

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppCreateFence",
  "ProductId": "NJ***T4",
  "DeviceName": "g***v",
  "FenceName": "appFence",
  "FenceArea": "{\n  \"type\": \"FeatureCollection\",\n  \"features\": [\n    {\n      \"type\": \"Feature\",\n      \"geometry\": {\n        \"type\": \"Polygon\",\n        \"coordinates\": [[[113.9498,22.5319],[113.9709,22.5319],[113.9709,22.5723],[113.9498,22.5723],[113.9498,22.5319]]]]\n      },\n      \"properties\": null\n    }\n  ]\n}",
  "FenceDesc": "test",
  "AlertCondition": "leave",
  "FenceEnable": true,
  "Method": "app push",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "FenceId": 7,
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InternalServerError.InternalServerErrorDB	DB 错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

错误码	描述
InvalidParameterValue.NoPermission	用户对该设备无权限
UnauthorizedOperation	无权操作
UnauthorizedOperation.APPNoPermissionToStudioProduct	App 对操作该产品无权限
UnauthorizedOperation.FenceDupKeyExist	存在重复围栏
UnauthorizedOperation.SpaceDupKeyExist	存在重复位置空间
ResourceNotFound	资源不存在
ResourceNotFound.StudioProductNotExist	产品尚未创建或已被删除

获取设备围栏告警事件列表

最近更新时间：2021-04-08 17:32:52

1. 接口描述

接口请求域名： `iot.cloud.tencent.com/api/exploreropen/tokenapi`。

本接口（AppGetFenceEventList）用于获取设备地理围栏告警事件列表。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppGetFenceEventList
FenceId	Int64	是	设备地理围栏 ID
ProductId	String	是	产品 ID
DeviceName	String	是	设备名称
FenceEnable	Boolean	是	启用状态
StartTime	Int	是	起始时间（Unix 时间戳，秒级），为0表示当前时间-24h
EndTime	Int	是	结束时间（Unix 时间戳，秒级），为0表示当前时间
Offset	Int	否	默认值为0，消息偏移量
Limit	Int	否	默认值为0，最大返回消息条数，最大值为 50

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
List	Array of DeviceFenceEvent	设备地理围栏告警事件列表

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppGetFenceEventList",
  "ProductId": "NJ2****ZT4",
  "FenceId": 8,
  "DeviceName": "g****",
  "StartTime": 1604307550000,
  "EndTime": 1604309654000,
  "FenceEnable": true,
  "Offset": 0,
  "Limit": 10,
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "List": [
      {
        "EventType": "enter",
        "CreateTime": 1604314065,
      }
    ],
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalError	内部错误

错误码	描述
InternalServerError.InternalServerErrorDB	DB 错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.NoPermission	用户对该设备无权限
UnauthorizedOperation	无权操作
UnauthorizedOperation.APPNoPermissionToStudioProduct	App 对操作该产品无权限
ResourceNotFound	资源不存在
ResourceNotFound.StudioProductNotExist	产品尚未创建或已被删除
ResourceNotFound.FenceNotExist	围栏未创建或是已删除

获取设备地理围栏列表

最近更新时间：2021-03-30 10:10:31

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppGetFenceList）用于获取设备地理围栏列表。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppGetFenceList
ProductId	String	是	产品 ID
DeviceName	String	是	设备名称
Offset	Int	否	消息偏移量
Limit	Int	否	最大返回消息条数，最大值为 50

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId
List	Array of Fence	设备地理围栏列表

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
```

```
{
  "RequestId": "req_1",
  "Action": "AppGetFenceList",
  "ProductId": "NJ****T4",
  "DeviceName": "gw****ev",
  "Offset": 0,
  "Limit": 10,
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "List": [
      {
        "FenceId": 8,
        "FenceName": "appFence",
        "FenceDesc": "test",
        "FenceArea": "{\"type\":\"FeatureCollection\",\"features\": [{\"type\":\"Feature\",\"geometry\":{\"type\":\"Polygon\",\"coordinates\": [[[113.9498,22.5319],[113.9709,22.5319],[113.9709,22.5723],[113.9498,22.5723],[113.9498,22.5319]]]},\"properties\":null}]}\",
        "AlertCondition": "leave",
        "FenceEnable": true,
        "Method": "app push",
        "CreateTime": 1604314065,
        "UpdateTime": 1604314065
      }
    ],
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InternalServerError.InternalServerErrorDB	DB 错误

错误码	描述
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

删除设备地理围栏

最近更新时间：2021-03-30 10:10:34

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppDeleteFence）用于删除设备地理围栏。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	AppDeleteFence
FenceId	Int64	是	设备地理围栏 ID
ProductId	String	是	产品 ID
DeviceName	String	是	设备名称
AlertCondition	String	是	触发条件
FenceEnable	Boolean	是	启用状态
Method	String	是	通知方式

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
```

```
"RequestId": "req_1",
"Action": "AppDeleteFence",
"ProductId": "NJ****T4",
"FenceId": 7,
"DeviceName": "gw****ev",
"AlertCondition": "enter",
"FenceEnable": true,
"Method": "app push",
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InternalServerError.InternalServerErrorDB	DB 错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.NoPermission	用户对该设备无权限
UnauthorizedOperation	无权操作
UnauthorizedOperation.APPNoPermissionToStudioProduct	App 对操作该产品无权限
UnauthorizedOperation.FenceDupKeyExist	存在重复围栏
UnauthorizedOperation.SpaceDupKeyExist	存在重复位置空间
ResourceNotFound	资源不存在
ResourceNotFound.StudioProductNotExist	产品尚未创建或已被删除

错误码	描述
ResourceNotFound.FenceNotExist	围栏未创建或是已删除

修改设备地理围栏

最近更新时间：2021-03-30 10:10:39

1. 接口描述

接口请求域名： `iot.cloud.tencent.com/api/exploreropen/tokenapi`。

本接口（AppModifyFence）用于修改设备地理围栏。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppModifyFence
FenceId	Int64	是	设备地理围栏 ID
ProductId	String	是	产品 ID
DeviceName	String	是	设备名称
FenceName	String	是	设备地理围栏名称
FenceArea	String	是	设备地理围栏区域范围
FenceDesc	String	是	设备地理围栏描述
AlertCondition	String	是	触发条件
FenceEnable	Boolean	是	启用状态
Method	String	是	通知方式

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "req_1",
  "Action": "AppModifyFence",
  "ProductId": "NJ***T4",
  "FenceId": 7,
  "DeviceName": "gw***ev",
  "FenceName": "appFence",
  "FenceArea": "{ \"type\": \"FeatureCollection\", \"features\": [ { \"type\": \"Feature\", \"geometry\": { \"type\": \"Polygon\", \"coordinates\": [[[ [113.9498, 22.5319], [113.9709, 22.5319], [113.9709, 22.5723], [113.9498, 22.5723], [113.9498, 22.5319] ] ] ] }, \"properties\": { \"type\": \"Polygon\", \"coordinates\": [ [ [113.9498, 22.5319], [113.9709, 22.5319], [113.9709, 22.5723], [113.9498, 22.5723], [113.9498, 22.5319] ] ] ] }, \"properties\": { \"type\": \"Polygon\", \"coordinates\": [ [ [113.9498, 22.5319], [113.9709, 22.5319], [113.9709, 22.5723], [113.9498, 22.5723], [113.9498, 22.5319] ] ] ] } } }\",
  "FenceDesc": "test",
  "AlertCondition": "enter",
  "FenceEnable": true,
  "Method": "app push",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InternalServerError.InternalServerErrorDB	DB 错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效

错误码	描述
InvalidParameterValue.NoPermission	用户对该设备无权限
UnauthorizedOperation	无权操作
UnauthorizedOperation.APPNoPermissionToStudioProduct	App 对操作该产品无权限
UnauthorizedOperation.FenceDupKeyExist	存在重复围栏
UnauthorizedOperation.SpaceDupKeyExist	存在重复位置空间
ResourceNotFound	资源不存在
ResourceNotFound.StudioProductNotExist	产品尚未创建或已被删除
ResourceNotFound.FenceNotExist	围栏未创建或是已删除

修改设备地理围栏启用状态

最近更新时间：2021-03-30 10:10:43

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi。

本接口（AppModifyFenceStatus）用于修改设备地理围栏启用状态。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求 ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppModifyFenceStatus
FenceId	Int64	是	设备地理围栏 ID
ProductId	String	是	产品 ID
DeviceName	String	是	设备名称
AlertCondition	String	是	触发条件
FenceEnable	Boolean	是	启用状态
Method	String	是	通知方式

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求 ID，与入参相同，定位问题时，需提供该次请求的 RequestId

4. 示例

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
```



```
"RequestId": "req_1",
"Action": "AppModifyFenceStatus",
"ProductId": "NJ****T4",
"FenceId": 7,
"DeviceName": "gw****ev",
"AlertCondition": "enter",
"FenceEnable": true,
"Method": "app push",
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "RequestId": "req_1"
  }
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InternalServerError.InternalServerErrorDB	DB 错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token 无效
InvalidParameterValue.NoPermission	用户对该设备无权限
UnauthorizedOperation	无权操作
UnauthorizedOperation.APPNoPermissionToStudioProduct	App 对操作该产品无权限
UnauthorizedOperation.FenceDupKeyExist	存在重复围栏
UnauthorizedOperation.SpaceDupKeyExist	存在重复位置空间
ResourceNotFound	资源不存在
ResourceNotFound.StudioProductNotExist	产品尚未创建或已被删除

错误码	描述
ResourceNotFound.FenceNotExist	围栏未创建或是已删除

长连接通信

注册监听

最近更新时间：2021-05-17 11:08:42

1. 接口描述

接口请求域名：iot.cloud.tencent.com/ws/explorer

本接口（DeviceRegistrationMonitor）用于向 WS 服务端进行订阅，订阅成功后可以通过 ws.onmessage 的监听获取到 WS 服务端实时推送设备上下线状态及属性数据。

2. 输入参数

名称	类型	必选	描述
action	String	是	公共参数，本接口取值：ActivePush
reqId	String	是	公共参数，标示请求序列,唯一请求 ID，可自行生成
params	Object of ActionPushParam	是	注册设备的时候传入的参数

3. 输出参数

名称	类型	描述
reqId	String	公共参数，标示请求序列，与入参相同
error	String	状态码，为空表示正确
error_message	String	错误详细说明，为空表示正确
data	Object of ActionPushData	监听之后的服务器回包

4. 示例

示例1

输入示例

```
const ws = new WebSocket("wss://iot.cloud.tencent.com/ws/explorer")

ws.send({
  "action": "ActivePush",
```

```
"reqId":"req0001",
"params":{
  "DeviceIds": ["产品ID/DeviceName","产品ID/DeviceName"],
  "AccessToken":"f6*****3aad"
}
})
```

输出示例：成功

```
{
  "error": "",
  "error_message": "",
  "data": {
    "deviceIds": ["产品ID/DeviceName","产品ID/DeviceName"]
  },
  "reqId": "req0001"
}
```

输出示例：失败

```
{
  "error": "",
  "error_message": "",
  "data": {
    "deviceIds": []
  },
  "reqId": "req0001"
}
```

心跳

最近更新时间：2021-05-17 11:09:06

1. 接口描述

接口请求域名： iot.cloud.tencent.com/ws/explorer。

设备保活，通过该接口更新设备时间戳，保持设备监听状态，同时保持 WebSocket 连接。可以多次调用，推荐每60s调用一次。

2. 输入参数

名称	类型	必选	描述
action	String	是	公共参数，本接口取值：YunApi
reqId	String	是	公共参数，标示请求序列，唯一请求 ID，可自行生成
params	Object	是	调用接口传入的详细参数，请参见 YunApiParam

3. 输出参数

名称	类型	描述
reqId	String	公共参数，标示请求序列，与入参相同
error	String	状态码,为空表示正确
error_message	String	错误详细说明，为空表示正确
data	Object	心跳接口的服务器回包，请参见 HeartBeatData

4. 示例

输入示例

```
const ws = new WebSocket("wss://iot.cloud.tencent.com/ws/explorer")
ws.send({
  "action": "YunApi",
  "reqId": "req0001",
  "params": {
    "Action": "AppDeviceTraceHeartBeat",
    "AccessToken": "f6*****3aad",
```

```
"RequestId":"rest-client",
"ActionParams": {
"DeviceIds": ["产品ID/DeviceName","产品ID/DeviceName"],
}
}
})
```

输出示例：成功

```
{
"error": "",
"error_message": "",
"data": {
"Response": {
"RequestId": "rest-client"
}
},
"reqId": "req0001"
}
```

输出示例：失败（服务器错误）

```
{
"error": "-1",
"error_message": "后台连接出错，请联系管理员",
"data": {},
"reqId": "req0001"
}
```

输出示例：失败（逻辑错误）

```
{
"error": "",
"error_message": "",
"data": {
"Response": {
"Error": {
"Code": "UnauthorizedOperation.APPNoPermissionToStudioProduct",

```

```
"Message": "App对操作该产品无权限"  
},  
"RequestId": "rest-client"  
}  
},  
"reqId": "req0001"  
}
```

设备状态推送

最近更新时间：2021-09-09 14:42:29

1. 接口描述

接口请求域名：iot.cloud.tencent.com/ws/explorer。

设备状态推送用于在小程序或 App 实时获取用户绑定设备的上下线状态、设备上报的属性与事件以及设备行为执行结果。需要成功调用注册设备监听接口，通过监听 WebSocket 的 OnMessage 获取设备状态与属性值的实时推送，若获取到的 event.data.push 为 true，则代表该条消息为设备状态变更的主动推送。

2. 输出参数

名称	类型	描述
push	Boolean	该条消息为设备状态变更的主动推送
action	String	固定为 DeviceChange，代表设备状态变更
params	String	返回设备信息，其中 Payload 需用 base64 解码

3. 示例

示例1

输入示例

```
ws.onmessage = (event) => {
  let data;
  try {
    data = JSON.parse(event.data);
  } catch (e) {
    console.log(`onMessage parse event.data error: ${event.data}`);
    return;
  }

  if (data.push) {
    this.emit('push', data);
  } else if (typeof data.reqId !== 'undefined' && this.requestHandlerMap.has(data.reqId)) {
    this.requestHandlerMap.get(data.reqId)(null, data);
  }
};
```



```
{
  "error": "",
  "error_message": "",
  "data": {
    "deviceId": []
  },
  "reqId": "req0001"
}
```

Type 类型说明

Type类型	描述
StatusChange	设备状态变化消息，包括设备上线与下线消息通知
Shadow	影子消息
Property	属性消息
Template	物模型消息
Event	事件消息
Action	行为消息
UserDefined	用户自定义

SubType 说明

SubType类型	描述
Online	设备上线
Offline	设备下线
Report	设备上报
Push	服务端推送到设备

❓ 说明:

关于不同消息类型的 params 更多的说明，把 params json 解析出来之后。

Template 消息例子

消息例子

```
{
  "Time": "2019-04-11T11:25:10+08:00",
  "Type": "Template",
  "SubType": "Report",
  "Topic": "",
  "Payload": "eyJ0eXBlljoldXBkYXRlliwic3Rh dGUiO n sicmVwb3J0ZWQiO n sldGVzdF9udW1fc3RhcnQiOjcsI
mxpZ2h0X3N3aXRjaCI6MSwiY29sb3liOjE sImJyaWdodG5lc3MiOjQxfX0sInZlcnNpb24iOjE1NSwiY2xpZW
50VG9rZW4iOiJBUEktQ29udHJvbERldmljZURhdGEtMTU1NDk1MzExMCJ9",
  "Seq": 2,
  "DeviceId": "US4CJ11DIK/LIAO"
}
```

Payload base64 解码之后

```
{
  "type": "update",
  "state": {
    "reported": {
      "test_num_start": 7,
      "light_switch": 1,
      "color": 1,
      "brightness": 41
    }
  }
}
```

```
},
"version": 155,
"clientToken": "API-ControlDeviceData-1554953110"
}
```

```
"Type": "Property",
"SubType": "Report",
"Topic": "$thing/up/property/WTJ TZX0MF2/device_property001",
"Payload": "eyJ0YW1lc3RhbnXAiOiAwLCAicGFyYW1zIjogeyJicmInaHRuZXNzljogNTI9LCAibWV0aG9kljogIn
JlcG9ydClslCJjbGllbnRUb2tlibil6lCJ0bTlwMTkxMDMwMTA0MDUzXzEifQ==",
"Seq": 6,
"DeviceId": "WTJ TZX0MF2/device_property001"
}
```

Payload base64 解码之后

```
{
  "method": "report",
  "params": {
    "timestamp": 0,
    "params": {
      "brightness": 59
    },
    "method": "report",
    "clientToken": "tm20191030104053_1"
  }
}
```

StatusChange 消息示例

消息例子

```
{
  "Time": "2019-03-28T15:17:33+08:00",
  "Type": "StatusChange",
  "SubType": "Online",
  "Topic": "",
  "Payload": "",
  "Seq": 6
}
```

```
{
  "Time": "2019-03-28T15:17:33+08:00",
  "Type": "StatusChange",
```

```
"SubType": "Offline",
"Topic": "",
"Payload": "",
"Seq": 7
}
```

消息例子

Payload base64 解码之后

```
{
  "method": "action",
  "clientToken": "136947493::req_1",
  "actionId": "light_blink",
  "timestamp": 1620912845,
  "params": {
    "color": 1,
    "time": 6,
    "total_time": 60
  }
}
```

```
{
  "Time": "2021-05-13T21:34:05+08:00",
  "Type": "Action",
```

```
"SubType": "Report",
"Topic": "",
"Payload": "eyJtZXRob2QiOiJhY3Rpb25fcmVwbHkiLCJjbGllbnRUb2tIbIl6ljEzNjk0NzQ5Mzo6cmVxXzEiLCJjb2RlIljowLCJzdGF0dXMiOiJzdWNjliwcmVzcg9uc2UiOmsicmVzdWx0IjogMX19",
"Seq": 1620912845,
"DeviceId": "0CPSFIRP28/testble"
}
```

Payload base64 解码之后

```
{
  "method": "action_reply",
  "clientToken": "136947493::req_1",
  "code": 0,
  "status": "succ",
  "response": {
    "result": 1
  }
}
```

文件管理

查询设备指定文件信息

最近更新时间：2021-08-23 17:02:23

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi

本接口(AppDescribeDeviceResource)用于 查询设备的指定文件信息。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后,获取的访问Token
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	AppDescribeDeviceResource
DeviceId	String	是	设备 ID
ResourceName	String	是	厂商侧的文件名称

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求ID，与入参相同
Resource	OBJECT	文件的详细信息

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
```

```
"RequestId": "1pZb3sz1N",
"Action": "AppDescribeDeviceResource",
"DeviceId": "T0IDHQ3AOQ/light100",
"ResourceName": "test"
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{"Response":{"Resource":{"ProductId":"T0IDHQ3AOQ","DeviceName":"light100","ResourceName":"test","ResourceType":"jpg","ResourceVer":"0.0.1","Status":0,"CreateTime":1576747101,"UpdateTime":1576747523},"RequestId":"1pZb3sz1N"}}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token无效

获取下载文件

最近更新时间：2021-08-23 17:02:27

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi

本接口(AppGetResourceDownloadUrl)用于获取文件下载地址。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后,获取的访问Token
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	AppGetResourceDownloadUrl
ProductId	String	是	产品 ID
UserResourceName	String	是	用户侧的文件名称
EffectiveTime	Int	是	有效时间
ResourceVer	String	是	文件的版本

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求ID，与入参相同
DownloadUrl	String	下载的URL

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
```

```
{
  "RequestId": "1pZb113sz1N",
  "Action": "AppGetResourceDownloadUrl",
  "ProductId": "BKTTNGIQOG",
  "UserResourceName": "retest",
  "ResourceVer": "0.0.1",
  "EffectiveTime": 1626329512,
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{
  "Response": {
    "DownloadUrl": "https://gz-g-resource-1256872341.cos.ap-guangzhou.myqcloud.com/res%2F96*****7f1135e7",
    "RequestId": "1pZb113sz1N"
  }
}
```

5. 错误码

错误码	描述
InternalError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token无效

获取文件上传地址

最近更新时间：2021-08-23 17:02:30

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi

本接口(AppGetResourceUploadURL)用于 获取文件上传地址。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后,获取的访问Token
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	AppGetResourceDownloadUrl
ProductId	String	是	产品 ID
UserResourceName	String	是	用户侧的文件名称
FileSize	Int	是	文件大小
ResourceVer	String	是	文件的版本

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求ID，与入参相同
UploadUrl	String	上传的URL
ResourceName	String	厂商侧的文件名称

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "1pZb113sz1N",
  "Action": "AppGetResourceDownloadUrl",
  "ProductId": "BKTTNGIQOG",
  "UserResourceName": "retest",
  "ResourceVer": "0.0.1",
  "FileSize": 5179,
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{ "Response":
  { "UploadUrl": "https://gz-g-resource-1256872341.cos.ap-guangzhou.myqcloud.com/res%2F96*****
*****7f1135e7",
    "ResourceName": "USER_72375312314273792_RES_retest", "RequestId": "1pZb113sz1N" }
}
```

5. 错误码

错误码	描述
InternalError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token无效

上传后创建文件

最近更新时间：2021-08-23 17:02:33

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi

本接口(AppCreateProductResource)用于上传后创建(登记)文件。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后,获取的访问Token
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	AppCreateProductResource
ProductId	String	是	产品 ID
UserResourceName	String	是	用户侧的文件名称
FileSize	Int	是	文件的文件字节大小
ResourceVer	String	是	文件的版本
FileHash	String	是	文件的hash值

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求ID，与入参相同
ResourceName	String	厂商侧的文件名称

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "1pZb3sz1N",
  "Action": "AppCreateProductResource",
  "ProductId": "BKTTNGIQOG",
  "UserResourceName": "retest",
  "ResourceVer": "0.0.1",
  "FileHash": "2a8116bfda285cecd77562decd99cc0f",
  "FileSize": 5179,
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{ "Response":
  { "ResourceName": "USER_72375312314273792_RES_retest",
    "RequestId": "1pZb3sz1N" } }
```

5. 错误码

错误码	描述
InternalError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token无效

下发文件到设备

最近更新时间：2021-08-23 17:02:36

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi

本接口(AppControlDeviceResource)用于下发文件到设备。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，用户通过微信号、手机或邮箱账号登录成功后,获取的访问Token
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuld。定位问题时，需提供该次请求的 RequestId
Action	String	是	AppCreateProductResource
DeviceId	String	是	设备 ID
ResourceName	String	是	厂商侧的文件名称
ResourceVer	String	是	文件的版本
Method	String	是	控制命令，只能为update或者delete。下发或者升级填update；删除则填delete。

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求ID，与入参相同

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
```

```
"RequestId": "1pZb3sz1N",
"Action": "AppControlDeviceResource",
"DeviceId": "BKTTNGIQOG/dev01",
"ResourceName": "test",
"ResourceVer": "0.0.1",
"Method": "update",
"AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec"
}
```

输出示例: 成功

```
{"Response":{"RequestId":"1pZb3sz1N"}}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token无效

音乐服务

歌单歌曲

最近更新时间：2021-09-01 20:01:58

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi

本接口(playlist_song)用于获取歌单歌曲。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值： AppKugouUserCommand
KGCommand	String	是	子命令，本接口取值：playlist_song
DeviceId	String	是	设备Id 由产品ID和设备Name组成
KugouParams	Object of KugouParams_playlist_song	是	子命令的参数

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
ErrorCode	Integer	请求返回的错误码，0为没有错误。
ErrorMsg	String	请求返回的错误信息。
Data	Object of Data_playlist_song	返回结果

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "test0191119-012345",
  "Action": "AppKugouUserCommand",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
  "DeviceId": "LAEG4YJE1A/dev01",
  "KGCommand": "playlist_song",
  "KugouParams": {...}
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": {...}
  },
  "ErrorCode": 0, "ErrorMsg": "",
  "RequestId": "keyi20191119-012345"
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token无效
InvalidParameterValue.InternalRPCError	调用超时
ResourceNotFound	资源不存在
ResourceNotFound.DeviceNotExist	设备未创建或是已删除

错误码	描述
ResourceNotFound.StudioProductNotExist	产品尚未创建或已被删除
ResourceNotFound.AppNotExists	应用未创建或是已删除
UnsupportedOperation	不支持的操作
UnsupportedOperation.DeviceNotGateway	不是网关设备
UnsupportedOperation.NotBindVirtualDevice	无法绑定虚拟设备
UnsupportedOperation.DeviceNotBind	子设备未绑定到网关设备
UnsupportedOperation.CannotBindSameFamily	该设备已经绑定请勿重复绑定
LimitExceeded	超出限制
LimitExceeded.FamilyDeviceExceedLimit	设备数量超出限制
UnauthorizedOperation	无权操作
UnauthorizedOperation.APPNoPermissionToStudioProduct	APP对操作该产品无权限

获取歌曲播放链接

最近更新时间：2021-09-01 19:50:55

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi

本接口(song_url)用于获取歌曲播放链接。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值： AppKugouUserCommand
KGCommand	String	是	子命令，本接口取值：song_url
DeviceId	String	是	设备Id 由产品ID和设备Name组成
KugouParams	Object of KugouParams_song_url	是	子命令的参数

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
ErrorCode	Integer	请求返回的错误码，0为没有错误。
ErrorMsg	String	请求返回的错误信息。
Data	Object of Data_song_url	返回结果

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "test0191119-012345",
  "Action": "AppKugouUserCommand",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
  "DeviceId": "LAEG4YJE1A/dev01",
  "KGCommand": "song_url",
  "KugouParams": {...}
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": {...}
  },
  "ErrorCode": 0, "ErrorMsg": "",
  "RequestId": "keyi20191119-012345"
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token无效
InvalidParameterValue.InternalRPCError	调用超时
ResourceNotFound	资源不存在
ResourceNotFound.DeviceNotExist	设备未创建或是已删除
ResourceNotFound.StudioProductNotExist	产品尚未创建或已被删除

错误码	描述
ResourceNotFound.AppNotExists	应用未创建或是已删除
UnsupportedOperation	不支持的操作
UnsupportedOperation.DeviceNotGateway	不是网关设备
UnsupportedOperation.NotBindVirtualDevice	无法绑定虚拟设备
UnsupportedOperation.DeviceNotBind	子设备未绑定到网关设备
UnsupportedOperation.CannotBindSameFamily	该设备已经绑定请勿重复绑定
LimitExceeded	超出限制
LimitExceeded.FamilyDeviceExceedLimit	设备数量超出限制
UnauthorizedOperation	无权操作
UnauthorizedOperation.APPNoPermissionToStudioProduct	APP对操作该产品无权限

每日推荐

最近更新时间：2021-09-01 19:50:50

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi

本接口(awesome_everyday)用于获取每日推荐歌单。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuld。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值： AppKugouUserCommand
KGCommand	String	是	子命令，本接口取值： awesome_everyday
DeviceId	String	是	设备Id 由产品ID和设备Name组成
KugouParams	Object of KugouParams_awesome_everyday	是	子命令的参数

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
ErrorCode	Integer	请求返回的错误码，0为没有错误。
ErrorMsg	String	请求返回的错误信息。
Data	Object of Data_awesome_everyday	返回结果

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "test0191119-012345",
  "Action": "AppKugouUserCommand",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
  "DeviceId": "LAEG4YJE1A/dev01",
  "KGCommand": "awesome_everyday",
  "KugouParams": {...}
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": {...}
  },
  "ErrorCode": 0, "ErrorMsg": "",
  "RequestId": "keyi20191119-012345"
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token无效
InvalidParameterValue.InternalRPCError	调用超时
ResourceNotFound	资源不存在
ResourceNotFound.DeviceNotExist	设备未创建或是已删除

错误码	描述
ResourceNotFound.StudioProductNotExist	产品尚未创建或已被删除
ResourceNotFound.AppNotExists	应用未创建或是已删除
UnsupportedOperation	不支持的操作
UnsupportedOperation.DeviceNotGateway	不是网关设备
UnsupportedOperation.NotBindVirtualDevice	无法绑定虚拟设备
UnsupportedOperation.DeviceNotBind	子设备未绑定到网关设备
UnsupportedOperation.CannotBindSameFamily	该设备已经绑定请勿重复绑定
LimitExceeded	超出限制
LimitExceeded.FamilyDeviceExceedLimit	设备数量超出限制
UnauthorizedOperation	无权操作
UnauthorizedOperation.APPNoPermissionToStudioProduct	APP对操作该产品无权限

批量查询歌曲信息

最近更新时间：2021-09-01 19:50:45

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi

本接口(song_infos)用于批量查询歌曲信息。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值：AppKugouUserCommand
KGCommand	String	是	子命令，本接口取值：song_infos
DeviceId	String	是	设备Id 由产品ID和设备Name组成
KugouParams	Object of KugouParams_song_infos	是	子命令的参数

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
ErrorCode	Integer	请求返回的错误码，0为没有错误。
ErrorMsg	String	请求返回的错误信息。
Data	Object of Data_song_infos	返回结果

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "test0191119-012345",
  "Action": "AppKugouUserCommand",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
  "DeviceId": "LAEG4YJE1A/dev01",
  "KGCommand": "song_infos",
  "KugouParams": {...}
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": {...}
  },
  "ErrorCode": 0, "ErrorMsg": "",
  "RequestId": "keyi20191119-012345"
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token无效
InvalidParameterValue.InternalRPCError	调用超时
ResourceNotFound	资源不存在
ResourceNotFound.DeviceNotExist	设备未创建或是已删除
ResourceNotFound.StudioProductNotExist	产品尚未创建或已被删除

错误码	描述
ResourceNotFound.AppNotExists	应用未创建或是已删除
UnsupportedOperation	不支持的操作
UnsupportedOperation.DeviceNotGateway	不是网关设备
UnsupportedOperation.NotBindVirtualDevice	无法绑定虚拟设备
UnsupportedOperation.DeviceNotBind	子设备未绑定到网关设备
UnsupportedOperation.CannotBindSameFamily	该设备已经绑定请勿重复绑定
LimitExceeded	超出限制
LimitExceeded.FamilyDeviceExceedLimit	设备数量超出限制
UnauthorizedOperation	无权操作
UnauthorizedOperation.APPNoPermissionToStudioProduct	APP对操作该产品无权限

推荐歌单

最近更新时间：2021-09-01 19:50:39

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi

本接口(playlist_awesome)用于获取推荐歌单。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuld。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值： AppKugouUserCommand
KGCommand	String	是	子命令，本接口取值： playlist_awesome
DeviceId	String	是	设备Id 由产品ID和设备Name组成
KugouParams	Object of KugouParams_playlist_awesome	是	子命令的参数

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
ErrorCode	Integer	请求返回的错误码，0为没有错误。
ErrorMsg	String	请求返回的错误信息。
Data	Object of data_playlist_awesome	返回结果

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "test0191119-012345",
  "Action": "AppKugouUserCommand",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
  "DeviceId": "LAEG4YJE1A/dev01",
  "KGCommand": "playlist_awesome",
  "KugouParams": {...}
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": {...}
  },
  "ErrorCode": 0, "ErrorMsg": "",
  "RequestId": "keyi20191119-012345"
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token无效
InvalidParameterValue.InternalRPCError	调用超时
ResourceNotFound	资源不存在
ResourceNotFound.DeviceNotExist	设备未创建或是已删除

错误码	描述
ResourceNotFound.StudioProductNotExist	产品尚未创建或已被删除
ResourceNotFound.AppNotExists	应用未创建或是已删除
UnsupportedOperation	不支持的操作
UnsupportedOperation.DeviceNotGateway	不是网关设备
UnsupportedOperation.NotBindVirtualDevice	无法绑定虚拟设备
UnsupportedOperation.DeviceNotBind	子设备未绑定到网关设备
UnsupportedOperation.CannotBindSameFamily	该设备已经绑定请勿重复绑定
LimitExceeded	超出限制
LimitExceeded.FamilyDeviceExceedLimit	设备数量超出限制
UnauthorizedOperation	无权操作
UnauthorizedOperation.APPNoPermissionToStudioProduct	APP对操作该产品无权限

新歌首发

最近更新时间：2021-09-01 19:50:34

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi

本接口(awesome_newsong)用于获取首发新歌。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuld。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值： AppKugouUserCommand
KGCommand	String	是	子命令，本接口取值： awesome_newsong
DeviceId	String	是	设备Id 由产品ID和设备Name组成
KugouParams	Object of KugouParams_awesome_newsong	是	子命令的参数

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
ErrorCode	Integer	请求返回的错误码，0为没有错误。
ErrorMsg	String	请求返回的错误信息。
Data	Object of Data_awesome_newsong	返回结果

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "test0191119-012345",
  "Action": "AppKugouUserCommand",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
  "DeviceId": "LAEG4YJE1A/dev01",
  "KGCommand": "awesome_newsong",
  "KugouParams": {...}
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": {...}
  },
  "ErrorCode": 0, "ErrorMsg": "",
  "RequestId": "keyi20191119-012345"
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token无效
InvalidParameterValue.InternalRPCError	调用超时
ResourceNotFound	资源不存在
ResourceNotFound.DeviceNotExist	设备未创建或是已删除

错误码	描述
ResourceNotFound.StudioProductNotExist	产品尚未创建或已被删除
ResourceNotFound.AppNotExists	应用未创建或是已删除
UnsupportedOperation	不支持的操作
UnsupportedOperation.DeviceNotGateway	不是网关设备
UnsupportedOperation.NotBindVirtualDevice	无法绑定虚拟设备
UnsupportedOperation.DeviceNotBind	子设备未绑定到网关设备
UnsupportedOperation.CannotBindSameFamily	该设备已经绑定请勿重复绑定
LimitExceeded	超出限制
LimitExceeded.FamilyDeviceExceedLimit	设备数量超出限制
UnauthorizedOperation	无权操作
UnauthorizedOperation.APPNoPermissionToStudioProduct	APP对操作该产品无权限

用户登出

最近更新时间：2021-09-01 19:50:20

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi

本接口(user_logout)用于用户登出。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值： AppKugouUserCommand
KGCommand	String	是	子命令，本接口取值：user_logout
DeviceId	String	是	设备Id 由产品ID和设备Name组成
KugouParams	Object of KugouParams_user_logout	是	子命令的参数

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
ErrorCode	Integer	请求返回的错误码，0为没有错误。
ErrorMsg	String	请求返回的错误信息。
Data	Object of data_user_logout	返回结果

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "test0191119-012345",
  "Action": "AppKugouUserCommand",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
  "DeviceId": "LAEG4YJE1A/dev01",
  "KGCommand": "user_logout",
  "KugouParams": {...}
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": {...}
  },
  "ErrorCode": 0, "ErrorMsg": "",
  "RequestId": "keyi20191119-012345"
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token无效
InvalidParameterValue.InternalRPCError	调用超时
ResourceNotFound	资源不存在
ResourceNotFound.DeviceNotExist	设备未创建或是已删除
ResourceNotFound.StudioProductNotExist	产品尚未创建或已被删除

错误码	描述
ResourceNotFound.AppNotExists	应用未创建或是已删除
UnsupportedOperation	不支持的操作
UnsupportedOperation.DeviceNotGateway	不是网关设备
UnsupportedOperation.NotBindVirtualDevice	无法绑定虚拟设备
UnsupportedOperation.DeviceNotBind	子设备未绑定到网关设备
UnsupportedOperation.CannotBindSameFamily	该设备已经绑定请勿重复绑定
LimitExceeded	超出限制
LimitExceeded.FamilyDeviceExceedLimit	设备数量超出限制
UnauthorizedOperation	无权操作
UnauthorizedOperation.APPNoPermissionToStudioProduct	APP对操作该产品无权限

用户信息

最近更新时间：2021-09-01 19:50:03

1. 接口描述

接口请求域名：iot.cloud.tencent.com/api/exploreropen/tokenapi

本接口(user_info)用于获取用户信息。

2. 输入参数

名称	类型	必选	描述
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId
Action	String	是	公共参数，本接口取值： AppKugouUserCommand
KGCommand	String	是	子命令，本接口取值：user_info
DeviceId	String	是	设备Id 由产品ID和设备Name组成
KugouParams	Object of KugouParams_user_info	是	子命令的参数

3. 输出参数

名称	类型	描述
RequestId	String	公共参数，唯一请求ID，与入参相同，定位问题时，需提供该次请求的 RequestId。
ErrorCode	Integer	请求返回的错误码，0为没有错误。
ErrorMsg	String	请求返回的错误信息。
Data	Object of Data_user_info	返回结果

4. 示例

示例1

输入示例

```
POST https://iot.cloud.tencent.com/api/exploreropen/tokenapi HTTP/1.1
content-type: application/json
{
  "RequestId": "test0191119-012345",
  "Action": "AppKugouUserCommand",
  "AccessToken": "8b4a70dd16105f*****18edd4e78a3bb8ec",
  "DeviceId": "LAEG4YJE1A/dev01",
  "KGCommand": "user_info",
  "KugouParams": {...}
}
```

输出示例: 成功

```
{
  "Response": {
    "Data": {...}
  },
  "ErrorCode": 0, "ErrorMsg": "",
  "RequestId": "keyi20191119-012345"
}
```

5. 错误码

错误码	描述
InternalServerError	内部错误
InvalidParameterValue	参数取值错误
InvalidParameterValue.InvalidAccessToken	Token无效
InvalidParameterValue.InternalRPCError	调用超时
ResourceNotFound	资源不存在
ResourceNotFound.DeviceNotExist	设备未创建或是已删除
ResourceNotFound.StudioProductNotExist	产品尚未创建或已被删除

错误码	描述
ResourceNotFound.AppNotExists	应用未创建或是已删除
UnsupportedOperation	不支持的操作
UnsupportedOperation.DeviceNotGateway	不是网关设备
UnsupportedOperation.NotBindVirtualDevice	无法绑定虚拟设备
UnsupportedOperation.DeviceNotBind	子设备未绑定到网关设备
UnsupportedOperation.CannotBindSameFamily	该设备已经绑定请勿重复绑定
LimitExceeded	超出限制
LimitExceeded.FamilyDeviceExceedLimit	设备数量超出限制
UnauthorizedOperation	无权操作
UnauthorizedOperation.APPNoPermissionToStudioProduct	APP对操作该产品无权限

数据结构

最近更新时间：2021-12-17 10:39:08

Action

自动智能联动动作

被如下接口引用：AppCreateAutomation、AppCreateScene、AppDescribeAutomation、AppGetSceneList、AppModifyAutomation、AppModifyScene

名称	类型	必选	描述
ActionType	Int	是	动作类型，0：设备动作 1：延时 2：场景 3：通知
ProductId	String	否	产品ID
DeviceName	String	否	设备名称
Data	String	是	ActionType=0 设备动作参数 ActionType=1 延时时间单位秒 ActionType=2 执行场景Id ActionType=3 通知内容
AliasName	String	否	设备别名或者场景名称
IconUrl	String	否	产品或者场景的图标URL

ActionHistories

设备动作历史

被如下接口引用：AppGetDeviceActionHistories

名称	类型	描述
DeviceName	String	设备名称
ActionId	String	动作Id
ActionName	String	动作名称
ReqTime	int64	请求时间
RspTime	int64	响应时间
InputParams	string	输入参数
OutputParams	string	输出参数
Calling	string	调用方式

名称	类型	描述
ClientToken	string	调用Id
Status	string	调用状态

ActionPushData

监听之后的服务器回包

被如下接口引用：DeviceRegistrationMonitor

名称	类型	描述
devicelds	[String]	返回成功注册监听的设备列表，如果入参的设备ID无返回表示：该用户未绑定要注册监听的设备

ActionPushParam

注册设备的时候传入的参数

被如下接口引用：DeviceRegistrationMonitor

名称	类型	必选	描述
AccessToken	String	是	AccessToken为调用登录接口之后获取的Token
Devicelds	[String]	是	需要注册监听的设备列表，必须是已经绑定到该用户名下的设备

AlarmDetailItem

预警详细信息

被如下接口引用：AppGetInsuranceAlarmDetailList

名称	类型	必选	描述
Name	String	是	预警名称
Reason	String	是	隐患分析
Detail	String	是	预警细节
Suggestion	String	是	处置建议
AlarmTimestamp	Int64	是	预警时间

AlarmSummaryItem

预警摘要信息

被如下接口引用：AppGetInsuranceAlarmSummaryList

名称	类型	必选	描述
Name	String	是	预警名称
Reason	String	是	隐患分析
Suggestion	String	是	处置建议
AlarmTimestamp	Int64	是	预警时间
AlarmCount	Int64	是	总预警数

AppDeviceInfo

App设备信息

被如下接口引用：AppBindSubDeviceInFamily、AppSecureAddDeviceInFamily、AppSigBindDeviceInFamily

名称	类型	描述
DeviceId	String	设备ID
ProductId	String	产品ID
DeviceName	String	设备名
AliasName	String	设备别名
CreateTime	String	创建 unix 时间戳（秒级）
UpdateTime	String	最后更新时间，unix时间戳（秒级）
FamilyId	String	家庭ID
RoomId	String	房间ID
IconUrl	String	图标Url

AppInsuranceDeviceInfo

设备信息

被如下接口引用：AppGetUnBindDeviceList

名称	类型	必选	描述

名称	类型	必选	描述
DeviceId	String	是	设备ID
DeviceNo	String	是	设备SN 或车架号

AppServiceInsuranceInfo

保单信息

被如下接口引用：AppGetInsuranceList

名称	类型	必选	描述
ItemId	String	是	保单类别
ItemType	Int64	是	保单类型 0-测试保单 1-正式保单
InsuranceNo	String	是	保单号
Address	String	是	地址
BeginTime	Int64	是	保单生效时间
EndTime	Int64	是	保单结束时间
Score	Int64	是	评分
AlarmNum	Int64	是	预警条数
Status	Int64	是	保单状态 1-失效 2-保障中 3-待生效
ServiceName	String	是	服务名称
BindDeviceId	String	是	绑定设备ID

Automation

自动智能联动实体

被如下接口引用：AppDescribeAutomation

名称	类型	必选	描述
AutomationId	String	是	自动智能联动 ID
Icon	String	是	自动智能联动的背景图片地址
Name	String	是	自动智能联动的名称

名称	类型	必选	描述
Status	int	是	自动智能联动开关状态，0：关闭，1：启用
MatchType	int	是	条件匹配类型，0：全部条件满足 1：其中一个条件满足
EffectiveBeginTime	String	是	生效开始时间
EffectiveEndTime	String	是	生效结束时间
EffectiveDays	String	是	生效日期 由0和1组成的7位数字，0表示不执行，1表示执行，第1位为周日，依次表示周一至周六
Conditions	Array of Condition	是	自动智能联动触发条件
Actions	Array of Action	是	自动智能联动动作列表

AutomationListItem

自动智能联动信息

被如下接口引用：AppGetAutomationList

名称	类型	必选	描述
AutomationId	String	是	自动智能联动 ID
Icon	String	是	自动智能联动的背景图片地址
Name	String	是	自动智能联动的名称
Status	int	是	自动智能联动开关状态，0：关闭，1：启用

Condition

App 设备信息

被如下接口引用：AppCreateAutomation、AppDescribeAutomation、AppModifyAutomation

名称	类型	必选	描述
CondId	String	是	条件ID
CondType	Int32	是	条件类型，0：设备属性条件 1：定时条件
Property	Property	否	条件属性
Timer	Timer	否	条件触发时间

Data

返回数据

被如下接口引用：device_activation

名称	类型	描述
activation_date	string	设备激活日期

DataHistoryItem

历史数据记录

被如下接口引用：AppGetDeviceDataHistory

名称	类型	描述
Time	String	毫秒时间戳
Value	String	数据取值

Data_accompany_awesome_personal

返回数据

被如下接口引用：accompany_awesome_personal

名称	类型	必选	描述
accompany	Array Of Object of accompany_awesome_personal	否	伴奏个性推荐

Data_accompany_awesome_top

返回数据

被如下接口引用：accompany_awesome_top

名称	类型	必选	描述
accompany	Array Of Object of accompany_awesome_top	否	伴奏推荐列表

Data_accompany_category_info

返回数据

被如下接口引用：accompany_category_info

名称	类型	必选	描述

名称	类型	必选	描述
category_id	String	是	分类id
category_name	String	是	分类名
category_img	String	是	分类图片

Data_accompany_category_list

返回数据

被如下接口引用：accompany_category_list

名称	类型	必选	描述
groups	Array Of groups_accompany_category_list of groups_accompany_category_list	否	伴奏分组
recommends	Array Of recommends_accompany_category_list of recommends_accompany_category_list	否	推荐分类

Data_accompany_category_song

返回数据

被如下接口引用：accompany_category_song

名称	类型	必选	描述
accompany	Array Of Object of accompany_category_song	否	伴奏列表

Data_accompany_favorite_list

被如下接口引用：accompany_favorite_list

名称	类型	必选	描述
playlist_id	String	否	歌单id
total	Integer	否	总数
accompany	Array Of Object of accompany_favorite_list	否	伴奏列表

Data_accompany_free_info

返回数据

被如下接口引用：accompany_free_info

名称	类型	必选	描述
accompany	Array Of accompany_accompany_free_info of accompany_accompany_free_info	是	体验信息

Data_accompany_free_song_url

返回数据

被如下接口引用：accompany_free_song_url

名称	类型	必选	描述
accompany_size	String	是	伴奏大小单位字节
has_original	Integer	否	音频是否有原唱音轨 0:无 1:有
singer_name	String	是	歌手名
playable_code	Integer	是	是否可以播放 枚举值依次对应: 0:可以 1:海外地区不能播放 2:歌曲无版权不能播放 3:会员歌曲, 非会员不能播放 4:付费内容, 须购买才可播放 5:牛方案策略, 非会员不能播放 6:因定向版权下架不能播放 (针对APP有权但设备端无权的情况) 9:未知原因,无权播放
accompany_id	String	是	伴奏 ID
accompany_url	String	是	伴奏下载链接
duration	Integer	是	伴奏时长单位毫秒
singer_id	String	是	歌手 ID
song_name	String	是	歌曲名

Data_accompany_free_top_list

返回数据

被如下接口引用：accompany_free_top_list

名称	类型	必选	描述

名称	类型	必选	描述
groups	Array Of groups_accompany_free_top_list of groups_accompany_free_top_list	否	分组列表

Data_accompany_free_top_song

返回数据

被如下接口引用：accompany_free_top_song

名称	类型	必选	描述
accompany	Array Of Object of accompany_free_top_song	否	伴奏列表

Data_accompany_free_url

返回数据

被如下接口引用：accompany_free_url

名称	类型	必选	描述
playable_code	Integer	是	是否可以播放 枚举值依次对应: 0:可以 1:海外地区不能播放 2:歌曲无版权不能播放 3:会员歌曲, 非会员不能播放 4:付费内容, 须购买才可播放 5:牛方案策略, 非会员不能播放 6:因定向版权下架不能播放 (针对APP有权但设备端无权的情况) 9:未知原因,无权播放
accompany_id	String	是	伴奏 ID
accompany_size	String	是	伴奏大小单位字节
accompany_url	String	是	伴奏下载链接
has_original	Integer	否	音频是否有原唱音轨 0:无 1:有
duration	Integer	是	伴奏时长单位毫秒
singer_name	String	是	歌手名
song_name	String	是	歌曲名

Data_accompany_hq_url

返回数据

被如下接口引用：accompany_hq_url

名称	类型	必选	描述
has_original	Integer	否	音频是否有原唱音轨 0:无 1:有
duration	Integer	是	伴奏时长单位毫秒
song_name	String	是	歌曲名
playable_code	Integer	是	是否可以播放 枚举值依次对应: 0:可以 1:海外地区不能播放 2:歌曲无版权不能播放 3:会员歌曲, 非会员不能播放 4:付费内容, 须购买才可播放 5:牛方案策略, 非会员不能播放 6:因定向版权下架不能播放 (针对APP有权但设备端无权的情况) 9:未知原因,无权播放
accompany_id	String	是	伴奏 ID
accompany_size	String	是	伴奏大小单位字节
singer_name	String	是	歌手名
accompany_url	String	是	伴奏下载链接
singer_id	String	是	歌手 ID

Data_accompany_info

返回数据

被如下接口引用：accompany_info

名称	类型	必选	描述
accompany	Array Of Object of accompany_accompany_info	否	伴奏列表

Data_accompany_lyric

返回数据

被如下接口引用：accompany_lyric

名称	类型	必选	描述

名称	类型	必选	描述
lyric	String	是	base64编码的歌词内容
offset	Integer	是	歌词偏移量
format	String	是	歌词格式

Data_accompany_mv

返回数据

被如下接口引用：accompany_mv

名称	类型	必选	描述
mv_id	String	是	MV id
mv_name	String	是	MV名
duration	Integer	否	MV时长,单位毫秒
offset	Integer	否	mv与伴奏对齐时间偏移量

Data_accompany_newsearch_song

返回数据

被如下接口引用：accompany_newsearch_song

名称	类型	必选	描述
accompany	Array Of Object of accompany_newsearch_song	否	伴奏列表
author	Array Of Object of author_newsearch_song	否	歌手列表

Data_accompany_opus_del

返回数据

被如下接口引用：accompany_opus_del

名称	类型	描述
data	object	objects

Data_accompany_opus_list

返回数据

被如下接口引用：accompany_opus_list

名称	类型	必选	描述
total	Integer	否	总数
opuses	Array Of opuses_accompany_opus_list of opuses_accompany_opus_list	否	opuses

Data_accompany_opus_upload

返回数据

被如下接口引用：accompany_opus_upload

名称	类型	必选	描述
opus_id	String	否	作品id

Data_accompany_opus_url

返回数据

被如下接口引用：accompany_opus_url

名称	类型	必选	描述
url	String	是	作品下载链接
duration	Integer	是	时长单位秒
size	Integer	是	伴奏大小单位字节

Data_accompany_search_song

返回数据

被如下接口引用：accompany_search_song

名称	类型	必选	描述
accompany	Array Of Object of accompany_search_song	否	伴奏搜索
total	Integer	否	总数

Data_accompany_search_tips

被如下接口引用：accompany_search_tips

名称	类型	必选	描述
tips	Array Of String	否	提示

Data_accompany_singer_list

歌手列表

被如下接口引用：accompany_singer_list

名称	类型	必选	描述
singers	Array Of singers_accompany_singer_list of singers_accompany_singer_list	否	热门歌手
letters	Array Of letters_accompany_singer_list of letters_accompany_singer_list	否	letters

Data_accompany_singer_song

返回数据

被如下接口引用：accompany_singer_song

名称	类型	必选	描述
total	Integer	否	总数
accompany	Array Of Object of accompany_singer_song	否	伴奏列表

Data_accompany_songpitch

返回数据

被如下接口引用：accompany_songpitch

名称	类型	必选	描述
pitch	String	是	音高
offset	Integer	是	音高偏移量
type	Integer	是	来源, 0为默认值, 1为算法制作, 2为人工导入
md5sum	String	是	md5签名,用于完整性校验

Data_accompany_theme_list

返回数据

被如下接口引用: accompany_theme_list

名称	类型	必选	描述
themes	Array Of themes_accompany_theme_list of themes_accompany_theme_list	否	主题列表

Data_accompany_theme_song

被如下接口引用: accompany_theme_song

名称	类型	必选	描述
accompany	Array Of Object of accompany_theme_song	否	伴奏列表

Data_accompany_top_list

被如下接口引用: accompany_top_list

名称	类型	必选	描述
groups	Array Of groups_accompany_top_list of groups_accompany_top_list	否	分组列表

Data_accompany_top_song

返回数据

被如下接口引用: accompany_top_song

名称	类型	必选	描述
accompany	Array Of Object of accompany_top_song	否	伴奏列表

Data_accompany_url

返回数据

被如下接口引用: accompany_url

名称	类型	必选	描述
accompany_id	String	是	伴奏 ID

名称	类型	必选	描述
duration	Integer	是	伴奏时长单位毫秒
singer_id	String	是	歌手 ID
song_name	String	是	歌曲名
playable_code	Integer	是	是否可以播放 枚举值依次对应: 0:可以 1:海外地区不能播放 2:歌曲无版权不能播放 3:会员歌曲, 非会员不能播放 4:付费内容, 须购买才可播放 5:牛方案策略, 非会员不能播放 6:因定向版权下架不能播放 (针对APP有权但设备端无权的情况) 9:未知原因,无权播放
status	Integer	是	0:成功获取到伴奏下载信息 1:当日免费体验次数已经用完
accompany_size	String	是	伴奏大小单位字节
accompany_url	String	是	伴奏下载链接
has_original	Integer	否	音频是否有原唱音轨 0:无 1:有
singer_name	String	是	歌手名

Data_album_info

返回数据

被如下接口引用: album_info

名称	类型	描述
album_id	Integer	专辑id
album_name	String	专辑名
album_translate_name	String	专辑译名
album_img	String	专辑封面
intro	String	专辑简介
company	String	唱片公司
singer_id	String	歌手id

名称	类型	描述
singer_name	String	歌手名
total	Integer	歌曲总数
publish_time	String	发布时间
songs	Array of songs_album_info	歌曲列表

Data_asset_purchased_albums

返回数据

被如下接口引用：asset_purchased_albums

名称	类型	描述
total	Integer	总数
albums	Array of albums_asset_purchased_albums	专辑信息

Data_asset_purchased_songs

返回数据

被如下接口引用：asset_purchased_songs

名称	类型	描述
total	Integer	总数
songs	Array of songs_asset_purchased_songs	歌曲列表

Data_awesome_everyday

返回数据

被如下接口引用：awesome_everyday

名称	类型	描述
songs	array of songs_awesome_everyday	歌曲列表

Data_awesome_newsong

返回数据

被如下接口引用：awesome_newsong

名称	类型	描述
total	Integer	总数
songs	Array of songs_awesome_newsong	歌曲列表

Data_awesome_recommend

返回数据

被如下接口引用：awesome_recommend

名称	类型	描述
songs	Array of songs_awesome_recommend	歌曲列表

Data_child_home

返回数据

被如下接口引用：child_home

名称	类型	必选	描述
groups	Array Of groups_child_home of groups_child_home	否	资源分组

Data_child_resource

返回数据

被如下接口引用：child_resource

名称	类型	必选	描述
type	Integer	否	大分类 3-歌单 4-专辑 7-小分类
total	Integer	否	总数
lists	Array Of lists_child_resource of lists_child_resource	否	资源分组

Data_favorite_oper_playlist

返回数据

被如下接口引用：favorite_oper_playlist

名称	类型	描述
playlist_extra_id	String	歌单附加id，移除收藏时使用
playlist_id	String	操作的歌单id

Data_favorite_oper_song

返回数据

被如下接口引用：favorite_oper_song

名称	类型	描述
song_extra_id	String	歌曲附加id，移除收藏时使用
song_id	String	操作的歌曲id

Data_favorite_other_list

返回数据

被如下接口引用：favorite_other_list

名称	类型	描述
playlists	Array of playlists_favorite_other_list	歌单

Data_favorite_self_list

返回数据

被如下接口引用：favorite_self_list

名称	类型	描述
playlists	Array of playlists_favorite_self_list	歌单

Data_favorite_song

返回数据

被如下接口引用：favorite_song

名称	类型	描述
total	Integer	总数

名称	类型	描述
songs	Array of songs_favorite_song	歌曲列表

Data_give_common_record

返回数据

被如下接口引用：give_common_record

名称	类型	描述
status	Integer	
times	Integer	剩余领取次数
collection_time	String	上一次领取时间

Data_give_common_scancodelogin

返回数据

被如下接口引用：give_common_scancodelogin

名称	类型	描述
status	Integer	
order_no	String	订单号(同sp内唯一)
partner_no	String	厂商订单号(同sp内唯一)
create_time	String	订单创建时间
vip_end_time	String	vip有效期终止时间
collection_time	String	上一次领取时间

Data_live_program_list

返回数据

被如下接口引用：live_program_list

名称	类型	必选	描述
programs	Array Of programs_live_program_list of programs_live_program_list	否	节目分组

Data_live_room_status

返回数据

被如下接口引用：live_room_status

名称	类型	必选	描述
status	Integer	是	房间内节目状态 0.停播;1.开播
horizontal	Integer	是	横屏流状态 0.停播;1.开播
vertical	Integer	是	竖屏流状态 0.停播;1.开播

Data_live_room_url

返回数据

被如下接口引用：live_room_url

名称	类型	必选	描述
vertical	Array Of vertical_live_room_url of vertical_live_room_url	否	竖屏流信息
status	String	是	房间内节目状态 0.停播;1.开播
room_id	Integer	否	房间id
expire	Integer	否	链接过期时间
horizontal	Array Of horizontal_live_room_url of horizontal_live_room_url	否	横屏流信息

Data_longaudio_free_home

返回数据

被如下接口引用：longaudio_free_home

名称	类型	必选	描述
groups	Array Of groups_longaudio_free_home of groups_longaudio_free_home	否	资源分组

Data_longaudio_resource

返回数据

被如下接口引用：longaudio_resource

名称	类型	必选	描述
is_end	String	否	是否存在下一页。【特别提醒】：返回结果个数可能少于请求的size，建议以瀑布流的方式设计UI交互
lists	Array Of lists_longaudio_resource of lists_longaudio_resource	否	资源列表

Data_minipauth

返回数据

被如下接口引用：wechat_minipauth

名称	类型	描述
token	string	用户令牌
expire	integer	令牌有效期 unix时间戳，单位秒
userid	string	用户id

Data_monitor_delay

返回值为空

被如下接口引用：monitor_delay

Data_monitor_listen

返回值为空

被如下接口引用：monitor_listen

Data_mv_category_list

返回数据

被如下接口引用：mv_category_list

名称	类型	必选	描述
categories	Array Of categories_mv_category_list of categories_mv_category_list	否	分类列表

Data_mv_category_mv

返回数据

被如下接口引用：mv_category_mv

名称	类型	必选	描述
mvs	Array Of mvs_mv_category_mv of mvs_mv_category_mv	否	MV列表

Data_mv_info

返回数据

被如下接口引用：mv_info

名称	类型	必选	描述
song_id	String	是	歌曲 ID
singer_id	String	是	歌手 ID
singer_name	String	是	歌手名称
author_name	String	否	MV 制作人名称
intro	String	否	简介
tags	tags_mv_info	否	MV 信息
mv_id	String	是	MV ID
mv_img	String	是	MV 图片
mv_name	String	是	MV 名称
accompany_id	String	否	伴奏 ID

Data_mv_infos

返回数据

被如下接口引用：mv_infos

名称	类型	必选	描述
mvInfos	Array Of mvInfos_mv_infos of mvInfos_mv_infos	否	MV 信息

Data_mv_live_tme

返回数据

被如下接口引用：mv_live_tme

名称	类型	必选	描述
mvs	Array Of mvs_mv_live_tme of mvs_mv_live_tme	否	MV 列表

Data_mv_search

返回数据

被如下接口引用：mv_search

名称	类型	必选	描述
mvs	Array Of mvs_mv_search of mvs_mv_search	否	MV 列表
total	Integer	否	总数

Data_mv_trial

返回数据

被如下接口引用：mv_trial

名称	类型	必选	描述
mv_size_ld	Integer	否	流畅版 MV 大小
mv_url_ld	String	否	流畅版 MV 下载链接
mv_size_sd	Integer	否	标清版 MV 大小
mv_url_sd	String	否	标清版 MV 下载链接
mv_size_qhd	Integer	否	高清版 MV 大小
mv_url_qhd	String	否	高清版 MV 下载链接
mv_size_hd	Integer	否	超清版 MV 大小
mv_url_hd	String	否	超清版 MV 下载链接
mv_size_fhd	Integer	否	蓝光版 MV大小
mv_url_fhd	String	否	蓝光版MV 下载链接
duration	Integer	是	MV 时长单位毫秒
mv_id	String	是	MV ID

名称	类型	必选	描述
mv_name	String	是	MV 名
singer_name	String	是	歌手名

Data_mv_url

返回数据

被如下接口引用：mv_url

名称	类型	必选	描述
mv_size_hd	Integer	否	超清版 MV 大小
mv_url_hd	String	否	超清版 MV 下载链接
mv_size_fhd	Integer	否	蓝光版 MV 大小
mv_url_fhd	String	否	蓝光版 MV 下载链接
mv_size_qhd	Integer	否	高清版 MV 大小
mv_url_qhd	String	否	高清版 MV 下载链接
mv_size_sd	Integer	否	标清版 MV 大小
mv_url_sd	String	否	标清版 MV 下载链接
mv_size_ld	Integer	否	流畅版 MV 大小
mv_url_ld	String	否	流畅版 MV 下载链接
duration	Integer	是	MV 时长单位毫秒
mv_id	String	是	MV ID
mv_name	String	是	MV 名
singer_name	String	是	歌手名

Data_playlist_awesome

返回数据

被如下接口引用：playlist_awesome

名称	类型	描述

名称	类型	描述
playlists	Array of playlists_playlist_awesome	歌单

Data_playlist_song

返回数据

被如下接口引用：playlist_song

名称	类型	描述
total	Integer	总数
songs	Array of songs_playlist_song	歌曲列表

Data_playlist_top

返回数据

被如下接口引用：playlist_top

名称	类型	描述
groups	Array of groups_playlist_top	分组列表

Data_radio_list

返回数据

被如下接口引用：radio_list

名称	类型	描述
groups	Array of groups_radio_list	分组列表

Data_radio_song

返回数据

被如下接口引用：radio_song

名称	类型	描述
total	Integer	总数
songs	Array of songs_radio_song	歌曲列表

Data_rigion_home

返回数据

被如下接口引用：rigion_home

名称	类型	必选	描述
groups	Array Of groups_rigion_home of groups_rigion_home	否	资源分组

Data_rigion_oldman_index

返回数据

被如下接口引用：rigion_oldman_index

名称	类型	必选	描述
groups	Array Of groups_rigion_oldman_index of groups_rigion_oldman_index	否	资源分组

Data_search_song

返回数据

被如下接口引用：search_song

名称	类型	必选	描述
total	Integer	是	总数
keyword	String	是	关键词
songs	Array Of songs_search_song of songs_search_song	是	歌曲列表

Data_search_tips

返回数据

被如下接口引用：search_tips

名称	类型	必选	描述
tips	Array Of String	是	tips

Data_search_voice

返回数据

被如下接口引用：search_voice

名称	类型	必选	描述
tts_text	String	是	用于语音提示的文本
songs	songs_search_voice	是	语音搜索歌曲列表

Data_singer_album

返回数据

被如下接口引用：singer_album

名称	类型	描述
total	Integer	歌曲总数
albums	Array of albums_singer_album	专辑列表

Data_singer_info

返回数据

被如下接口引用：singer_info

名称	类型	描述
singer_id	String	歌手id
singer_name	String	歌手名
area	String	歌手地区
translate_name	String	歌手译名
song_count	Integer	歌手歌曲数
singer_img	String	歌手相片

Data_singer_list

返回数据

被如下接口引用：singer_list

名称	类型	描述
singers	Array of singers_singer_list	歌手

Data_singer_song

返回数据

被如下接口引用：singer_song

名称	类型	描述
total	Integer	歌曲总数
songs	Array of songs_singer_song	歌曲列表

Data_song_accompany

返回数据

被如下接口引用：song_accompany

名称	类型	描述
accompany	Array of accompany_song_accompany	伴奏

Data_song_infos

返回数据

被如下接口引用：song_infos

名称	类型	描述
songs	Array of songs_song_infos	歌曲列表

Data_song_lyric

返回数据

被如下接口引用：song_lyric

名称	类型	描述
lyric	string	歌词

Data_song_tolisten

返回数据

被如下接口引用：song_tolisten

名称	类型	描述

名称	类型	描述
songs	Array of songs_song_tolisten	歌曲列表

Data_song_url

返回数据

被如下接口引用：song_url

名称	类型	描述
song_id	String	歌曲id
song_name	String	歌曲名
singer_name	String	歌手名
song_url	String	歌曲下载链接
song_size	String	歌曲大小
song_url_hq	String	歌曲(hq)下载链接
song_size_hq	String	歌曲(hq)大小
song_url_sq	String	歌曲(sq)下载链接
song_size_sq	String	歌曲(sq)大小
duration	Integer	歌曲时长,单位毫秒
playable_code	Integer	是否可以播放 枚举值依次对应: 0:可以 1:海外地区不能播放 2:歌曲无版权不能播放 3:会员歌曲, 非会员不能播放 4:付费内容, 须购买才可播放 5:牛方案策略, 非会员不能播放 6:因定向版权下架不能播放 (针对 APP有权但设备端无权的情况) 9:未知原因,无权播放
try_url	String	歌曲试听链接
try_size	Integer	歌曲试听大小,单位字节
try_begin	Integer	歌曲试听开始,单位毫秒
try_end	Integer	歌曲试听结束,单位毫秒

Data_top_list

返回数据

被如下接口引用：top_list

名称	类型	描述
groups	Array of groups_top_list	分组

Data_top_song

返回数据

被如下接口引用：top_song

名称	类型	描述
total	Integer	总数
songs	songs_top_song	歌曲列表

Data_user_info

返回数据

被如下接口引用：user_info

名称	类型	描述
userid	String	用户 id
nick_name	String	用户昵称
img	String	用户头像
is_vip	Integer	是否vip 0:否 1:是
vip_end_time	String	vip有效期终止时间
car_vip_end_time	String	车机会员有效期终止时间
svip_end_time	String	豪V有效期终止时间

Data_user_logout

返回数据为空

被如下接口引用：user_logout

Data_user_qrcode_auth

返回数据

被如下接口引用：user_qrcode_auth

名称	类型	描述
userid	String	用户 ID
token	String	用户令牌
expire	Integer	令牌有效期 unix时间戳，单位秒

Data_user_qrcode_get

返回数据

被如下接口引用：user_qrcode_get

名称	类型	描述
qrcode	String	登录二维码 url,自行渲染
ticket	String	获取二维码授权信息的凭证

Data_vip_activity_partner_give

返回数据

被如下接口引用：vip_activity_partner_give

名称	类型	必选	描述
create_time	String	否	TME 订单创建时间
vip_end_time	String	否	vip 有效期终止时间
status	Integer	是	
order_no	String	否	TME 订单号(同sp内唯一)
partner_no	String	否	厂商订单号(同sp内唯一)

Data_vip_ksing_info

返回数据

被如下接口引用：vip_ksing_info

名称	类型	必选	描述
userid	String	是	用户 ID
nick_name	String	是	用户昵称
img	String	是	用户头像
is_vip	Integer	是	是否vip 0:否 1:是
vip_end_time	String	是	vip有效期终止时间

Data_vip_ksing_partner_give

返回数据

被如下接口引用：vip_ksing_partner_give

名称	类型	必选	描述
order_no	String	是	TME订单号(同sp内唯一)
partner_no	String	是	厂商订单号(同sp内唯一)
create_time	String	是	TME订单创建时间
vip_end_time	String	是	vip有效期终止时间

Data_vip_svip_common_give

返回数据

被如下接口引用：vip_svip_common_give

名称	类型	必选	描述
order_no	String	是	TME订单号(同sp内唯一)
partner_no	String	是	厂商订单号(同sp内唯一)
create_time	String	是	TME订单创建时间

Data_vip_tme_partner_give

返回数据

被如下接口引用：vip_tme_partner_give

名称	类型	必选	描述

名称	类型	必选	描述
order_no	String	是	TME订单号(同sp内唯一)
partner_no	String	是	厂商订单号(同sp内唯一)
create_time	String	是	TME订单创建时间
vip_end_time	String	是	vip有效期终止时间

Data_wechat_appinfocode

返回数据

被如下接口引用：wechat_appinfocode

名称	类型	描述
ticket	String	获取第三方app信息凭证
req_name	String	拉起的小程序的请求名
req_path	String	拉起的小程序的请求路径

Data_wechat_openlink

返回数据

被如下接口引用：wechat_openlink

名称	类型	描述
ticket	String	获取扫码授权结果的凭证
openlink	String	微信小程序openlink
expire	Integer	令牌有效期 unix时间戳，单位秒

Data_wechat_qrcodeauth

返回数据

被如下接口引用：wechat_qrcodeauth

名称	类型	描述
token	String	用户令牌
userid	String	用户id

名称	类型	描述
expire	Integer	令牌有效期 unix时间戳，单位秒

Data_wechat_qrcodeget

返回数据

被如下接口引用：wechat_qrcodeget

名称	类型	描述
ticket	String	获取扫码授权结果的凭证
qrcode	String	微信小程序二维码图像二进制数据base64编码

Device

设备信息

被如下接口引用：AppGetDeviceStatuses

名称	类型	描述
DeviceName	String	设备名
Online	Int	在线状态
LoginTime	Int64	登录时间
LastDataTime	Int64	最近登录时间

DeviceFenceEvent

设备地理围栏告警事件

被如下接口引用：AppGetFenceEventList

名称	类型	必选	描述
EventType	String	是	设备地理围栏名称
CreateTime	String	否	创建 unix 时间戳（秒级）

DeviceItem

设备状态

被如下接口引用：AppGetInsuranceDeviceStatus

名称	类型	必选	描述
ProductName	String	是	设备展示名称
OnlineStatus	Int	是	设备状态：0: 离线, 1: 在线, 2: 未知

DeviceList

设备列表

被如下接口引用：AppGetFamilyDeviceList、AppGetFamilySubDeviceList、AppGetGatewayBindDeviceList

名称	类型	描述
ProductId	String	设备ID
DeviceName	String	设备名称
DeviceId	String	设备 Id
AliasName	String	设备别名
UserId	String	用户Id
RoomId	String	房间ID
FamilyId	String	家庭ID
IconUrl	String	图标Url
DeviceType	Int32	设备类型（0 普通 1 网关设备 2 子设备）
CreateTime	Int64	创建 unix 时间戳（秒级）
UpdateTime	Int64	最后更新的 unix 时间戳（秒级）

ErrorMessage

错误信息

名称	类型	描述
Code	String	错误码
Message	String	错误信息

EventHistory

搜集结果集

名称	类型	描述
TimeStamp	Int64	事件的时间戳
ProductId	String	事件的产品ID
DeviceName	String	事件的设备名称
EventId	String	事件的标识符ID
Type	String	事件的类型
Data	string	事件的数据

FamilyList

被如下接口引用：AppGetFamilyList

名称	类型	描述
FamilyId	String	家庭 ID
Name	String	名称
Role	Int	角色
CreateTime	Int	创建时间
UpdateTime	Int	最后更新时间

Fence

设备地理围栏信息

被如下接口引用：AppGetFenceList

名称	类型	必选	描述
FenceId	Int64	是	设备地理围栏ID
FenceName	String	是	设备地理围栏名称
FenceArea	String	是	设备地理围栏区域范围
FenceDesc	String	是	设备地理围栏描述
AlertCondition	String	是	触发条件

名称	类型	必选	描述
FenceEnable	Boolean	是	启用状态
Method	String	是	通知方式
CreateTime	String	否	创建 unix 时间戳（秒级）
UpdateTime	String	否	最后更新时间，unix时间戳（秒级）

HeartBeatData

心跳接口的服务器回包

被如下接口引用：HeartBeat

名称	类型	描述
RequestId	String	公共参数，唯一请求ID，与入参相同，定位问题时，需提供该次请求的 RequestId。

HeartBeatParam

设备心跳接口入参

被如下接口引用：HeartBeat

名称	类型	描述
DeviceIds	[String]	设备列表

lotUser

用户信息

被如下接口引用：AppFindUser

名称	类型	描述
UserId	String	用户Id
NickName	String	昵称
Avatar	String	头像
CountryCode	String	手机号国家码
PhoneNumber	String	手机号码
Email	String	邮箱

lotUserDetail

用户信息

被如下接口引用：AppGetUser

名称	类型	描述
UserId	String	用户Id
NickName	String	昵称
Avatar	String	头像
CountryCode	String	手机号国家码
PhoneNumber	String	手机号码
Email	String	邮箱
HasPassword	Int	是否已设置密码，0 未设置，1 已设置
HasWxOpenID	Int	是否已绑定微信，0 未绑定，1 已绑定

KugouParams

请求参数这里为空，传递空对象 {} 即可

被如下接口引用：device_activation

KugouParams_accompany_awesome_personal

请求参数

被如下接口引用：accompany_awesome_personal

名称	类型	必选	描述
size	Integer	是	每页?项
page	Integer	是	第?页

KugouParams_accompany_awesome_top

请求参数

被如下接口引用：accompany_awesome_top

名称	类型	必选	描述
size	Integer	是	每页?项

名称	类型	必选	描述
page	Integer	是	第?页
type	Integer	是	排行榜类型。1: 热门榜 2:飙升榜

KugouParams_accompany_category_info

请求参数

被如下接口引用： accompany_category_info

名称	类型	必选	描述
category_id	String	是	伴奏分类id

KugouParams_accompany_category_list

[object Object]

被如下接口引用： accompany_category_list

KugouParams_accompany_category_song

请求参数

被如下接口引用： accompany_category_song

名称	类型	必选	描述
page	Integer	是	第?页
size	Integer	是	每页?项
category_id	String	是	伴奏分类id

KugouParams_accompany_favorite_list

请求参数

被如下接口引用： accompany_favorite_list

名称	类型	必选	描述
page	Integer	是	第?页
size	Integer	是	每页?项

KugouParams_accompany_free_info

请求参数这里为空，传递空对象 {} 即可
被如下接口引用：accompany_free_info

KugouParams_accompany_free_song_url

请求参数
被如下接口引用：accompany_free_song_url

名称	类型	必选	描述
accompany_id	String	是	伴奏id

KugouParams_accompany_free_top_list

请求参数这里为空，传递空对象 {} 即可
被如下接口引用：accompany_free_top_list

KugouParams_accompany_free_top_song

请求参数
被如下接口引用：accompany_free_top_song

名称	类型	必选	描述
group_id	String	是	分组id
top_id	String	是	榜单、电台id

KugouParams_accompany_free_url

请求参数
被如下接口引用：accompany_free_url

名称	类型	必选	描述
accompany_id	String	是	伴奏id
free_token	String	否	免费令牌

KugouParams_accompany_hq_url

请求参数

被如下接口引用：accompany_hq_url

名称	类型	必选	描述
accompany_id	String	是	伴奏id

KugouParams_accompany_info

请求参数

被如下接口引用：accompany_info

名称	类型	必选	描述
accompany_id	String	是	伴奏id

KugouParams_accompany_lyric

请求参数

被如下接口引用：accompany_lyric

名称	类型	必选	描述
accompany_id	String	是	伴奏id

KugouParams_accompany_mv

请求参数

被如下接口引用：accompany_mv

名称	类型	必选	描述
accompany_id	String	是	伴奏id

KugouParams_accompany_newsearch_song

请求参数

被如下接口引用：accompany_newsearch_song

名称	类型	必选	描述
keyword	String	是	关键词

KugouParams_accompany_opus_del

请求参数

被如下接口引用：accompany_opus_del

名称	类型	必选	描述
opus_id	String	是	作品id

KugouParams_accompany_opus_list

请求参数

被如下接口引用：accompany_opus_list

名称	类型	必选	描述
page	Integer	是	第?页
size	Integer	是	每页?项

KugouParams_accompany_opus_upload

请求参数

被如下接口引用：accompany_opus_upload

名称	类型	必选	描述
opus_hash	String	是	作品hash
opus_desc	String	是	作品描述
krc_id	String	是	歌词id
grade	String	是	等级
opus_name	String	是	作品名
accompany_id	String	是	伴奏id
duration	Integer	是	作品播放时长,单位秒
score	String	是	总分
average_score	String	是	平均分

KugouParams_accompany_opus_url

请求参数

被如下接口引用：accompany_opus_url

名称	类型	必选	描述
opus_id	String	是	作品id

KugouParams_accompany_search_song

请求参数

被如下接口引用：accompany_search_song

名称	类型	必选	描述
page	Integer	是	第?页
size	Integer	是	每页?项
keyword	String	是	关键词

KugouParams_accompany_search_tips

请求参数

被如下接口引用：accompany_search_tips

名称	类型	必选	描述
keyword	String	是	关键词

KugouParams_accompany_singer_list

请求参数

被如下接口引用：accompany_singer_list

名称	类型	必选	描述
type	Integer	是	歌手类型 1：全部 2：男歌手 3：女歌手 4：组合
area	Integer	是	歌手区域 1：华语 2：欧美 3：日韩

KugouParams_accompany_singer_song

请求参数

被如下接口引用：accompany_singer_song

名称	类型	必选	描述
page	Integer	是	第?页
size	Integer	是	每页?项
singer_id	String	是	歌手id

KugouParams_accompany_songpitch

请求参数

被如下接口引用：accompany_songpitch

名称	类型	必选	描述
accompany_id	String	是	伴奏id

KugouParams_accompany_theme_list

请求参数

被如下接口引用：accompany_theme_list

名称	类型	必选	描述
page	Integer	是	第?页
size	Integer	是	每页?项

KugouParams_accompany_theme_song

请求参数

被如下接口引用：accompany_theme_song

名称	类型	必选	描述
page	Integer	是	第?页
size	Integer	是	每页?项
theme_id	String	是	主题id

KugouParams_accompany_top_list

请求参数这里为空，传递空对象 {} 即可

被如下接口引用：accompany_top_list

KugouParams_accompany_top_song

请求参数

被如下接口引用：accompany_top_song

名称	类型	必选	描述
page	Integer	是	第?页
size	Integer	是	每页?项
group_id	String	是	分组id
top_id	String	否	榜单id

KugouParams_accompany_url

请求参数

被如下接口引用：accompany_url

名称	类型	必选	描述
accompany_id	String	是	伴奏id

KugouParams_album_info

请求参数

被如下接口引用：album_info

名称	类型	描述
page	Integer	第?页
size	Integer	每页?项
album_id	String	专辑id

KugouParams_asset_purchased_albums

请求参数

被如下接口引用：asset_purchased_albums

名称	类型	描述
page	Integer	第?页

名称	类型	描述
size	Integer	每页?项

KugouParams_asset_purchased_songs

请求参数

被如下接口引用: asset_purchased_songs

名称	类型	描述
page	Integer	第?页
size	Integer	每页?项

KugouParams_awesome_everyday

请求参数这里为空，传递空对象 {} 即可

被如下接口引用: awesome_everyday

KugouParams_awesome_newsong

请求参数

被如下接口引用: awesome_newsong

名称	类型	描述
page	Integer	第?页
size	Integer	每页?项
top_id	Int	榜单id 1:华语, 2:欧美, 3:韩语, 4:日语

KugouParams_awesome_recommend

请求参数这里为空，传递空对象 {} 即可

被如下接口引用: awesome_recommend

KugouParams_child_home

请求参数这里为空，传递空对象 {} 即可

被如下接口引用: child_home

KugouParams_child_resource

请求参数

被如下接口引用：child_resource

名称	类型	必选	描述
page	Integer	是	第?页
size	Integer	是	每页?项
resource_type	Integer	是	大分类 7-小分类
resource_id	String	是	资源id

KugouParams_favorite_oper_playlist

请求参数

被如下接口引用：favorite_oper_playlist

名称	类型	描述
cmd	Integer	1:收藏歌单, 2:取消收藏
playlist_id	String	歌单id
playlist_extra_id	String	歌单附加id 1.收藏时传入歌单id 2.取消收藏时传入获取歌单列表(或收藏歌单)时返回的同名字段

KugouParams_favorite_oper_song

请求参数

被如下接口引用：favorite_oper_song

名称	类型	描述
cmd	Integer	1:收藏歌单, 2:取消收藏
playlist_id	String	歌单id
song_id	String	歌曲id
song_extra_id	String	歌曲附加id 1.收藏时传入歌曲id 2.取消收藏时传入获取歌单歌曲列表时返回的同名字段

KugouParams_favorite_other_list

请求参数这里为空，传递空对象 {} 即可
被如下接口引用：favorite_other_list

KugouParams_favorite_self_list

请求参数这里为空，传递空对象 {} 即可
被如下接口引用：favorite_self_list

KugouParams_favorite_song

请求参数
被如下接口引用：favorite_song

名称	类型	描述
playlist_id	String	歌单id
page	Integer	第?页
size	Integer	每页?项
type	String	self:自建歌单 other:收藏的歌单, 默认:other

KugouParams_give_common_record

请求参数这里为空，传递空对象 {} 即可
被如下接口引用：give_common_record

KugouParams_give_common_scancodelogin

请求参数
被如下接口引用：give_common_scancodelogin

名称	类型	描述
partner_no	String	厂商订单号(同sp内唯一)

KugouParams_live_program_list

请求参数这里为空，传递空对象 {} 即可
被如下接口引用：live_program_list

KugouParams_live_room_status

请求参数

被如下接口引用：live_room_status

名称	类型	必选	描述
room_id	String	是	房间id

KugouParams_live_room_url

请求参数

被如下接口引用：live_room_url

名称	类型	必选	描述
room_id	String	是	房间id

KugouParams_longaudio_free_home

请求参数这里为空，传递空对象 {} 即可

被如下接口引用：longaudio_free_home

KugouParams_longaudio_resource

请求参数

被如下接口引用：longaudio_resource

名称	类型	必选	描述
page	Integer	是	第?页
size	Integer	是	每页?项
tag_id	String	是	分类tag_id

KugouParams_monitor_delay

请求参数

被如下接口引用：monitor_delay

名称	类型	必选	描述
pid	String	是	牛方案2.0 pid,产品接入时申请获得
device_id	String	是	厂商设备唯一标示，由ascii可见字符组成

名称	类型	必选	描述
client_ver	String	是	客户端版本号，厂商自行填写自家版本号，用于辅助监控统计
platform	String	是	客户端平台版本信息，用于辅助监控统计
timestamp	Integer	是	unix时间戳，单位秒
nonce	String	是	随机字串，由ascii可见字符组成
data	Array Of data_monitor_delay of data_monitor_delay	是	数据

KugouParams_monitor_listen

请求参数

被如下接口引用：monitor_listen

名称	类型	必选	描述
pid	String	是	牛方案2.0 pid,产品接入时申请获得
device_id	String	是	厂商设备唯一标示，由ascii可见字符组成
timestamp	Integer	是	unix时间戳，单位秒
nonce	String	是	随机字串，由ascii可见字符组成
data	Array Of data_monitor_listen of data_monitor_listen	是	数据

KugouParams_mv_category_list

请求参数这里为空，传递空对象 {} 即可

被如下接口引用：mv_category_list

KugouParams_mv_category_mv

请求参数

被如下接口引用：mv_category_mv

名称	类型	必选	描述

名称	类型	必选	描述
page	Integer	是	第?页
size	Integer	是	每页?项
category_id	String	是	分类id
sort	Integer	是	排序 1: 发布时间 2: 标签创建时间 3: 播放量（倒序） 4: 算法 5: mv飙升值

KugouParams_mv_info

请求参数

被如下接口引用: mv_info

名称	类型	必选	描述
mv_id	String	是	mv_id

KugouParams_mv_infos

请求参数

被如下接口引用: mv_infos

名称	类型	必选	描述
mvs_id	Array Of String	是	mvid列表

KugouParams_mv_live_tme

请求参数

被如下接口引用: mv_live_tme

名称	类型	必选	描述
page	Integer	是	第?页
size	Integer	是	每页?项
live_type	Integer	否	1-所有视频, 2-完整演唱会视频

KugouParams_mv_search

请求参数

被如下接口引用：mv_search

名称	类型	必选	描述
userid	String	是	TME用户id
page	Integer	是	第?页
size	Integer	是	每页?项
keyword	String	是	关键词

KugouParams_mv_trial

请求参数

被如下接口引用：mv_trial

名称	类型	必选	描述
mv_id	String	是	mv_id

KugouParams_mv_url

请求参数

被如下接口引用：mv_url

名称	类型	必选	描述
mv_id	String	是	mv_id

KugouParams_playlist_awesome

请求参数

被如下接口引用：playlist_awesome

名称	类型	描述
page	Integer	第?页
size	Integer	每页?项

KugouParams_playlist_song

请求参数

被如下接口引用：playlist_song

名称	类型	描述
playlist_id	String	歌单分类id
page	Integer	第?页
size	Integer	每页?项

KugouParams_playlist_top

请求参数这里为空，传递空对象 {} 即可

被如下接口引用：playlist_top

KugouParams_radio_list

请求参数这里为空，传递空对象 {} 即可

被如下接口引用：radio_list

KugouParams_radio_song

请求参数

被如下接口引用：radio_song

名称	类型	描述
radio_id	String	电台id

KugouParams_rigion_home

请求参数

被如下接口引用：rigion_home

名称	类型	必选	描述
id	String	是	专区id

KugouParams_rigion_oldman_index

请求参数这里为空，传递空对象 {} 即可

被如下接口引用：rigion_oldman_index

KugouParams_search_song

请求参数

被如下接口引用：search_song

名称	类型	必选	描述
page	Integer	是	第?页
size	Integer	是	每页?项
keyword	String	是	关键词

KugouParams_search_tips

请求参数

被如下接口引用：search_tips

名称	类型	必选	描述
keyword	String	是	关键词

KugouParams_search_voice

请求参数

被如下接口引用：search_voice

名称	类型	必选	描述
query	String	是	用户语音在处理后的原始文本
slots	Array Of slots_search_voice of slots_search_voice	是	语音搜索列表

KugouParams_singer_album

请求参数

被如下接口引用：singer_album

名称	类型	描述
singer_id	String	歌手id
page	Integer	第?页
size	Integer	每页?项

名称	类型	描述
sort	Integer	排序 1:最热 2:最新

KugouParams_singer_info

请求参数

被如下接口引用：singer_info

名称	类型	描述
singer_id	String	歌手id

KugouParams_singer_list

请求参数

被如下接口引用：singer_list

名称	类型	描述
page	Integer	第?页
size	Integer	每页?项
area	Integer	歌手地区 0:全部 1:内地 2:欧美 3:日本 4:韩国 5:港台 6:其它
type	Integer	歌手类型 0:全部 1:男 2:女 3:组合

KugouParams_singer_song

请求参数

被如下接口引用：singer_song

名称	类型	描述
singer_id	String	歌手id
page	Integer	第?页
size	Integer	每页?项

KugouParams_song_accompany

请求参数

被如下接口引用：song_accompany

名称	类型	描述
song_id	String	歌曲id

KugouParams_song_infos

请求参数

被如下接口引用：song_infos

名称	类型	描述
songs_id	Array of songs_id_song_infos	歌曲id列表

KugouParams_song_lyric

请求参数

被如下接口引用：song_lyric

名称	类型	描述
song_id	String	歌曲id

KugouParams_song_tolisten

请求参数

被如下接口引用：song_tolisten

名称	类型	描述
song_id	String	歌曲id

KugouParams_song_url

请求参数

被如下接口引用：song_url

名称	类型	描述
song_id	String	歌曲id

KugouParams_top_list

请求参数这里为空，传递空对象 {} 即可

被如下接口引用：top_list

KugouParams_top_song

请求参数

被如下接口引用：top_song

名称	类型	描述
page	Integer	第?页
size	Integer	每页?项
top_id	String	榜单id

KugouParams_user_info

请求参数这里为空，传递空对象 {} 即可

被如下接口引用：user_info

KugouParams_user_logout

请求参数这里为空，传递空对象 {} 即可

被如下接口引用：user_logout

KugouParams_user_qrcode_auth

请求参数

被如下接口引用：user_qrcode_auth

名称	类型	描述
ticket	String	获取二维码授权信息的凭证

KugouParams_user_qrcode_get

请求参数这里为空，传递空对象 {} 即可

被如下接口引用：user_qrcode_get

KugouParams_vip_activity_partner_give

请求参数

被如下接口引用：vip_activity_partner_give

名称	类型	必选	描述
partner_no	String	是	厂商订单号(同sp内唯一)
month	Integer	是	购买月份数
day	Integer	是	1. 购买天数,天数和月数二选一,不可同时为0

KugouParams_vip_ksing_info

请求参数这里为空，传递空对象 {} 即可

被如下接口引用：vip_ksing_info

KugouParams_vip_ksing_partner_give

请求参数

被如下接口引用：vip_ksing_partner_give

名称	类型	必选	描述
day	Integer	是	1. 购买天数,天数和月数二选一,不可同时为0 2. 天数为1为酷狗线上测试充值渠道(请提前联系TME导入配额),酷我Q音目前暂时不支持
month	Integer	是	购买月份数
partner_no	String	是	厂商订单号(同sp内唯一)

KugouParams_vip_svip_common_give

请求参数

被如下接口引用：vip_svip_common_give

名称	类型	必选	描述
partner_no	String	是	厂商订单号(同sp内唯一)
pay_no	String	是	用户支付订单号
day	Integer	是	1. 购买天数,天数和月数二选一,不可同时为0 2. 天数为1为酷狗线上测试充值渠道(请提前联系TME导入配额),酷我Q音目前暂时不支持
month	Integer	是	购买月份数

KugouParams_vip_tme_partner_give

请求参数

被如下接口引用：vip_tme_partner_give

名称	类型	必选	描述
userid	String	是	TME用户id
token	String	否	TME用户token,非首次充值支持免token（KW和QM 传过期或有效token，KG传有效token或空）
partner_no	String	是	厂商订单号(同sp内唯一)
day	Integer	是	1. 购买天数,天数和月数二选一,不可同时为0 2. 天数为1为酷狗线上测试充值渠道(请提前联系TME导入配额),酷我Q音目前暂时不支持
month	Integer	是	购买月份数

KugouParams_wechat_appinfo

请求参数

被如下接口引用：wechat_appinfo

名称	类型	描述
package	String	包名需事先向TME注册

KugouParams_wechat_minipauth

请求参数

被如下接口引用：wechat_minipauth

名称	类型	描述
auth_code	string	获取用户信息凭证

KugouParams_wechat_openlink

请求参数

被如下接口引用：wechat_openlink

名称	类型	描述
package	String	包名需事先向TME注册对应手机APP的包名（如没有对应手机APP可暂时预留一个）

KugouParams_wechat_qrcodeauth

请求参数

被如下接口引用：wechat_qrcodeauth

名称	类型	描述
ticket	String	获取扫码授权结果的凭证

KugouParams_wechat_qrcodeget

请求参数

被如下接口引用：wechat_qrcodeget

名称	类型	描述
package	String	包名需事先向TME注册

Location

经纬度

被如下接口引用：AppGetDeviceLocation、AppGetDeviceLocationHistory

名称	类型	描述
Longitude	float32	经度
Latitude	float32	纬度

MemberList

被如下接口引用：AppGetFamilyMemberList

名称	类型	描述
UserId	String	用户 Id
NickName	String	用户昵称
Avatar	String	头像
Role	Int	角色类型,0 是普通成员，1 是管理员。

MessageAttachment

消息加信息

名称	类型	描述
ShareToken	String	分享设备或邀请成员加入家庭时，用于绑定的凭证

Msgs

消息体

名称	类型	描述
UserId	String	接受者UserId
FromUserId	String	发送者UserId
MsgId	String	消息 Id
Category	Int	主类型（1：设备，2：家庭，3：通知）
MsgType	Int64	类型（101: 设备告警提醒 200：成员添加结果通知 201: 成员删除结果通知 202：社区成员添加结果通知 203: 社区成员删除结果通知 204：邀请成员加入 300: 设备分享结果通知 301: 设备分享邀请）
MsgTitle	String	消息标题
MsgContent	String	消息内容
MsgTimestamp	Int64	消息时间戳
ProductId	String	产品 ID
DeviceName	String	设备名
DeviceAlias	String	设备别名
FamilyId	String	家庭ID
FamilyName	String	家庭名称
RelateUserId	String	关联用户，在家庭消息中，一般指被添加或删除的用户
CreateAt	String	创建时间，RFC3339 格式
Attachments	Object of MessageAttachment	消息加信息

Position

位置

被如下接口引用：AppGetDeviceLocation、AppGetDeviceLocationHistory

名称	类型	描述
Location	Location	经纬度
CreateTime	int64	创建时间

ProductList

关联产品列表

名称	类型	描述
ProductId	String	产品ID
Name	String	产品名称
Description	String	产品描述
UIConfig	String	交互配置信息json字符串，包括产品图片、配网引导等

Products

产品列表

名称	类型	描述
ProductId	String	产品ID
Name	String	产品名称
Description	String	产品描述
DataTemplate	String	产品数据模板
NetType	String	产品网络连接类型
CategoryId	Int32	产品分类ID
ProductType	Int32	产品类型
UpdateTime	Int64	最后更新的 unix 时间戳（秒级）

ProductsConfigRsp

产品界面配置返回值

被如下接口引用：AppGetProductsConfig

名称	类型	描述
ProductId	String	产品ID
Config	String	产品配置

Property

条件属性

被如下接口引用：AppCreateAutomation、AppDescribeAutomation、AppModifyAutomation

名称	类型	必选	描述
ProductId	String	是	产品ID
DeviceName	String	是	设备名称
AliasName	String	否	设备别名
IconUrl	String	否	产品图标URL
PropertyId	String	否	产品属性ID
Op	String	是	条件操作符，eq：等于 ne：不等于 gt：大于 lt：小于 ge：大于等于 le：小于等于
Value	Any	是	比较的值

RiskItem

扣分项结构体

被如下接口引用：AppGetInsuranceEvaluation

名称	类型	必选	描述
Name	String	是	扣分项名称
Reason	String	是	隐患分析
Suggestion	String	是	处置建议

Roomlist

房间列表信息

被如下接口引用：AppGetRoomList

名称	类型	描述
RoomId	String	房间的所属 Id
RoomName	String	房间名称
DeviceNum	Int	房间拥有的设备数
CreateTime	Int	创建时间，unix 时间戳(秒级)
UpdateTime	Int	最后更新时间，unix 时间戳(秒级)

Scene

场景实体

被如下接口引用：AppGetSceneList

名称	类型	必选	描述
Sceneld	String	是	手动智能联动 ID
FamilyId	String	是	手动智能联动所属家庭 ID
SceneName	String	是	手动智能联动的名称
Scenelcon	String	否	手动智能联动的背景图片地址
Actions	Array of Action	是	手动智能联动动作列表
UserId	String	是	用户 ID
CreateTime	int	否	创建时间戳
UpdateTime	int	否	更新时间戳
Flag	int	否	设备状态 0：正常 1：设备异常
Status	int	否	运行状态 0：未运行 1：运行中

ShareDeviceTokenInfo

响应结果

被如下接口引用：AppDescribeShareDeviceToken

名称	类型	描述

名称	类型	描述
FromUserId	String	分享用户Id
FromUserNick	String	分享用户昵称
UserId	String	被分享用户Id
UserNick	String	被分享用户昵称
ProductId	String	产品ID
DeviecName	String	设备名
AliasName	String	设备别名
IconUrl	String	图标URL
BindTime	Int64	绑定的 unix 时间戳（秒级）
ExpireTime	Int64	有效期（秒级）
CreateTime	Int64	创建的 unix 时间戳(秒级)

ShareDevices

分享设备信息

被如下接口引用：AppDescribeShareDevice、AppListUserShareDevices

名称	类型	描述
ProductId	String	产品ID
DeviceName	String	设备名称
DeviceId	String	设备ID，是产品ID/设备名称拼接而来。
IconUrl	String	图标的 url
AliasName	String	设备别名、备注。
CreateTime	String	设备创建时间，这里是 RFC3339 的格式。

ShareUserInfo

分享用户信息

被如下接口引用：AppListShareDeviceUsers

名称	类型	描述
UserId	String	被分享用户Id
NickName	String	昵称
CountryCode	String	国家码
PhoneNumber	String	手机号
BindTime	Int64	绑定时间，秒级 UNIX 时间戳

Timer

条件触发时间

被如下接口引用：AppCreateAutomation、AppDescribeAutomation、AppModifyAutomation

名称	类型	必选	描述
Days	String	是	定时器开启时间，每一位——0:关闭,1:开启, 从左至右依次表示: 周日 周一 周二 周三 周四 周五 周六
TimePoint	String	是	定时器开启时间点

TimerList

定时器列表信息

被如下接口引用：AppGetTimerList

名称	类型	描述
TimerId	String	定时器 ID
TimerName	String	定时器名称
ProductId	String	定时器所属的产品ID
DeviceName	String	定时器对应的设备名称
Days	String	定时器开启时间，每一位——0:关闭,1:开启, 从左至右依次表示: 周日 周一 周二 周三 周四 周五 周六
TimePoint	String	定时器开启时间点
Repeat	Int	是否循环，0: 不需要，1: 需要
Data	String	定时器启动时下发的状态

名称	类型	描述
Status	String	开启状态，0表示关闭，1表示开启。
CreateTime	Int64	创建时间，unix 时间戳(秒级)
UpdateTime	Int64	最后一次更新时间,unix 时间戳(秒级)

TokenResultInfo

用户信息

被如下接口引用：AppGetToken、AppGetTokenByWeiXin

名称	类型	描述
ExpireAt	Int64	截止时间，Unix 秒级时间戳
Token	String	开发平台返回的AccessToken，通过该Token进行登录后的接口请求
CancelAccountTime	Int64	仅当用户提交注销申请（AppUserCancelAccount）后再次登录时返回（仅申请首次登录返回），此字段用来返回注销申请提交时间，提醒用户注销流程已取消，Unix 秒级时间戳

UserSetting

用户设置

被如下接口引用：AppGetUserSetting

名称	类型	必选	描述
EnableWechatPush	Int	否	微信推送
EnableDeviceMessagePush	Int	否	设备信息推送
EnableFamilyMessagePush	Int	否	家庭信息推送
EnableNotifyMessagePush	Int	否	通知信息推送
AllowEditions	Int	否	小程序白名单版本，0 表示仅限家庭版 1 表示家庭版和社区版都允许
UsingEdition	Int	否	小程序当前使用版本，0 表示家庭版 1 表示社区版
TemperatureUnit	String	否	温度单位：C 表示 摄氏度，F 表示 华氏度
Region	String	否	时区设置，有效时区参见：AppGetGlobalConfig 中的 RegionListEN 或 RegionListCN。

名称	类型	必选	描述
NoDisturbTime	String	否	消息免打扰时段，为空则表示未开启免打扰，开启时则用：HH:MM–HH:MM 表示免打扰时段，例如：21:00–08:00
NoDisturbDeviceList	Array of String	否	免打扰设备ID列表
NoDisturbFamilyList	Array of String	否	免打扰家庭列表

YunApiParam

设备心跳接口

被如下接口引用：HeartBeat

名称	类型	必选	描述
AccessToken	String	是	AccessToken为调用登录接口之后获取Token
ActionParams	Object of HeartBeatParam	是	设备心跳接口入参
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId。

accompany_accompany_free_info

体验信息

被如下接口引用：accompany_free_info

名称	类型	必选	描述
accompany_id	String	否	已经体验的伴奏id

accompany_accompany_info

伴奏列表

被如下接口引用：accompany_info

名称	类型	必选	描述

名称	类型	必选	描述
has_original	Integer	否	音频是否有原唱音轨 0:无 1:有
is_hq	Integer	是	是否显示HQ标识 0:不显示 1:显示
mv_id	String	是	MVID
hash_key	String	是	hashkey
song_id	String	是	歌曲id
accompany_id	String	是	伴奏id
song_name	String	是	伴奏名
singer_id	String	是	歌手id
singer_name	String	是	歌手名
duration	Integer	是	伴奏时长单位毫秒
album_img	String	是	专辑图片
has_original	Integer	是	音频是否有原唱音轨 0:无 1:有
is_ktv	Integer	是	是否有KTV 0:没有 1:有
bit_rate	Integer	是	比特率
has_pitch	Integer	是	是否有音高 0:没有 1:有
krc_id	String	是	歌词ID
original_mv_id	String	是	原版本MV ID

accompany_awesome_personal

伴奏个性推荐

被如下接口引用：accompany_awesome_personal

名称	类型	必选	描述
is_hq	Integer	是	是否显示HQ标识 0:不显示 1:显示
accompany_id	String	是	伴奏id
song_name	String	是	伴奏名
singer_id	String	是	歌手id

名称	类型	必选	描述
singer_name	String	是	歌手名
duration	Integer	是	伴奏时长单位毫秒
album_img	String	是	专辑图片
has_original	Integer	是	音频是否有原唱音轨 0:无 1:有
is_ktv	Integer	是	是否有KTV 0:没有 1:有
bit_rate	Integer	是	比特率
has_pitch	Integer	是	是否有音高 0:没有 1:有

accompany_awesome_top

伴奏推荐列表

被如下接口引用：accompany_awesome_top

名称	类型	必选	描述
weight	Integer	是	热度值/飙升值
is_hq	Integer	是	是否显示HQ标识 0:不显示 1:显示
accompany_id	String	是	伴奏id
song_name	String	是	伴奏名
singer_id	String	是	歌手id
singer_name	String	是	歌手名
duration	Integer	是	伴奏时长单位毫秒
album_img	String	是	专辑图片
has_original	Integer	是	音频是否有原唱音轨 0:无 1:有
is_ktv	Integer	是	是否有KTV 0:没有 1:有
bit_rate	Integer	是	比特率
has_pitch	Integer	是	是否有音高 0:没有 1:有

accompany_category_song

伴奏列表

被如下接口引用：accompany_category_song

名称	类型	必选	描述
is_hq	Integer	是	是否显示HQ标识 0:不显示 1:显示
accompany_id	String	是	伴奏id
song_name	String	是	伴奏名
singer_id	String	是	歌手id
singer_name	String	是	歌手名
duration	Integer	是	伴奏时长单位毫秒
album_img	String	是	专辑图片
has_original	Integer	是	音频是否有原唱音轨 0:无 1:有
is_ktv	Integer	是	是否有KTV 0:没有 1:有
bit_rate	Integer	是	比特率
has_pitch	Integer	是	是否有音高 0:没有 1:有

accompany_favorite_list

伴奏列表

被如下接口引用：accompany_favorite_list

名称	类型	必选	描述
is_hq	Integer	是	是否显示HQ标识 0:不显示 1:显示
has_original	Integer	否	音频是否有原唱音轨 0:无 1:有
singer_name	String	是	歌手名
accompany_id	String	是	伴奏id
song_name	String	是	伴奏名字
duration	Integer	是	伴奏时长单位毫秒
singer_id	String	是	歌手id
album_img	String	是	专辑图片

名称	类型	必选	描述
is_ktv	Integer	是	是否有KTV 0:没有 1:有
bit_rate	Integer	是	比特率
has_pitch	Integer	是	是否有音高 0:没有 1:有

accompany_free_top_song

伴奏列表

被如下接口引用：accompany_free_top_song

名称	类型	必选	描述
is_hq	Integer	是	是否显示HQ标识 0:不显示 1:显示
has_original	Integer	否	音频是否有原唱音轨 0:无 1:有
singer_name	String	是	歌手名
free_token	String	是	免费令牌
accompany_id	String	是	伴奏id
song_name	String	是	伴奏名字
duration	Integer	是	伴奏时长单位毫秒
singer_id	String	是	歌手id
album_img	String	是	专辑图片
is_ktv	Integer	是	是否有KTV 0:没有 1:有
bit_rate	Integer	是	比特率
has_pitch	Integer	是	是否有音高 0:没有 1:有

accompany_newsearch_song

伴奏列表

被如下接口引用：accompany_newsearch_song

名称	类型	必选	描述
song_hot	Integer	是	热度

名称	类型	必选	描述
accompany_id	String	是	伴奏id
song_name	String	是	伴奏名
singer_id	String	是	歌手id
singer_name	String	是	歌手名
duration	Integer	是	伴奏时长单位毫秒
album_img	String	是	专辑图片
is_ktv	Integer	是	是否有KTV 0:没有 1:有
bit_rate	Integer	是	比特率
has_pitch	Integer	是	是否有音高 0:没有 1:有

accompany_search_song

伴奏搜索

被如下接口引用：accompany_search_song

名称	类型	必选	描述
song_hot	Integer	是	热度
accompany_id	String	是	伴奏id
song_name	String	是	伴奏名
singer_id	String	是	歌手id
singer_name	String	是	歌手名
duration	Integer	是	伴奏时长单位毫秒
album_img	String	是	专辑图片
has_original	Integer	是	音频是否有原唱音轨 0:无 1:有
is_ktv	Integer	是	是否有KTV 0:没有 1:有
bit_rate	Integer	是	比特率
has_pitch	Integer	是	是否有音高 0:没有 1:有

accompany_singer_song

伴奏列表

被如下接口引用：accompany_singer_song

名称	类型	必选	描述
is_hq	Integer	是	是否显示HQ标识 0:不显示 1:显示
has_original	Integer	否	音频是否有原唱音轨 0:无 1:有
singer_name	String	是	歌手名
accompany_id	String	是	伴奏id
song_name	String	是	伴奏名字
duration	Integer	是	伴奏时长单位毫秒
singer_id	String	是	歌手id
album_img	String	是	专辑图片
is_ktv	Integer	是	是否有KTV 0:没有 1:有
bit_rate	Integer	是	比特率
has_pitch	Integer	是	是否有音高 0:没有 1:有

accompany_song_accompany

伴奏

被如下接口引用：song_accompany

名称	类型	描述
accompany_id	String	伴奏id
song_name	String	伴奏名
singer_id	String	歌手id
singer_name	String	歌手名
duration	Integer	伴奏时长单位毫秒
album_img	String	专辑图片
has_original	Integer	音频是否有原唱音轨 0:无 1:有

名称	类型	描述
is_ktv	Integer	是否有KTV 0:没有 1:有
bit_rate	Integer	比特率
has_pitch	Integer	是否有音高 0:没有 1:有

accompany_theme_song

伴奏列表

被如下接口引用：accompany_theme_song

名称	类型	必选	描述
is_hq	Integer	是	是否显示HQ标识 0:不显示 1:显示
has_original	Integer	否	音频是否有原唱音轨 0:无 1:有
singer_name	String	是	歌手名
accompany_id	String	是	伴奏id
song_name	String	是	伴奏名字
duration	Integer	是	伴奏时长单位毫秒
singer_id	String	是	歌手id
album_img	String	是	专辑图片
is_ktv	Integer	是	是否有KTV 0:没有 1:有
bit_rate	Integer	是	比特率
has_pitch	Integer	是	是否有音高 0:没有 1:有

accompany_top_song

伴奏列表

被如下接口引用：accompany_top_song

名称	类型	必选	描述
is_hq	Integer	是	是否显示HQ标识 0:不显示 1:显示
has_original	Integer	否	音频是否有原唱音轨 0:无 1:有

名称	类型	必选	描述
singer_name	String	是	歌手名
accompany_id	String	是	伴奏id
song_name	String	是	伴奏名字
duration	Integer	是	伴奏时长单位毫秒
singer_id	String	是	歌手id
album_img	String	是	专辑图片
is_ktv	Integer	是	是否有KTV 0:没有 1:有
bit_rate	Integer	是	比特率
has_pitch	Integer	是	是否有音高 0:没有 1:有

albums_asset_purchased_albums

专辑信息

被如下接口引用：asset_purchased_albums

名称	类型	描述
album_id	String	专辑id
album_name	String	专辑名
album_translate_name	String	专辑译名
album_img	String	专辑封面
album_img_mini	String	专辑封面100px左右
album_img_small	String	专辑封面300px左右
album_img_medium	String	专辑封面500px左右
intro	String	专辑简介
company	String	唱片公司
singer_id	String	歌手id
singer_name	String	歌手名
publish_time	String	发布时间

albums_singer_album

专辑信息

被如下接口引用：singer_album

名称	类型	描述
album_id	String	专辑id
album_name	String	专辑名
album_translate_name	String	专辑译名
album_img	String	专辑封面
album_img_mini	String	专辑封面100px左右
album_img_small	String	专辑封面300px左右
album_img_medium	String	专辑封面500px左右
intro	String	专辑简介
company	String	唱片公司
singer_id	String	歌手id
singer_name	String	歌手名
publish_time	String	发布时间

appapiCommon

AppApi 公共参数

名称	类型	必选	描述
Action	String	是	公共参数，本接口取值：AppCreateEmailUser
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId。
AppKey	String	是	公共参数，应用 AppKey，用于标识对应的 App。
Signature	String	是	公共参数，请求签名，需用户自行生成，用于校验请求的合法性。
Timestamp	Int64	是	公共参数，请求的 Unix 时间戳（秒级）
Nonce	Int	是	公共参数，随机正整数，与 Timestamp 联合起来，防止重放攻击。

author_newsearch_song

歌手列表

被如下接口引用：accompany_newsearch_song

名称	类型	必选	描述
singer_id	String	是	歌手id
singer_name	String	是	歌手名
singer_img	String	是	歌手图片

categories_accompany_category_list

分类

被如下接口引用：accompany_category_list

名称	类型	必选	描述
category_name	String	是	分类名
category_id	String	是	分类id

categories_mv_category_list

分类列表

被如下接口引用：mv_category_list

名称	类型	必选	描述
category_id	String	是	分类id
category_name	String	是	分类名

data_monitor_delay

数据

被如下接口引用：monitor_delay

名称	类型	必选	描述
api	String	是	api path,必须与api实际path保持一致
userid	String	是	TME用户id

名称	类型	必选	描述
client_ip	String	是	接口使用者ip,如设备直接调用TME api则上报设备ip，反之则厂商服务器ip
server_ip	String	是	接口实际调用时的服务器ip
ecode	Integer	是	业务错误码（error_code）
reqtime	Integer	是	请求时间时间戳单位秒
rspsize	Integer	是	响应包大小，单位字节
sp	String	是	服务提供商对应枚举值分别为 酷狗,酷我,Q音
apn	String	是	网络类型
etype	Integer	是	错误类型 0:调用成功,1:网络错误,2:http错误码,3:业务错误码
reqsize	Integer	是	请求包大小，单位字节
retry	Integer	是	重试次数（本次是第几次重试）
delay	Integer	是	接口调用时延,单位ms

data_monitor_listen

数据

被如下接口引用：monitor_listen

名称	类型	必选	描述
play_quality	String	否	歌曲/长音频播放品质d，取值：SQ、HQ、standard；视频播放品质：FHD、HD、LD、QHD、SD；
duration	Integer	是	歌曲时长（单位ms）
play_time	Integer	是	歌曲实际播放时长（单位ms）
play_type	Integer	是	1.在线 2.缓存 3.下载
client_ip	String	是	接口实际使用者ip,即硬件终端ip
media_type	Integer	否	1-听歌、2.视频MV、3-伴奏K歌业务、4-有声音频、5-直播
lvt	String	是	用户播放时间format: 2019-09-19 00:00:00
try_play	Integer	否	0-可完整播放、1-vip试听类型、2-免登录试听；说明：可完整播放（该用户有完播权限），vip试听类型（非vip用户试听vip歌曲片段），免登录试听（用户未登录，免费试听权限）

名称	类型	必选	描述
sp	String	是	服务提供商对应枚举值分别为 酷狗,酷我,Q音
userid	String	是	TME用户id
song_id	String	是	根据多媒体类型, 赋值相应的ID: 歌曲ID、伴奏ID、MV的ID
api	String	是	api path,必须与api实际path保持一致
source_id	String	是	歌曲id来源, 歌曲来源为歌单上传歌单id, 歌曲来源为榜单, 上传榜单id。如果都不是, 传空

groups_accompany_category_list

伴奏分组

被如下接口引用: accompany_category_list

名称	类型	必选	描述
group_name	String	否	分组名
categories	Array Of categories_accompany_category_list of categories_accompany_category_list	否	分类

groups_accompany_free_top_list

分组列表

被如下接口引用: accompany_free_top_list

名称	类型	必选	描述
group_id	String	是	分组id
group_name	String	是	分组名
tops	Array Of tops_accompany_free_top_list of tops_accompany_free_top_list	否	榜单列表

groups_accompany_top_list

分组列表

被如下接口引用: accompany_top_list

名称	类型	必选	描述
group_id	String	是	分组id
group_name	String	是	分组名
tops	Array Of tops_accompany_top_list of tops_accompany_top_list	否	tops

groups_child_home

资源分组

被如下接口引用：child_home

名称	类型	必选	描述
name	String	否	资源名称
infos	Array Of infos_child_home of infos_child_home	否	资源列表

groups_longaudio_free_home

资源分组

被如下接口引用：longaudio_free_home

名称	类型	必选	描述
name	String	否	资源名称
tag_id	String	否	资源tag_id，值为-1即没有更多的资源
infos	Array Of infos_longaudio_free_home of infos_longaudio_free_home	否	资源列表

groups_playlist_top

分组列表

被如下接口引用：playlist_top

名称	类型	描述
category_id	String	分类id 推荐歌单传入同名字段获取相应歌单

名称	类型	描述
category_name	String	分类名
category_icon	String	分类封面

groups_radio_list

分组列表

被如下接口引用：radio_list

名称	类型	描述
group_id	String	分组id
group_name	String	分组名
radios	Array of radios_radio_list	电台列表

groups_rigion_home

资源分组

被如下接口引用：rigion_home

名称	类型	必选	描述
name	String	否	资源名称
infos	Array Of infos_rigion_home of infos_rigion_home	否	资源列表

groups_rigion_oldman_index

资源分组

被如下接口引用：rigion_oldman_index

名称	类型	必选	描述
id	Integer	否	专区id
name	String	否	专区名称
pic	String	否	专区封面
summary	String	否	描述

groups_top_list

分组

被如下接口引用：top_list

名称	类型	描述
group_id	String	分组id
group_name	String	分组名
tops	Array of tops_top_list	榜单

horizontal_live_room_url

横屏流信息

被如下接口引用：live_room_url

名称	类型	必选	描述
rate	String	否	清晰度标识 (1:流畅 2:标清 3:高清 4:超清 5:1080P)
hls	Array Of String	否	hls流信息
httpflv	Array Of String	否	httpflv流信息
httpsflv	Array Of String	否	httpsflv流信息
rtmp	Array Of String	否	rtmp流信息
httpshls	Array Of String	否	httpshls流信息
streamName	String	否	流名称

infos_child_home

资源列表

被如下接口引用：child_home

名称	类型	必选	描述
resource_type	String	是	大分类 3-歌单 4-专辑 7-小分类
resource_id	String	是	资源id
name	String	是	资源名称
pic	String	是	资源图

infos_longaudio_free_home

资源列表

被如下接口引用：longaudio_free_home

名称	类型	必选	描述
resource_id	Integer	是	资源id
name	String	是	资源名称
pic	String	是	资源封面

infos_rigion_home

资源列表

被如下接口引用：rigion_home

名称	类型	必选	描述
resource_type	String	是	大分类 2-单曲 3-歌单 103-歌单合集 4-专辑 104-专辑合集 5-歌手 6-排行榜 7-分类 8-MV 108-MV合集 9-电台 109-电台合集
resource_id	String	是	资源id
name	String	是	资源名称
pic	String	是	资源封面

letters_accompany_singer_list

letters

被如下接口引用：accompany_singer_list

名称	类型	必选	描述
letter	String	是	歌手首字母
singers	Array Of singers_accompany_singer_list of singers_accompany_singer_list	是	歌手列表

lists_child_resource

资源分组

被如下接口引用：child_resource

名称	类型	必选	描述
id	String	是	资源id
name	String	是	资源名称
pic	String	是	资源图片

lists_longaudio_resource

资源列表

被如下接口引用：longaudio_resource

名称	类型	必选	描述
resource_id	Integer	是	资源id
name	String	是	资源名称
pic	String	是	资源封面

mvInfos_mv_infos

mv信息

被如下接口引用：mv_infos

名称	类型	必选	描述
author_name	String	否	MV制作人名称
tags	tags_mv_infos	否	mv信息
mv_id	String	是	MV id
mv_name	String	是	MV名称
singer_id	String	是	歌手id
singer_name	String	是	歌手名称
accompany_id	String	否	伴奏id
intro	String	否	简介
song_id	String	是	歌曲id
mv_img	String	是	MV图片

mvs_mv_category_mv

MV列表

被如下接口引用：mv_category_mv

名称	类型	必选	描述
mv_img	String	是	MV图片
mv_name	String	是	MV名称
singer_id	String	是	歌手id
singer_name	String	是	歌手名称
song_id	String	是	歌曲id
mv_id	String	是	MV id

mvs_mv_live_tme

MV列表

被如下接口引用：mv_live_tme

名称	类型	必选	描述
mv_id	String	是	MV id
mv_img	String	是	MV图片
mv_name	String	是	MV名称
singer_id	String	是	歌手id
singer_name	String	是	歌手名称
song_id	String	是	歌曲id

mvs_mv_search

mv列表

被如下接口引用：mv_search

名称	类型	必选	描述
singer_name	String	是	歌手名称
has_acc	Integer	是	是否有伴奏 0:没有 1:有

名称	类型	必选	描述
mv_id	String	是	MV id
mv_img	String	是	MV图片
mv_name	String	是	MV名称
song_id	String	是	歌曲id

opuses_accompany_opus_list

opuses

被如下接口引用：accompany_opus_list

名称	类型	必选	描述
opus_name	String	是	作品名称
score	Integer	否	总分数
grade	String	否	等级
listen_num	Integer	是	试听人数
album_img	String	是	专辑封面
accompany_id	String	是	伴奏id
opus_id	String	是	作品id
average_score	Integer	否	平均分
is_private	Integer	是	是否公开，0：公开作品,1：私密作品
duration	Integer	是	作品播放时间,单位秒

playlists_favorite_other_list

歌单

被如下接口引用：favorite_other_list

名称	类型	描述
playlist_id	String	歌单id
playlist_name	String	歌单名称

名称	类型	描述
pic	String	歌单封面
playlist_extra_id	String	歌单附加id

playlists_favorite_self_list

歌单

被如下接口引用：favorite_self_list

名称	类型	描述
playlist_id	String	歌单id
playlist_name	String	歌单名称
pic	String	歌单封面
update_time	String	歌单更新时间
create_time	String	歌单创建时间
total	Integer	歌单歌曲数量

playlists_playlist_awesome

歌单

被如下接口引用：playlist_awesome

名称	类型	描述
playlist_id	String	歌单id
playlist_name	String	歌单名称
pic	String	歌单封面
update_time	String	歌单更新时间
create_time	String	歌单创建时间
intro	string	歌单简介

programs_live_program_list

节目分组

被如下接口引用：live_program_list

名称	类型	必选	描述
room_id	Integer	是	房间id
status	String	是	节目状态 0.未开播;1.开播
price	String	否	价格
name	String	是	节目名称
startTime	String	是	开始时间
endTime	String	是	结束时间
intro	String	是	节目描述
pic	String	是	封面图片
pay_type	Integer	否	是否付费演出会 0.不是;1.是

radios_radio_list

电台列表

被如下接口引用：radio_list

名称	类型	描述
radio_id	String	电台id
radio_name	String	电台名
radio_img	String	电台封面

recommends_accompany_category_list

推荐分类

被如下接口引用：accompany_category_list

名称	类型	必选	描述
category_id	String	是	分类id
category_name	String	是	分类名
category_img	String	是	分类图片

singers_accompany_singer_list

热门歌手

被如下接口引用： accompany_singer_list

名称	类型	必选	描述
singer_id	String	是	歌手id
singer_name	String	是	歌手名
singer_img	String	是	歌手图片

singers_accompany_singer_list_1

歌手列表

名称	类型	必选	描述
singer_name	String	是	歌手名
singer_img	String	是	歌手图片
singer_id	String	是	歌手id

singers_singer_list

歌手

被如下接口引用： singer_list

名称	类型	描述
singer_id	String	歌手id
singer_name	String	歌手名
area	String	歌手地区
translate_name	String	歌手译名
singer_img	String	歌手相片

slots_search_voice

语音搜索列表

被如下接口引用： search_voice

名称	类型	必选	描述
name	String	是	
value	String	是	词槽参数值

songs_album_info

歌曲列表

被如下接口引用：album_info

名称	类型	描述
singer_img	String	歌手头像
album_img	String	专辑封面
album_img_mini	String	专辑封面100px左右
album_img_small	String	专辑封面300px左右
album_img_medium	String	专辑封面500px左右
is_vip_song	Integer	是否vip歌曲。0：否 1.是
mv_id	String	mv id
has_accompany	Integer	是否有伴奏。0：否 1.是
song_id	String	歌曲id
song_name	String	歌曲名
singer_id	String	歌手id
singer_name	String	歌手名
album_id	String	专辑id
album_name	String	专辑名

名称	类型	描述
playable_code	Integer	是否可以播放 枚举值依次对应: 0:可以 1:海外地区不能播放 2:歌曲无版权不能播放 3:会员歌曲, 非会员不能播放 4:付费内容, 须购买才可播放 5:牛方案策略, 非会员不能播放 6:因定向版权下架不能播放 (针对 APP 有权但设备端无权的情况) 9:未知原因,无权播放
try_playable	Integer	是否有试听片段 0:否 1:有
language	String	语言

songs_asset_purchased_songs

歌曲列表

被如下接口引用: asset_purchased_songs

名称	类型	描述
singer_img	String	歌手头像
album_img	String	专辑封面
album_img_mini	String	专辑封面100px左右
album_img_small	String	专辑封面300px左右
album_img_medium	String	专辑封面500px左右
is_vip_song	Integer	是否vip歌曲。0: 否 1.是
mv_id	String	mv id
has_accompany	Integer	是否有伴奏。0: 否 1.是
song_id	String	歌曲id
song_name	String	歌曲名
singer_id	String	歌手id
singer_name	String	歌手名
album_id	String	专辑id

名称	类型	描述
album_name	String	专辑名
playable_code	Integer	是否可以播放 枚举值依次对应: 0:可以 1:海外地区不能播放 2:歌曲无版权不能播放 3:会员歌曲, 非会员不能播放 4:付费内容, 须购买才可播放 5:牛方案策略, 非会员不能播放 6:因定向版权下架不能播放 (针对 APP 有权但设备端无权的情况) 9:未知原因,无权播放
try_playable	Integer	是否有试听片段 0:否 1:有
language	String	语言

songs_awesome_everyday

歌曲列表

被如下接口引用: awesome_everyday

名称	类型	描述
singer_img	String	歌手头像
album_img	String	专辑封面
album_img_mini	String	专辑封面100px左右
album_img_small	String	专辑封面300px左右
album_img_medium	String	专辑封面500px左右
is_vip_song	Integer	是否vip歌曲。0: 否 1.是
mv_id	String	mv id
has_accompany	Integer	是否有伴奏。0: 否 1.是
song_id	String	歌曲id
song_name	String	歌曲名
singer_id	String	歌手id
singer_name	String	歌手名

名称	类型	描述
album_id	String	专辑id
album_name	String	专辑名
playable_code	Integer	是否可以播放 枚举值依次对应: 0:可以 1:海外地区不能播放 2:歌曲无版权不能播放 3:会员歌曲, 非会员不能播放 4:付费内容, 须购买才可播放 5:牛方案策略, 非会员不能播放 6:因定向版权下架不能播放 (针对 APP 有权但设备端无权的情况) 9:未知原因,无权播放
try_playable	Integer	是否有试听片段 0:否 1:有
language	String	语言

songs_awesome_newsong

歌曲列表

被如下接口引用：awesome_newsong

名称	类型	描述
singer_img	String	歌手头像
album_img	String	专辑封面
album_img_mini	String	专辑封面100px左右
album_img_small	String	专辑封面300px左右
album_img_medium	String	专辑封面500px左右
is_vip_song	Integer	是否vip歌曲。0：否 1.是
mv_id	String	mv id
has_accompany	Integer	是否有伴奏。0：否 1.是
song_id	String	歌曲id
song_name	String	歌曲名
singer_id	String	歌手id

名称	类型	描述
singer_name	String	歌手名
album_id	String	专辑id
album_name	String	专辑名
playable_code	Integer	是否可以播放 枚举值依次对应: 0:可以 1:海外地区不能播放 2:歌曲无版权不能播放 3:会员歌曲, 非会员不能播放 4:付费内容, 须购买才可播放 5:牛方案策略, 非会员不能播放 6:因定向版权下架不能播放 (针对 APP 有权但设备端无权的情况) 9:未知原因,无权播放
try_playable	Integer	是否有试听片段 0:否 1:有
language	String	语言

songs_awesome_recommend

歌曲列表

被如下接口引用: awesome_recommend

名称	类型	描述
singer_img	String	歌手头像
album_img	String	专辑封面
album_img_mini	String	专辑封面100px左右
album_img_small	String	专辑封面300px左右
album_img_medium	String	专辑封面500px左右
is_vip_song	Integer	是否vip歌曲。0: 否 1.是
mv_id	String	mv id
has_accompany	Integer	是否有伴奏。0: 否 1.是
song_id	String	歌曲id
song_name	String	歌曲名

名称	类型	描述
singer_id	String	歌手id
singer_name	String	歌手名
album_id	String	专辑id
album_name	String	专辑名
playable_code	Integer	是否可以播放 枚举值依次对应: 0:可以 1:海外地区不能播放 2:歌曲无版权不能播放 3:会员歌曲，非会员不能播放 4:付费内容，须购买才可播放 5:牛方案策略，非会员不能播放 6:因定向版权下架不能播放（针对 APP 有权但设备端无权的情况） 9:未知原因,无权播放
try_playable	Integer	是否有试听片段 0:否 1:有
language	String	语言

songs_favorite_song

歌曲列表

被如下接口引用：favorite_song

名称	类型	描述
song_id	String	歌曲id
song_name	String	歌曲名
singer_id	String	歌手id
singer_name	String	歌手名
album_id	String	专辑id
album_name	String	专辑名
song_extra_id	String	歌曲附加id,仅获取自建歌单歌曲列表时返回（取消收藏时使用）

名称	类型	描述
playable_code	Integer	是否可以播放 枚举值依次对应: 0:可以 1:海外地区不能播放 2:歌曲无版权不能播放 3:会员歌曲, 非会员不能播放 4:付费内容, 须购买才可播放 5:牛方案策略, 非会员不能播放 6:因定向版权下架不能播放 (针对 APP 有权但设备端无权的情况) 9:未知原因,无权播放
is_vip_song	Integer	是否vip歌曲。0: 否 1.是

songs_id_song_infos

歌曲id列表

被如下接口引用: song_infos

名称	类型	必选	描述
Values	Array of String	是	一个或者多个过滤值

songs_playlist_song

歌曲列表

被如下接口引用: playlist_song

名称	类型	描述
playable_code	Integer	是否可以播放 枚举值依次对应: 0:可以 1:海外地区不能播放 2:歌曲无版权不能播放 3:会员歌曲, 非会员不能播放 4:付费内容, 须购买才可播放 5:牛方案策略, 非会员不能播放 6:因定向版权下架不能播放 (针对 APP 有权但设备端无权的情况) 9:未知原因,无权播放
song_id	String	歌曲id
song_name	String	歌曲名
singer_id	String	歌手id
singer_name	String	歌手名

名称	类型	描述
album_id	String	专辑id
album_name	String	专辑名
is_vip_song	Integer	是否vip歌曲。0：否 1.是

songs_radio_song

歌曲列表

被如下接口引用：radio_song

名称	类型	描述
singer_img	String	歌手头像
album_img	String	专辑封面
album_img_mini	String	专辑封面100px左右
album_img_small	String	专辑封面300px左右
album_img_medium	String	专辑封面500px左右
is_vip_song	Integer	是否vip歌曲。0：否 1.是
mv_id	String	mv id
has_accompany	Integer	是否有伴奏。0：否 1.是
song_id	String	歌曲id
song_name	String	歌曲名
singer_id	String	歌手id
singer_name	String	歌手名
album_id	String	专辑id
album_name	String	专辑名

名称	类型	描述
playable_code	Integer	是否可以播放 枚举值依次对应: 0:可以 1:海外地区不能播放 2:歌曲无版权不能播放 3:会员歌曲, 非会员不能播放 4:付费内容, 须购买才可播放 5:牛方案策略, 非会员不能播放 6:因定向版权下架不能播放 (针对 APP 有权但设备端无权的情况) 9:未知原因,无权播放
language	String	语言

songs_search_song

歌曲列表

被如下接口引用: search_song

名称	类型	必选	描述
singer_name	String	是	歌手名
album_id	String	是	歌曲所属专辑id
album_name	String	是	歌曲所属专辑名
song_id	String	是	歌曲id
song_name	String	是	歌曲名
singer_id	String	是	歌手id
playable_code	Integer	是	是否可以播放 枚举值依次对应: 0:可以 1:海外地区不能播放 2:歌曲无版权不能播放 3:会员歌曲, 非会员不能播放 4:付费内容, 须购买才可播放 5:牛方案策略, 非会员不能播放 6:因定向版权下架不能播放 (针对 APP 有权但设备端无权的情况) 9:未知原因,无权播放
has_accompany	Integer	是	是否有伴奏。0: 否 1.是
singer_img	String	是	歌手头像
album_img_mini	String	是	专辑封面100px左右

名称	类型	必选	描述
language	String	否	语言
is_vip_song	Integer	是	是否vip歌曲。0: 否 1.是
album_img	String	是	专辑封面
album_img_small	String	是	专辑封面300px左右
album_img_medium	String	是	专辑封面500px左右
mv_id	String	是	mv id

songs_search_voice

语音搜索歌曲列表

被如下接口引用：search_voice

名称	类型	必选	描述
singer_name	String	否	歌手名
duration	Integer	是	歌曲时长,单位ms
song_id	String	是	歌曲id
song_name	String	是	歌曲名

songs_singer_song

歌曲列表

被如下接口引用：singer_song

名称	类型	描述
singer_img	String	歌手头像
album_img	String	专辑封面
album_img_mini	String	专辑封面100px左右
album_img_small	String	专辑封面300px左右
album_img_medium	String	专辑封面500px左右
is_vip_song	Integer	是否vip歌曲。0: 否 1.是

名称	类型	描述
mv_id	String	mv id
has_accompany	Integer	是否有伴奏。0：否 1.是
song_id	String	歌曲id
song_name	String	歌曲名
singer_id	String	歌手id
singer_name	String	歌手名
album_id	String	专辑id
album_name	String	专辑名
playable_code	Integer	是否可以播放 枚举值依次对应： 0:可以 1:海外地区不能播放 2:歌曲无版权不能播放 3:会员歌曲，非会员不能播放 4:付费内容，须购买才可播放 5:牛方案策略，非会员不能播放 6:因定向版权下架不能播放（针对 APP 有权但设备端无权的情况） 9:未知原因,无权播放
try_playable	Integer	是否有试听片段 0:否 1:有
language	String	语言

songs_song_infos

歌曲列表

被如下接口引用：song_infos

名称	类型	描述
singer_img	String	歌手头像
album_img	String	专辑封面
album_img_mini	String	专辑封面100px左右
album_img_small	String	专辑封面300px左右
album_img_medium	String	专辑封面500px左右

名称	类型	描述
is_vip_song	Integer	是否vip歌曲。0：否 1.是
mv_id	String	mv id
has_accompany	Integer	是否有伴奏。0：否 1.是
song_id	String	歌曲id
song_name	String	歌曲名
singer_id	String	歌手id
singer_name	String	歌手名
album_id	String	专辑id
album_name	String	专辑名
playable_code	Integer	是否可以播放 枚举值依次对应： 0:可以 1:海外地区不能播放 2:歌曲无版权不能播放 3:会员歌曲，非会员不能播放 4:付费内容，须购买才可播放 5:牛方案策略，非会员不能播放 6:因定向版权下架不能播放（针对 APP 有权但设备端无权的情况） 9:未知原因,无权播放
try_playable	Integer	是否有试听片段 0:否 1:有
topic_url	String	歌曲专题页面(购买链接)
language	String	语言

songs_song_tolisten

歌曲列表

被如下接口引用：songs_song_tolisten

名称	类型	描述
singer_img	String	歌手头像
album_img	String	专辑封面
album_img_mini	String	专辑封面100px左右

名称	类型	描述
album_img_small	String	专辑封面300px左右
album_img_medium	String	专辑封面500px左右
is_vip_song	Integer	是否vip歌曲。0：否 1.是
mv_id	String	mv id
has_accompany	Integer	是否有伴奏。0：否 1.是
song_id	String	歌曲id
song_name	String	歌曲名
singer_id	String	歌手id
singer_name	String	歌手名
album_id	String	专辑id
album_name	String	专辑名
playable_code	Integer	是否可以播放 枚举值依次对应： 0:可以 1:海外地区不能播放 2:歌曲无版权不能播放 3:会员歌曲，非会员不能播放 4:付费内容，须购买才可播放 5:牛方案策略，非会员不能播放 6:因定向版权下架不能播放（针对 APP 有权但设备端无权的情况） 9:未知原因,无权播放
try_playable	Integer	是否有试听片段 0:否 1:有
topic_url	String	歌曲专题页面(购买链接)
language	String	语言

songs_top_song

歌曲列表

被如下接口引用：top_song

名称	类型	描述
singer_name	String	歌手名

名称	类型	描述
album_id	String	歌曲所属专辑id
album_name	String	歌曲所属专辑名
song_id	String	歌曲id
song_name	String	歌曲名
singer_id	String	歌手id
playable_code	Integer	是否可以播放 枚举值依次对应:
is_vip_song	Integer	是否vip歌曲。0: 否 1.是
mv_id	String	mv id
has_accompany	Integer	是否有伴奏。0: 否 1.是
singer_img	String	歌手头像
album_img_mini	String	专辑封面100px左右
album_img_small	String	专辑封面300px左右
language	String	语言
album_img	String	专辑封面
album_img_medium	String	专辑封面500px左右

tags_mv_info

mv信息

被如下接口引用：mv_info

名称	类型	必选	描述
tag_id	String	否	标签id
parent_id	String	否	标签parentId
tag_name	String	否	标签的名称

tags_mv_infos

mv信息

被如下接口引用：mv_infos

名称	类型	必选	描述
tag_id	String	否	标签id
parent_id	String	否	标签parentId
tag_name	String	否	标签的名称

themes_accompany_theme_list

主题列表

被如下接口引用：accompany_theme_list

名称	类型	必选	描述
theme_thumbnail	String	是	主题缩略图
theme_id	String	是	主题id
theme_name	String	是	主题名
theme_img	String	是	主题图片

tokenapiCommon

tokenApi的公共参数

名称	类型	必选	描述
Action	String	是	公共参数，本接口取值：AppLogoutUser
AccessToken	String	是	公共参数，AccessToken 用于对一个已经登录的用户鉴权
RequestId	String	是	公共参数，唯一请求ID，可自行生成，推荐使用 uuid。定位问题时，需提供该次请求的 RequestId

tops_accompany_free_top_list

榜单列表

被如下接口引用：accompany_free_top_list

名称	类型	必选	描述
top_id	String	是	榜单id
top_name	String	是	榜单名

名称	类型	必选	描述
header_url	String	是	榜单图片

tops_accompany_top_list

tops

被如下接口引用：accompany_top_list

名称	类型	必选	描述
top_id	String	是	榜单id
top_name	String	是	榜单名

tops_top_list

榜单

被如下接口引用：top_list

名称	类型	描述
top_id	String	榜单id
top_name	String	榜单名
header_url	String	榜单图片

vertical_live_room_url

竖屏流信息

被如下接口引用：live_room_url

名称	类型	必选	描述
rtmp	Array Of String	否	rtmp流信息
httpshls	Array Of String	否	httpshls流信息
streamName	String	否	流名称
rate	String	否	清晰度标识 (1:流畅 2:标清 3:高清 4:超清 5:1080P)
hls	Array Of String	否	hls流信息
httpflv	Array Of String	否	httpflv流信息

名称	类型	必选	描述
httpsflv	Array Of String	否	httpsflv流信息

小程序插件使用指南

最近更新时间：2020-12-10 08:56:36

功能概述

目前平台针对蓝牙测温仪设备和物联网实时音视频服务的相关功能提供了腾讯连连小程序插件，方便开发者使用插件进行小程序开发。

蓝牙测温仪设备

小程序插件提供了蓝牙测温仪设备的发现、连接、断开连接、状态获取、数据获取等功能。

- **发现蓝牙体温计：**本插件提供小程序发现蓝牙体温计设备的接口，第三方开发者无需熟悉蓝牙设备细节协议以及微信小程序蓝牙开发，即可发现设备。
- **连接蓝牙体温计：**本插件提供连接某个蓝牙体温计的接口，其他小程序通过发现接口返回数据，进行连接调用。
- **断开蓝牙体温计：**本插件提供主动断开蓝牙体温计设备连接接口。
- **小程序与蓝牙设备状态查询或通知：**本插件提供相关接口可给上层应用返回当前连接状态。
- **获取蓝牙当前检测体温数据接口：**本插件提供接口将每次检测获取的蓝牙体温数据返回给上层应用。
- **获取体温枪扫码数据：**本插件提供接口获取体温枪设备扫描二维码获取自定义信息。
- **支持离线数据上报：**本插件提供接口获取体温枪离线场景下上报的体温和二维码信息数据。

物联网实时音视频服务

实时音视频呼叫：本插件提供小程序实时音视频的接口，帮助开发者快速实现与设备端低成本、低延时、高品质的音视频通话。

使用场景

蓝牙测温仪设备

支持小程序开发的蓝牙体温计设备可以广泛应用于家庭、企业、社区、园区等不同场景。

- **蓝牙体温枪设备应用于家庭场景可开发实现**
体温实时记录上报、成员信息即时管理、异常体温数据告警提醒等功能。
- **蓝牙体温枪设备应用于企业/社区/园区等场景可开发实现**
除可开发家庭版本全部功能外，还可以开发支持多级权限管理、社区/园区/办公大楼的出入口管理、全体成员大盘检测数据的统计分析等。

物联网实时音视频服务

适用于小程序与可穿戴设备、智能家居设备视音频通话的场景。

- **儿童智能手表**
实现家长端对儿童手表端的主动音视频呼叫，配合小程序 SDK 可实现远程控制指令下发，手表数据上传等功能。

- 视频门禁楼宇对讲

实现小程序主动呼叫视频门禁等物联网设备，配合小程序 SDK 可满足远程门禁管理、报警与联动等功能。

使用指南

基本使用步骤如下：

1. 小程序后台添加插件，详情请参见 [官方文档](#)。
2. 引入插件代码。

```
{
  "plugins": {
    "iotexplorer-weapp-sdk-plugin": {
      "version": "1.0.0", // 具体需要使用的版本号
      "provider": "wx711dd9e4296e7f6" // 连连插件的appid
    }
  }
}
```

3. 实现 demo 的详细操作，请参见 [demo](#)。

插件的名词解释和接口使用说明，请参见 [腾讯连连小程序插件使用说明](#)。