RT-THREAD ALI-IOTKIT 用户手册

RT-THREAD 文档中心

WWW.RT-THREAD.ORG

Friday 28th September, 2018

版本和修订

Date	Version	Author	Note
2018-07-21	v0.1	MurphyZhao	初始版本

目录

版本和修订 **i**

目录 **ii**

1 软件包介绍 **1**

1.1 软件框架图 2

1.2 软件包目录结构 2

1.2.1 iotkit-embedded 软件包目录结构 3

1.3 功能特点 4

2 示例程序 **6**

2.1 LinkDevelop 平台 6

2.1.1 准备工作 6

2.1.2 MQTT 示例 11

2.1.3 OTA 示例 15

2.2 LinkPlatform 平台 20

2.2.1 准备工作 20

2.2.2 MQTT 示例 24

2.2.3 OTA 示例 27

2.3 注意事项 30

2.4 常见问题 31

3 工作原理 **32**

3.1 MQTT 数据交互流程 33

3.2 OTA 数据交互流程 33

4 使用指南 **35**

4.1 MQTT 连接 . . . 35

4.1.1 特征 . . . 35

4.1.2 安全等级 . . . 35

4.1.3 连接域名 . . . 36

4.1.4 Topic 规范 . . . 36

4.1.5 建立 MQTT 连接 . . . 36

4.1.6 订阅 Topic 主题 . . . 37

4.1.7 发布消息 . . . 37

4.1.8 取消订阅 . . . 38

4.1.9 下行数据接收 . . . 38

4.1.10 销毁 MQTT 连接 . . . 38

4.1.11 检查连接状态 . . . 38

4.1.12 MQTT 保持连接 . . . 38

4.2 CoAP 连接 . . . 39

4.2.1 CoAP 约定 . . . 39

4.2.2 应用场景 . . . 39

4.2.3 建立连接 . . . 40

4.2.4 收发数据 . . . 40

4.2.5 下行数据接收 . . . 41

4.2.6 销毁 CoAP 连接 . . . 41

4.3 OTA 升级 . . . 41

4.3.1 固件升级 Topic . . . 41

4.3.2 固件升级说明 . . . 42

4.3.3 OTA 代码说明 . . . 42

4.4 参考 . . . 44

5 API 说明 **45**

5.1 必选 API . . . 45

5.2 MQTT 功能相关 API . . . 45

5.3 CoAP 功能相关 API . . . 46

5.4 HTTP 功能相关 API 47
5.5 OTA 功能相关 API 48
5.6 云端连接 Cloud Connection 功能相关 API 49
5.7 CMP 功能相关 API 49
5.8 设备影子相关 (可选功能) API 50
5.9 主子设备相关 (可选功能) API 50
5.10 linkkit 功能相关 API 51

第 1 章

软件包介绍

ali-iotkit 是 RT-Thread 移植的用于连接阿里云 IoT 平台的软件包。基础 SDK 是阿里提供的 iotkit-embedded C-SDK。

物联网套件提供了如下的能力:

- 嵌入式设备快速接入 (设备端 SDK)
- 设备管理
- 设备和数据信息安全
- 桥接到阿里云其他产品, 对设备数据存储/计算

图 1.1: *阿里云物联网平台场景架构图*

在物联网平台场景架构图中，左边物联网设备端 SDK 就是将嵌入式设备连接到阿里云的部分。

1.1 软件框架图

iotkit SDK 为了方便设备上云封装了丰富的连接协议，如 MQTT、CoAP、HTTP、TLS，并且对硬件平台进行了抽象，使其不受具体的硬件平台限制而更加灵活。在代码架构方面，iotkit SDK 分为三层，如下图所示：

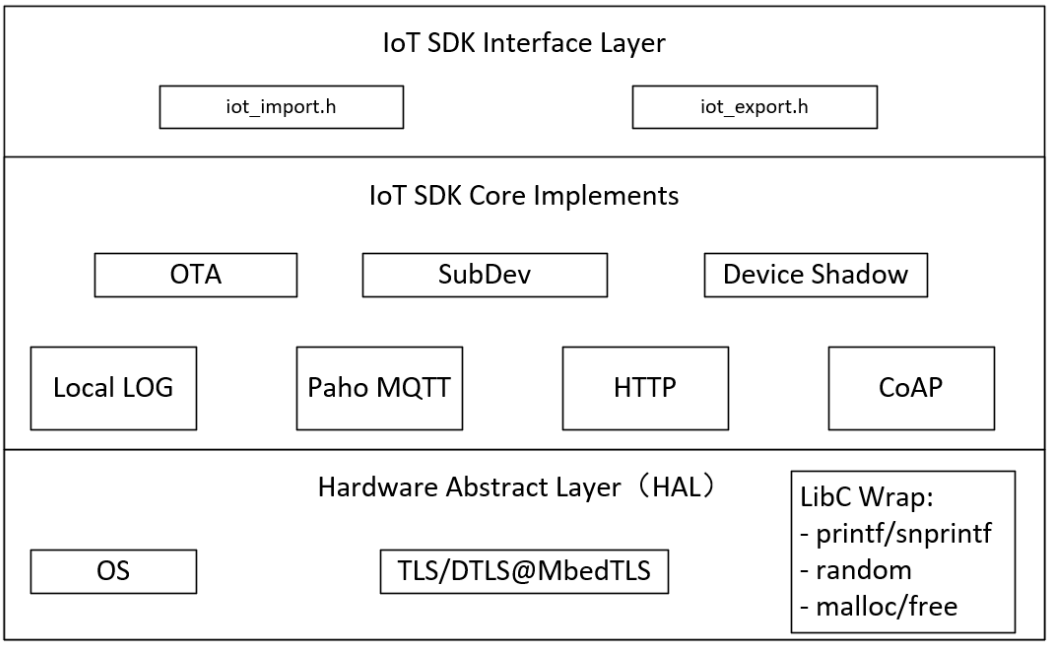

图 1.2: *iotkit SDK 软件框架图*

- 最底层称为硬件平台抽象层, 也简称 HAL 层（Hardware Abstract Layer）

  抽象不同的嵌入式目标板上，操作系统对 SDK 的支撑函数，包括网络收发、TLS/DTLS 通道建立和读写, 内存申请是否和互斥量加锁解锁等。

- 中间层称为 SDK 内核实现层（IoT SDK Core Implements）

  物联网平台 C-SDK 的核心实现部分, 它基于 HAL 层接口完成了 MQTT/CoAP 通道等的功能封装，包括 MQTT 的连接建立、报文收发、CoAP 的连接建立、报文收发、OTA 的固件状态查询和 OTA 的固件下载等。

  中间层的封装，使得用户无需关心内部实现逻辑，可以不经修改地应用。

- 最上层称为 SDK 接口声明层（IoT SDK Interface Layer）

  最上层是为应用提供 API 的，用户使用该层的 API 完成具体的业务逻辑。

1.2 软件包目录结构

ports 目录是 RT-Thread 移植 iotkit-embedded 软件包时所涉及到的移植文件，使用 scons 进行重新构建。

iotkit-embedded 软件包是阿里物联网平台 C-SDK 源码，包含连接阿里云所必须的软件包。

```
ali-iotkit
|   README.md                       // 软件包使用说明
|   SConscript                      // RT-Thread 默认的构建脚本
+---docs
|   +---figures                     // 文档使用图片
|   |   api.md                      // API 使用说明
|   |   introduction.md             // 软件包详细介绍
|   |   LICENSE                     // 许可证文件
|   |   principle.md                // 实现原理
|   |   README.md                   // 文档结构说明
|   |   samples.md                  // 软件包示例
|   |   user-guide.md               // 使用说明
|   +---version.md                  // 版本说明
+---ports                           // 移植文件
|   +---rtthread                    // OS 相关移植文件
|   +---ssl                         // MbedTLS 相关的移植文件
+---samples
|   +---mqtt                        // MQTT 通道接入阿里云的示例程序
|   +---ota                         // 阿里云 OTA 功能演示例程
+---iotkit-embedded                 // iotkit 源码
```

1.2.1 iotkit-embedded 软件包目录结构

iotkit-embedded 软件包是阿里物联网平台 C-SDK 源码，未经修改，包含里了连接阿里云 IoT 所必须的软件包。RT-Thread 移植后，没有使用 iotkit-embedded 中默认的 Makefile 构建脚本，而是使用 scons 重新进行的构建。

iotkit-embedded 软件包目录结构如下所示：

```
+-- LICENSE            : 软件许可证，Apache-2.0 版本软件许可证
+-- make.settings      : 功能裁剪配置，如 MQTT|CoAP，或裁剪如 OTA|Shadow
+-- README.md          : 快速开始导引
+-- sample             : 例程目录，演示通信模块和服务模块的使用
|   +-- mqtt           :     演示如何使用通信模块 MQTT 的 API
|   +-- coap           :     演示如何使用通信模块 CoAP 的 API
|   +-- device-shadow  :     演示如何使用服务模块 DeviceShadow 的 API
|   +-- http           :     演示如何使用通信模块 HTTP 的 API
|   +-- ota            :     演示如何使用服务模块 OTA 的 API
+-- src
```

```
+-- sdk-impl      : SDK 的接口层，提供总体的头文件，和一些 API 的接口
    封装
+-- sdk-tests     : SDK 的单元测试
+-- mqtt          : 通信模块，实现以 MQTT 协议接入
+-- coap          : 通信模块，实现以 CoAP 协议接入
+-- http          : 通信模块，实现以 HTTP 协议接入
+-- ota           : 服务模块，实现基于 MQTT|CoAP+HTTP+TLS 的固件下载
    通道
+-- shadow        : 服务模块，实现设备影子
+-- platform      : 硬件平台抽象层，需要移植适配
+-- import        : 外部输入目录，存放芯片/模组厂商提供的头文件/二进
    制库
+-- configs       : 硬件平台编译配置，如交叉编译工具链设置，功能模块
    裁剪等
+-- scripts       : 编译过程将要外部引用的脚本，用户不必关注
+-- packages      : SDK 引用的外部软件模块，用户不必关注
+-- log           : 基础模块，实现运行日志
+-- system        : 基础模块，保存全局信息，如 TLS 根证书，设备标识
    ID 等
+-- tls           : 基础模块，实现 TLS/DTLS，来自裁剪过的开源软件
    mbedtls
+-- utils         : 基础模块，实现工具函数，如 SHA1 摘要计算、NTP 对
    时等
```

1.3 功能特点

- 不同网络接入

  提供不同网络的设备接入方案，例如 2/3/4G、NB-IoT、LoRa 等，解决企业异构网络设备接入管理的痛点。

- 不同协议接入

  提供多种协议的设备 SDK，例如 MQTT、CoAP、HTTP 等，这样既能满足设备需要长连接保证实时性的需求，也能满足设备需要短连接降低功耗的需求。

- 双向通信

  提供设备与云端的上下行通道，能够稳定可靠的支撑设备上报与指令下发设备的场景。

- 设备影子

  提供设备影子缓存机制，将设备与应用解耦，解决在无线网络不稳定情况下的通信不可靠痛点。

- 设备认证

  提供一机一密的设备认证机制，降低设备被攻破的安全风险。

- 安全传输

  提供 TLS 标准的数据传输通道，保证数据的机密性和完整性。

第 2 章

示例程序

ali-iotkit 软件包同时支持阿里现有的 **LinkDevelop** 和 **LinkPlatform** 平台。

本文针对这两个平台分别进行示例程序的演示，用户可以根据自己的需求选择使用其中的一个。

2.1 LinkDevelop 平台

LinkDevelop 平台以 RGB_LED 为例，介绍设备与云端如何进行双向通讯。

2.1.1 准备工作

- 注册 LinkDevelop 平台

图 2.1: *注册 LinkDevelop 平台*

- 新建项目

图 2.2: *新建项目*

- 新增产品

  新增产品的时候，根据需要选择数据格式，这里使用 **Alink** 数据格式演示，并选择 WiFi 通信方式。

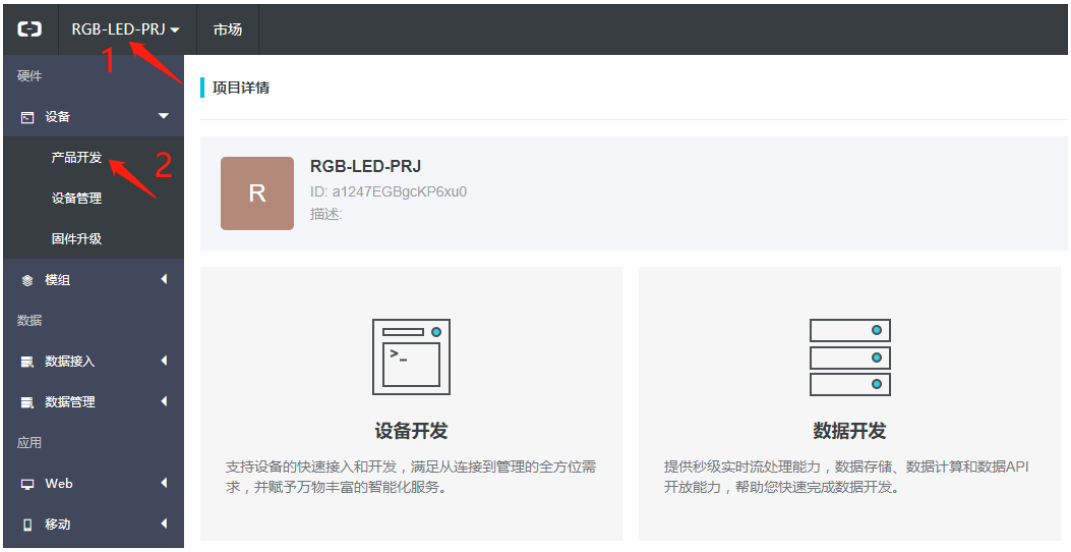

图 2.3: *打开产品开发页面*

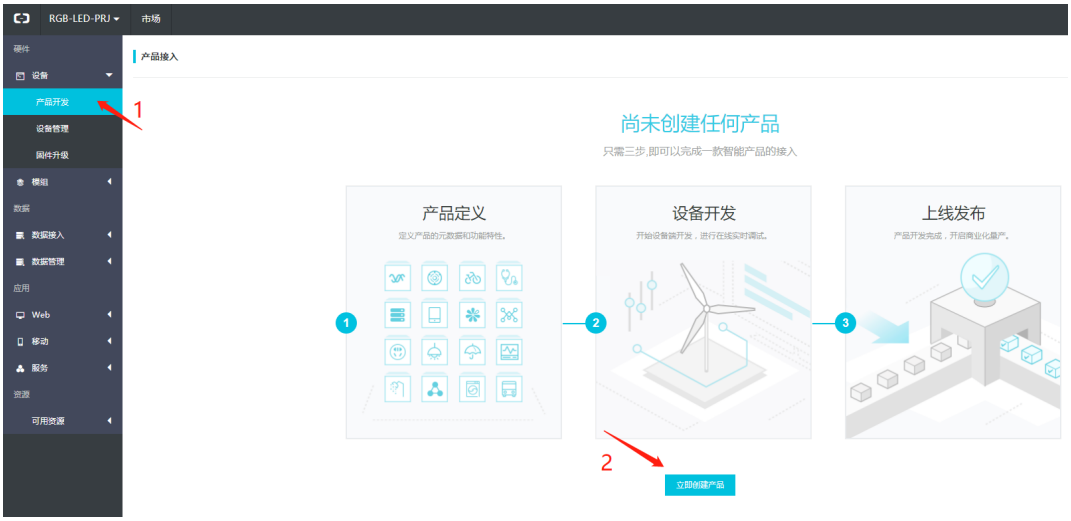

图 2.4: *创建产品*

图 2.5: *填写产品信息*

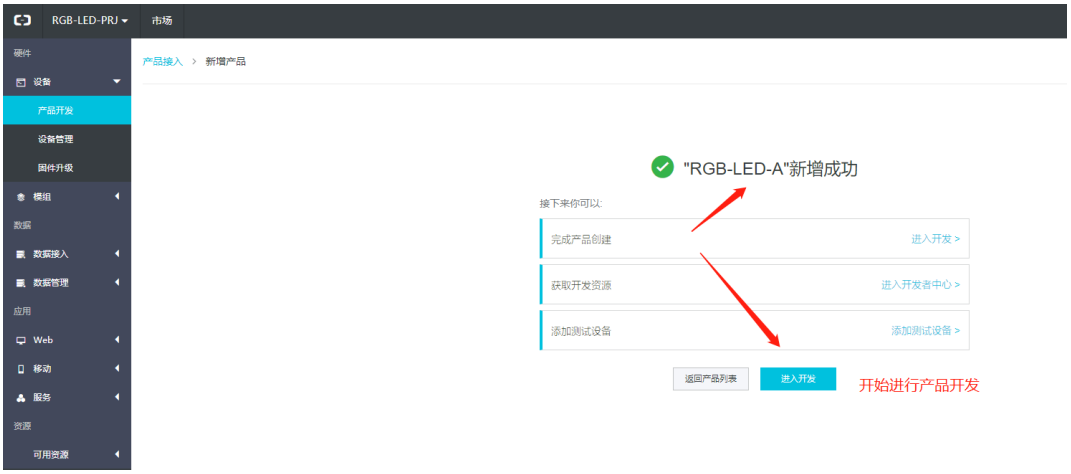

图 2.6: *进入产品开发*

- 增加功能

  为 RGB LED 演示产品添加 RGB 调色功能，如下图所示：

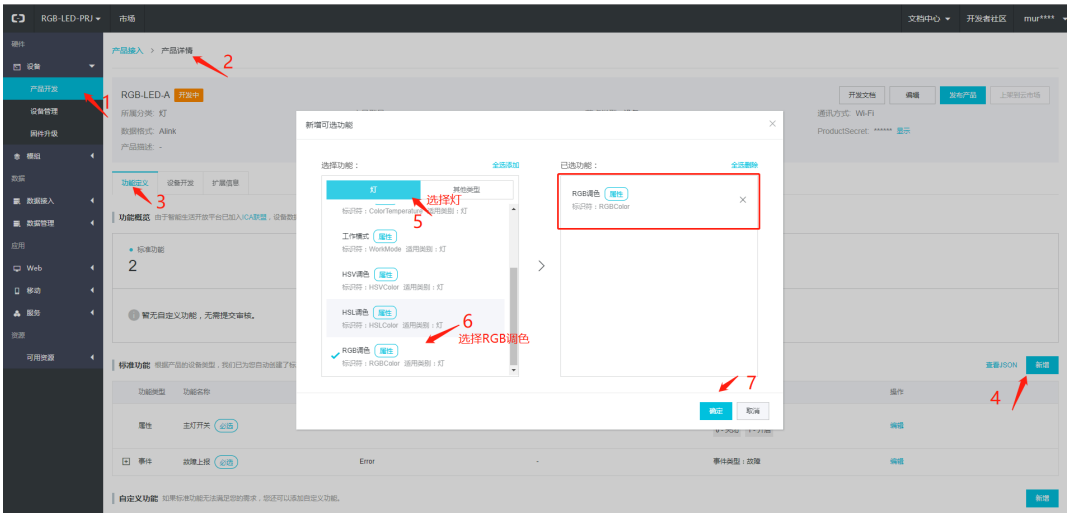

图 2.7: *增加产品功能*

- 添加设备

  创建产品后，点击查看进入产品详情页面，点击设备开发，新增一个调试设备。

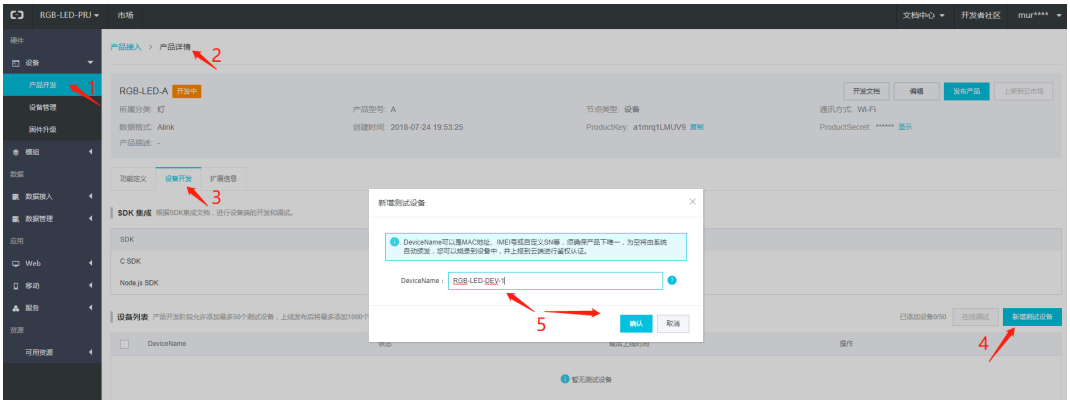

图 2.8: *添加设备*

成功创建设备后，可以获取到设备激活需要的三元组（**ProductKey**、**DeviceName**、**DeviceSecret**），后面需要使用 **menuconfig** 配置到设备 SDK 中。

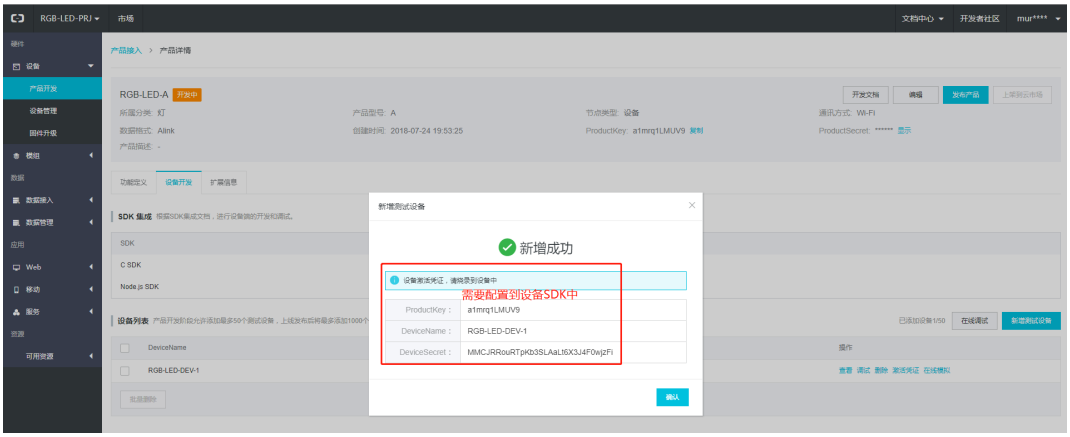

图 2.9: *获取设备激活凭证*

- **获取软件包**

  打开 RT-Thread 提供的 ENV 工具，使用 **menuconfig** 配置软件包。

  – 配置 iotkit 软件包

    配置使能 iotkit 软件包并**填写设备激活凭证**。

    `menuconfig` 中选择阿里云平台为 **LinkDevelop**，`OTA channel` 选择 **MQTT**（以 MQTT 为例），详细的配置如下所示：

```
RT-Thread online packages  --->
    IoT - internet of things  --->
        IoT Cloud  --->
          [*] Ali-iotkit:  Ali Cloud SDK for IoT platform  --->
                Select Aliyun platform (LinkDevelop Platform)  --->
          (a1dSQSGZ77X) Config Product Key
```

```
          (RGB-LED-DEV-1) Config Device Name
          (Ghuiyd9nmGowdZzjPqFtxhm3WUHEbIlI) Config Device Secret
          -*-   Enable MQTT
          [*]     Enable MQTT sample
          [*]     Enable MQTT direct connect
          [*]     Enable SSL
          [ ]   Enable COAP
          [*]   Enable OTA
                      Select OTA channel (Use MQTT OTA channel)
                          --->
                    version (latest)  --->
```

– 增加 mbedTLS 帧大小

阿里 TLS 认证过程中数据包较大，这里需要增加 TLS 帧大小，OTA 的时候**至少需要 8K** 大小。

打开 RT-Thread 提供的 ENV 工具，使用 **menuconfig** 配置 **TLS** 帧大小。

```
RT-Thread online packages  --->
    security packages  --->
      -*- mbedtls:An open source, portable, easy to use,
                  readable and flexible SSL library  --->
      (8192) Maxium fragment length in bytes
```

– 使用 pkgs --update 命令下载软件包

2.1.2 MQTT 示例

该 MQTT 示例程序以 RGB-LED 为例，演示了如何在设备上使用 MQTT + TLS/SSL 通道与阿里云平台建立双向通信。

示例文件

示例程序路径	验证平台	说明
samples/mqtt/ mqtt-example.c	LinkDevelop, LinkPlatform	基于 MQTT 通道的设备和云双向通信例程

命令列表

例程中，使用 MSH 命令启动 MQTT 例程，命令如下所示：

命令	说明
ali_mqtt_test start	启动 MQTT 示例
ali_mqtt_test pub open	开灯，并向云端同步开灯状态
ali_mqtt_test pub close	关灯，并向云端同步关灯状态
ali_mqtt_test stop	停止 MQTT 示例

启动 MQTT

使用 **ali_mqtt_test start** 命令启动 MQTT 示例，成功后设备 log 显示订阅成功。

设备 log 如下所示：

```
msh />ali_mqtt_test start
ali_mqtt_main|645 :: iotkit-embedded sdk version: V2.10
[inf] iotx_device_info_init(40): device_info created successfully!
[dbg] iotx_device_info_set(50): start to set device info!
[dbg] iotx_device_info_set(64): device_info set successfully!
...
[inf] iotx_mc_init(1703): MQTT init success!
[inf] _ssl_client_init(175): Loading the CA root certificate ...
...
[inf] _TLSConnectNetwork(420):   . Verifying peer X.509 certificate..
[inf] _real_confirm(92): certificate verification result: 0x200
[inf] iotx_mc_connect(2035): mqtt connect success!
...
[inf] iotx_mc_subscribe(1388): mqtt subscribe success,topic = /sys/
    a1HETlEuvri/RGB-LED-DEV-1/thing/service/property/set!
[inf] iotx_mc_subscribe(1388): mqtt subscribe success,topic = /sys/
    a1HETlEuvri/RGB-LED-DEV-1/thing/event/property/post_reply!
[dbg] iotx_mc_cycle(1269): SUBACK
event_handle|124 :: subscribe success, packet-id=0
[dbg] iotx_mc_cycle(1269): SUBACK
event_handle|124 :: subscribe success, packet-id=0
[inf] iotx_mc_keepalive_sub(2226): send MQTT ping...
[inf] iotx_mc_cycle(1295): receive ping response!
```

设备发布消息

使用 **ali_mqtt_test pub open** 命令发送 LED 状态到云端，成功后设备 log 显示**成功码 200**。

设备 log 如下所示：

```
msh />ali_mqtt_test pub open
...
[dbg] iotx_mc_cycle(1277): PUBLISH
[dbg] iotx_mc_handle_recv_PUBLISH(1091):         Packet Ident : 00000000
[dbg] iotx_mc_handle_recv_PUBLISH(1092):         Topic Length : 57
[dbg] iotx_mc_handle_recv_PUBLISH(1096):           Topic Name : /sys/
    a1HETlEuvri/RGB-LED-DEV-1/thing/service/property/set
[dbg] iotx_mc_handle_recv_PUBLISH(1099):     Payload Len/Room : 100 / 962
[dbg] iotx_mc_handle_recv_PUBLISH(1100):       Receive Buflen : 1024
[dbg] iotx_mc_handle_recv_PUBLISH(1111): delivering msg ...
[dbg] iotx_mc_deliver_message(866): topic be matched
_demo_message_arrive|182 :: ----
_demo_message_arrive|183 :: packetId: 0
_demo_message_arrive|187 :: Topic: '/sys/a1HETlEuvri/RGB-LED-DEV-1/thing/
    service/property/set' (Length: 57)
_demo_message_arrive|191 :: Payload:
'{"method": "thing.service.property.set","id": "36195462","params":{"
    LightSwitch":1},"version":"1.0.0"}' (Length: 100)
_demo_message_arrive|192 :: ----
```

云端查看发布的消息

在设备详情里的**运行状态**里可以查看设备的上报到云端的消息内容。

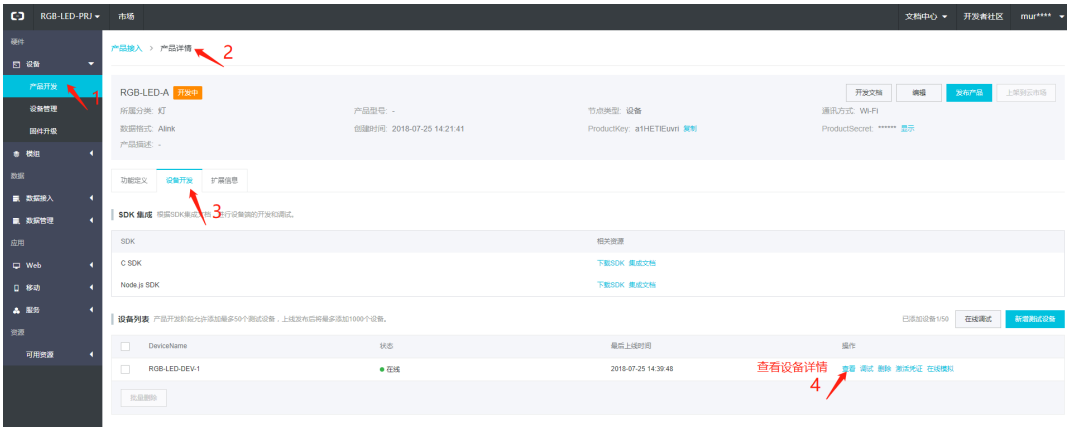

图 2.10: *查看设备详情*

图 2.11: *查看设备运行状态*

云端推送消息到设备

使用云端的调试控制台给设备推送消息。

- 打开调试控制台

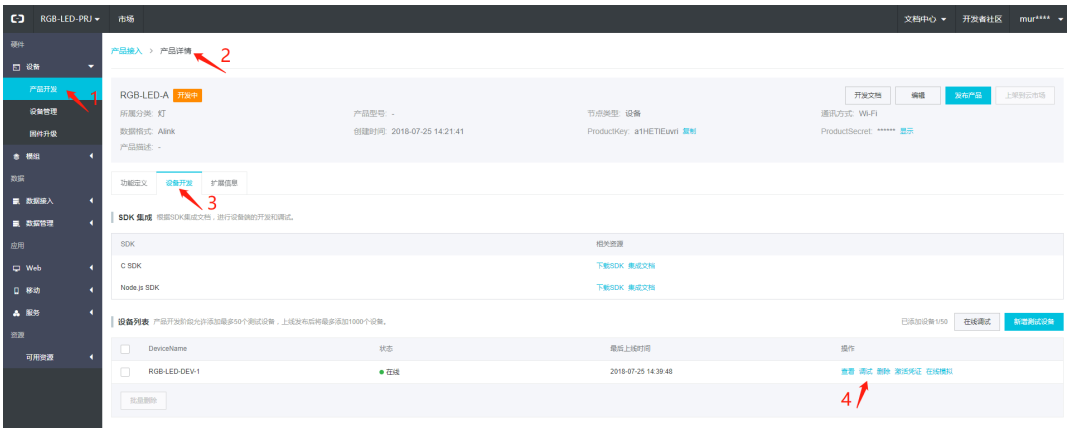

图 2.12: *打开调试控制台*

- 发送调试命令

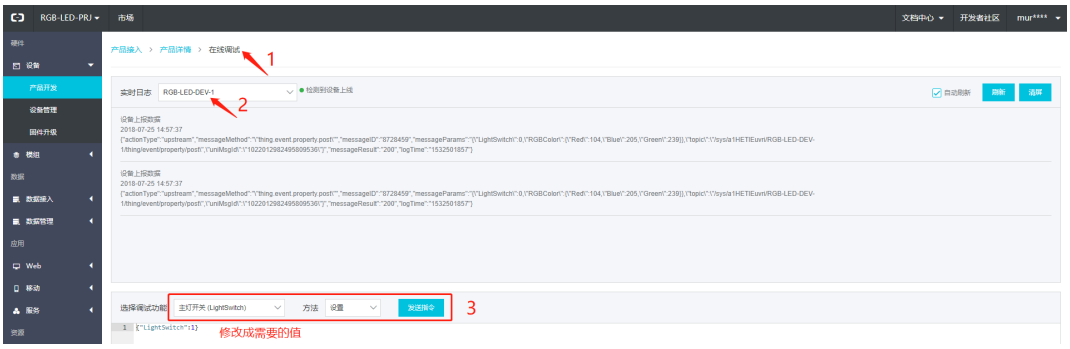

图 2.13: *在线调试页面*

查看设备订阅日志

使用调试控制台发送命令后，设备可以接受到命令，log 如下所示：

```
[dbg] iotx_mc_handle_recv_PUBLISH(1091):         Packet Ident : 00000000
[dbg] iotx_mc_handle_recv_PUBLISH(1092):         Topic Length : 52
[dbg] iotx_mc_handle_recv_PUBLISH(1096):           Topic Name : /sys/
    a1Ayv8xhoIl/RGB-DEV1/thing/service/property/set
[dbg] iotx_mc_handle_recv_PUBLISH(1099):     Payload Len/Room : 100 / 967
[dbg] iotx_mc_handle_recv_PUBLISH(1100):       Receive Buflen : 1024
[dbg] iotx_mc_handle_recv_PUBLISH(1111): delivering msg ...
[dbg] iotx_mc_deliver_message(866): topic be matched
_demo_message_arrive|178 :: ----
_demo_message_arrive|179 :: packetId: 0
_demo_message_arrive|183 :: Topic: '/sys/a1Ayv8xhoIl/RGB-DEV1/thing/
    service/property/set' (Length: 52)
_demo_message_arrive|187 :: Payload:
'{"method":"thing.service.property.set","id":"35974024","params":{"
    LightSwitch":0},"version":"1.0.0"}' (Length: 100)
_demo_message_arrive|188 :: ----
```

退出 MQTT 示例

使用 **ali_mqtt_test stop** 命令退出 MQTT 示例，设备 log 如下所示：

```
msh />ali_mqtt_test stop
[inf] iotx_mc_unsubscribe(1423): mqtt unsubscribe success,topic = /sys/
    a1HETlEuvri/RGB-LED-DEV-1/thing/event/property/post_reply!
[inf] iotx_mc_unsubscribe(1423): mqtt unsubscribe success,topic = /sys/
    a1HETlEuvri/RGB-LED-DEV-1/thing/service/property/set!
event_handle|136 :: unsubscribe success, packet-id=0
event_handle|136 :: unsubscribe success, packet-id=0
[dbg] iotx_mc_disconnect(2121): rc = MQTTDisconnect() = 0
[inf] _network_ssl_disconnect(514): ssl_disconnect
[inf] iotx_mc_disconnect(2129): mqtt disconnect!
[inf] iotx_mc_release(2175): mqtt release!
[err] LITE_dump_malloc_free_stats(594): WITH_MEM_STATS = 0
mqtt_client|329 :: out of sample!
```

2.1.3 OTA 示例

固件升级支持对设备的固件进行远程空中升级（Over-The-Air），实现对设备的远程维护、功能升级、问题修复等场景的使用。您可以指定产品新增一个固件，对固件进行验证，验证通过后开始批量升级，并在固件详情中查看升级结果。

示例文件

示例程序路径	验证平台	说明
samples/ota/ ota_mqtt-example.c	LinkDevelop, LinkPlatform	基于 MQTT 通道的设备 OTA 例程

命令列表

例程中，使用 MSH 命令启动 OTA 例程，命令如下所示：

命令	说明
ali_ota_test start	启动 OTA 示例
ali_ota_test stop	手动退出 OTA 示例

运行 OTA 示例

使用 **ali_ota_test start** 命令启动 OTA 例程，然后等待云端发送 OTA 指令。

设备 log 如下所示：

```
msh />ali_ota_test start
ali_ota_main|372 :: iotkit-embedded sdk version: V2.10
[inf] iotx_device_info_init(40): device_info created successfully!
[dbg] iotx_device_info_set(50): start to set device info!
[dbg] iotx_device_info_set(64): device_info set successfully!
...
[inf] iotx_mc_init(1703): MQTT init success!
[inf] _ssl_client_init(175): Loading the CA root certificate ...
...
[inf] _TLSConnectNetwork(420):   . Verifying peer X.509 certificate..
[inf] _real_confirm(92): certificate verification result: 0x200
[inf] iotx_mc_connect(2035): mqtt connect success!
...
[inf] iotx_mc_subscribe(1388): mqtt subscribe success,topic = /ota/device
    /upgrade/a1HETlEuvri/RGB-LED-DEV-1!
mqtt_client|241 :: wait ota upgrade command....
[dbg] iotx_mc_cycle(1260): PUBACK
event_handle|130 :: publish success, packet-id=2
[dbg] iotx_mc_cycle(1269): SUBACK
event_handle|106 :: subscribe success, packet-id=1
mqtt_client|241 :: wait ota upgrade command....
mqtt_client|241 :: wait ota upgrade command....
```

新增固件

这里需要用户上传一个 bin 类型的测试固件，随意一个 bin 固件即可，演示例程只进行固件下载及校验，不会写入 Flash，所以也不会真正进行固件搬运升级。

图 2.14: *LinkDevelop 平台新增 OTA 固件*

验证固件

图 2.15: *LinkDevelop 平台验证 OTA 固件*

设备日志

推送成功后，设备开始下载固件，下载完成后自动进行固件完整性校验，设备端测试日志如下所示：

```
...
mqtt_client|254 :: Here write OTA data to file....
[dbg] IOT_OTA_Ioctl(457):
origin=e4e54df52a3b530c7e0544b2872f1305, now=
    e4e54df52a3b530c7e0544b2872f1305
mqtt_client|280 :: The firmware is valid!  Download firmware successfully
    .
mqtt_client|294 :: OTA FW version: v10
```

云端升级进度展示

设备升级过程中云端会显示设备下载固件的进度，固件下载完成并校验固件成功，设备 SDK 上报新的版本号到云端，云端会显示升级成功，如下图所示：

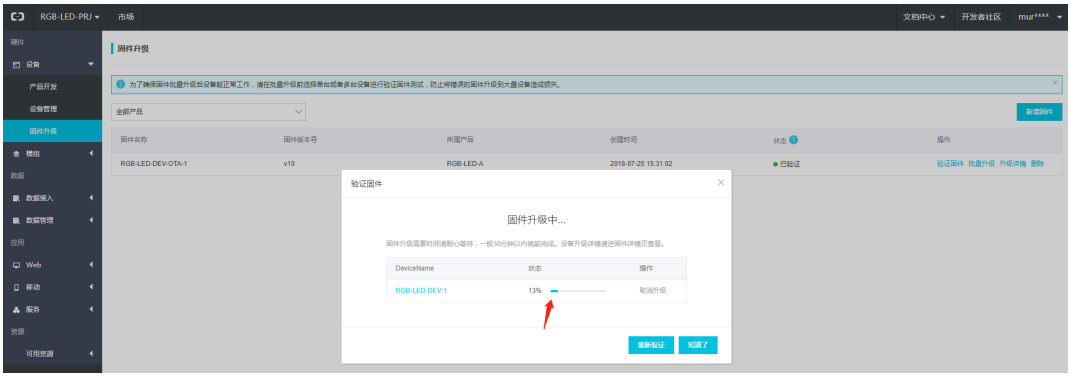

图 2.16: *升级进度*

升级进度 100% 后，再次运行 `ali_ota_test start` 命令，将最新的版本号上传到云端，版本号匹配成功后，云端显示升级成功，如下图所示：

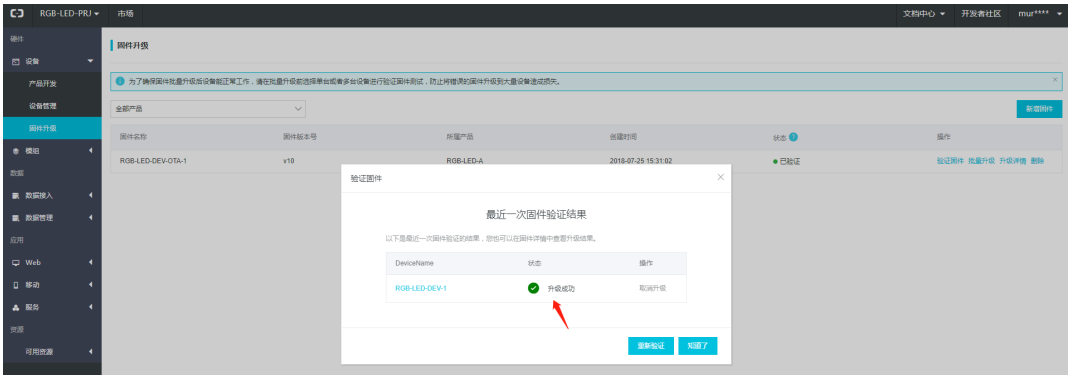

图 2.17: *升级成功*

退出 OTA 例程

升级成功或者升级失败会自动退出 OTA 例程，如果需要手动退出 OTA 例程，请使用 `ali_ota_test stop` 命令。

```
msh />ali_ota_test stop
msh />[dbg] iotx_mc_disconnect(2121): rc = MQTTDisconnect() = 0
[inf] _network_ssl_disconnect(514): ssl_disconnect
[inf] iotx_mc_disconnect(2129): mqtt disconnect!
[inf] iotx_mc_release(2175): mqtt release!
[err] LITE_dump_malloc_free_stats(594): WITH_MEM_STATS = 0
mqtt_client|340 :: out of sample!
```

2.2 LinkPlatform 平台

2.2.1 准备工作

- 注册 LinkPlatform 平台

图 2.18: *注册 LinkPlatform 平台*

- 创建产品

图 2.19: *创建产品*

- 添加设备

  在设备管理菜单下，新增一个测试设备，点击查看进入设备详情页面。

  成功创建设备后，可以获取到设备激活需要的三元组（**ProductKey**、**DeviceName**、**DeviceSecret**），后面需要使用 **menuconfig** 配置到设备 SDK 中。

图 2.20: 添加设备

图 2.21: 设备激活凭证

- 查看消息 Topic 列表

  进入设备详情页面，然后在 **Topic 列表**选项查看创建设备默认分配的 Topic 列表，以及 Topic 权限。

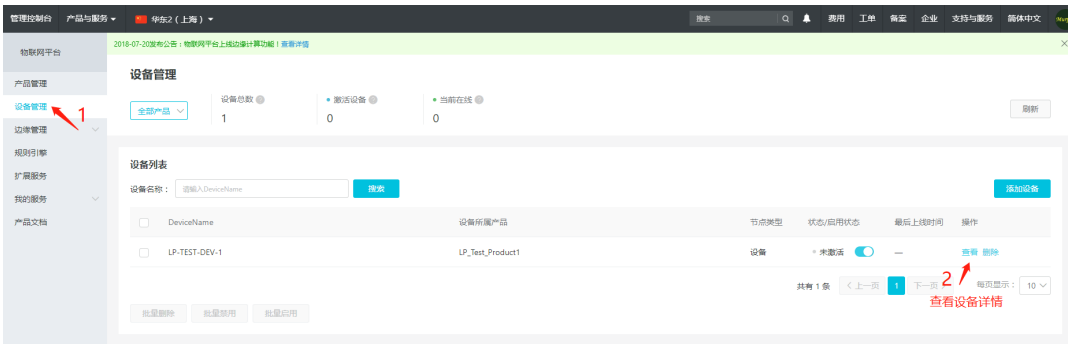

图 2.22: 进入设备详情页

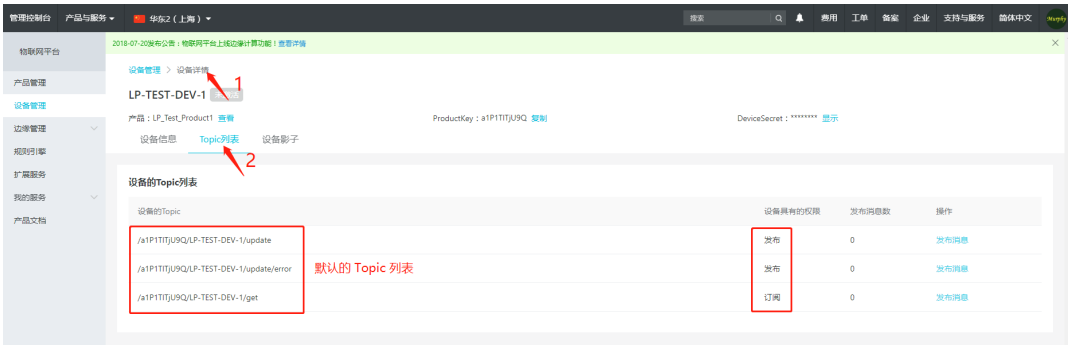

图 2.23: *查看 MQTT Topic 列表*

- **自定义 Topic**

  MQTT 示例程序中会用到名为 **data** 的 Topic，Topic 权限为**发布和订阅**，因此这里必须自定义一个 **data** Topic，如下图所示：

图 2.24: *自定义 Topic*

- **开通固件升级服务**

图 2.25: *开通固件升级服务*

图 2.26: *开通固件升级服务*

- **获取软件包**

  打开 RT-Thread 提供的 ENV 工具，使用 **menuconfig** 配置软件包。

  – 配置 iotkit 软件包

  配置使能 iotkit 软件包并**填写设备激活凭证**。

  `menuconfig` 中选择阿里云平台为 **LinkPlatform**，`OTA channel` 选择 **MQTT**（以 MQTT 为例），详细的配置如下所示：

```
RT-Thread online packages  --->
    IoT - internet of things  --->
        IoT Cloud  --->
          [*] Ali-iotkit:  Ali Cloud SDK for IoT platform  --->
                Select Aliyun platform (LinkPlatform Platform)  --->
          (a1dSQSGZ77X) Config Product Key
          (RGB-LED-DEV-1) Config Device Name
          (Ghuiyd9nmGowdZzjPqFtxhm3WUHEbIlI) Config Device Secret
          -*-   Enable MQTT
          [*]     Enable MQTT sample
          [*]     Enable MQTT direct connect
          [*]     Enable SSL
          [ ]   Enable COAP
          [*]   Enable OTA
                      Select OTA channel (Use MQTT OTA channel)
                          --->
                    version (latest)  --->
```

  – 增加 `mbedTLS` 帧大小

  阿里 TLS 认证过程中数据包较大，这里需要增加 TLS 帧大小，OTA 的时候**至少需要 8K** 大小。

  打开 RT-Thread 提供的 ENV 工具，使用 **menuconfig** 配置 **TLS** 帧大小。

```
RT-Thread online packages  --->
    security packages  --->
      -*- mbedtls:An open source, portable, easy to use,
                  readable and flexible SSL library  --->
      (8192) Maxium fragment length in bytes
```

– 使用 `pkgs --update` 命令下载软件包

2.2.2 MQTT 示例

该 MQTT 示例程序以**data Topic** 为例，演示了如何在设备上使用 MQTT + TLS/SSL 通道与阿里云平台建立双向通信。

示例文件

示例程序路径	验证平台	说明
samples/mqtt/ mqtt-example.c	LinkDevelop, LinkPlatform	基于 MQTT 通道的设备和云双向通信例程

命令列表

例程中，使用 MSH 命令启动 MQTT 例程，命令如下所示：

命令	说明
ali_mqtt_test start	启动 MQTT 示例
ali_mqtt_test pub open	开灯，并向云端同步开灯状态
ali_mqtt_test pub close	关灯，并向云端同步关灯状态
ali_mqtt_test stop	停止 MQTT 示例

启动 MQTT

使用 **`ali_mqtt_test start`** 命令启动 MQTT 示例，成功后设备 log 显示订阅成功。

设备 log 如下所示：

```
msh />ali_mqtt_test start
ali_mqtt_main|645 :: iotkit-embedded sdk version: V2.10
[inf] iotx_device_info_init(40): device_info created successfully!
[dbg] iotx_device_info_set(50): start to set device info!
```

```
[dbg] iotx_device_info_set(64): device_info set successfully!
...
[inf] iotx_mc_init(1703): MQTT init success!
[inf] _ssl_client_init(175): Loading the CA root certificate ...
...
[inf] _TLSConnectNetwork(420):   . Verifying peer X.509 certificate..
[inf] _real_confirm(92): certificate verification result: 0x200
[inf] iotx_mc_connect(2035): mqtt connect success!
...
[inf] iotx_mc_subscribe(1388): mqtt subscribe success,topic = /
    a1P1TlTjU9Q/LP-TEST-DEV-1/data!
[dbg] iotx_mc_cycle(1269): SUBACK
event_handle|124 :: subscribe success, packet-id=0
[inf] iotx_mc_keepalive_sub(2226): send MQTT ping...
[inf] iotx_mc_cycle(1295): receive ping response!
```

设备发布消息

使用 **ali_mqtt_test pub open** 命令发送消息 **data** Topic。

设备 log 如下所示：

```
msh />ali_mqtt_test pub open
ali_mqtt_test_pub|583 ::
 publish message:
 topic: /a1P1TlTjU9Q/LP-TEST-DEV-1/data
 payload: {"id" : "1","version":"1.0","params" : {"RGBColor" : {"Red"
     :247,"Green":60,"Blue":74},"LightSwitch" : 1},"method":"thing.event.
     property.post"}
 rc = 3
msh />[dbg] iotx_mc_cycle(1260): PUBACK
event_handle|148 :: publish success, packet-id=0
[dbg] iotx_mc_cycle(1260): PUBACK
event_handle|148 :: publish success, packet-id=0
[dbg] iotx_mc_cycle(1277): PUBLISH
...
[dbg] iotx_mc_handle_recv_PUBLISH(1100):       Receive Buflen : 1024
[dbg] iotx_mc_handle_recv_PUBLISH(1111): delivering msg ...
[dbg] iotx_mc_deliver_message(866): topic be matched
_demo_message_arrive|182 :: ----
_demo_message_arrive|183 :: packetId: 19324
_demo_message_arrive|187 :: Topic: '/a1P1TlTjU9Q/LP-TEST-DEV-1/data' (
    Length: 31)
_demo_message_arrive|191 :: Payload:
'{"id" : "1","version":"1.0","params" : {"RGBColor" : {"Red":247,"Green
    ":60,"Blue":74},"LightSwitch" : 1},
```

```
"method":"thing.event.property.post"}' (Length: 142)
_demo_message_arrive|192 :: ----
```

查看云端日志

在设备详情里的**运行状态**里可以查看设备的上报到云端的消息内容。

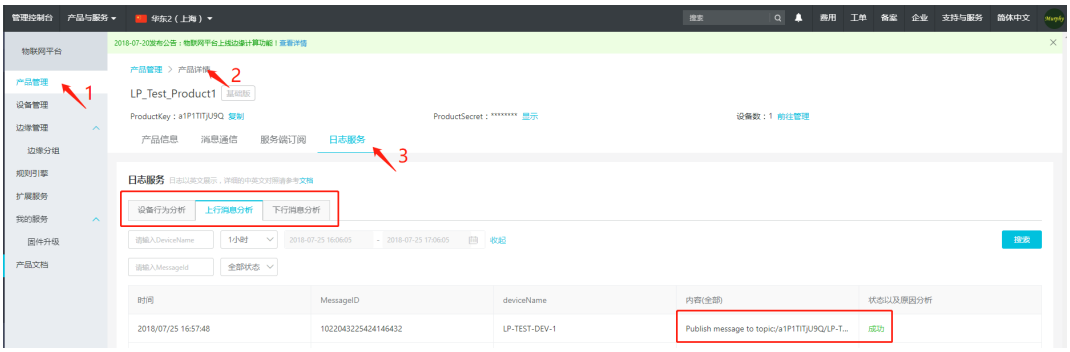

图 2.27: *查看云端日志*

云端推送消息到设备

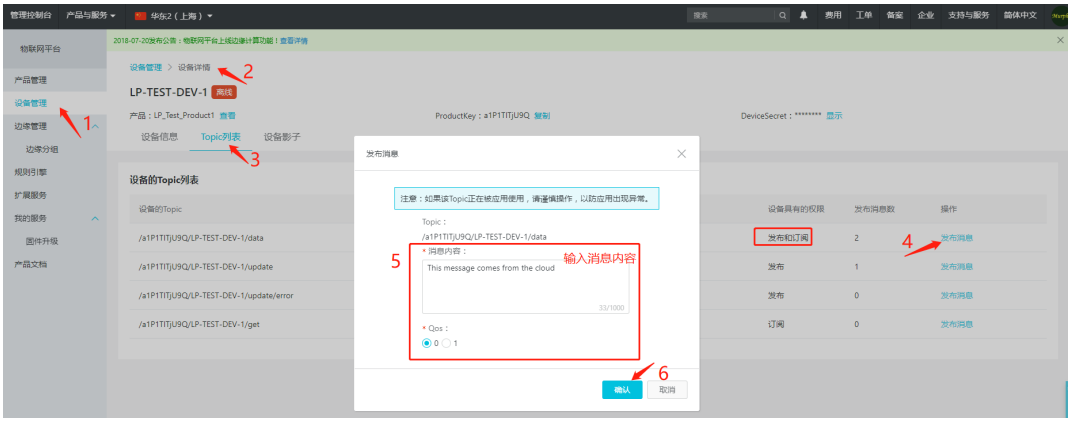

图 2.28: *云端发布消息到设备*

查看设备订阅日志

使用调试控制台发送命令后，设备可以接受到命令，log 如下所示：

```
msh />[dbg] iotx_mc_cycle(1277): PUBLISH
[dbg] iotx_mc_handle_recv_PUBLISH(1091):          Packet Ident : 00000000
[dbg] iotx_mc_handle_recv_PUBLISH(1092):          Topic Length : 31
[dbg] iotx_mc_handle_recv_PUBLISH(1096):            Topic Name : /
    a1P1TlTjU9Q/LP-TEST-DEV-1/data
[dbg] iotx_mc_handle_recv_PUBLISH(1099):      Payload Len/Room : 33 / 989
[dbg] iotx_mc_handle_recv_PUBLISH(1100):        Receive Buflen : 1024
```

```
[dbg] iotx_mc_handle_recv_PUBLISH(1111): delivering msg ...
[dbg] iotx_mc_deliver_message(866): topic be matched
_demo_message_arrive|182 :: ----
_demo_message_arrive|183 :: packetId: 0
_demo_message_arrive|187 :: Topic: '/a1P1TlTjU9Q/LP-TEST-DEV-1/data' (
    Length: 31)
_demo_message_arrive|191 :: Payload: 'This message comes from the cloud'
    (Length: 33)
_demo_message_arrive|192 :: ----
```

退出 MQTT 示例

使用 **ali_mqtt_test stop** 命令退出 MQTT 示例，设备 log 如下所示：

```
msh />ali_mqtt_test stop
msh />[inf] iotx_mc_unsubscribe(1423): mqtt unsubscribe success,topic = /
    a1P1TlTjU9Q/LP-TEST-DEV-1/data!
event_handle|136 :: unsubscribe success, packet-id=0
[dbg] iotx_mc_disconnect(2121): rc = MQTTDisconnect() = 0
[inf] _network_ssl_disconnect(514): ssl_disconnect
[inf] iotx_mc_disconnect(2129): mqtt disconnect!
[inf] iotx_mc_release(2175): mqtt release!
[err] LITE_dump_malloc_free_stats(594): WITH_MEM_STATS = 0
mqtt_client|329 :: out of sample!
```

2.2.3 OTA 示例

固件升级支持对设备的固件进行远程空中升级（Over-The-Air），实现对设备的远程维护、功能升级、问题修复等场景的使用。您可以指定产品新增一个固件，对固件进行验证，验证通过后开始批量升级，并在固件详情中查看升级结果。

示例文件

示例程序路径	验证平台	说明
samples/ota/ ota_mqtt-example.c	LinkDevelop, LinkPlatform	基于 MQTT 通道的设备 OTA 例程

命令列表

例程中，使用 MSH 命令启动 OTA 例程，命令如下所示：

命令	说明
ali_ota_test start	启动 OTA 示例
ali_ota_test stop	手动退出 OTA 示例

运行 OTA 示例

使用 **ali_ota_test start** 命令启动 OTA 例程，然后等待云端发送 OTA 指令。

设备 log 如下所示：

```
msh />ali_ota_test start
ali_ota_main|372 :: iotkit-embedded sdk version: V2.10
[inf] iotx_device_info_init(40): device_info created successfully!
[dbg] iotx_device_info_set(50): start to set device info!
[dbg] iotx_device_info_set(64): device_info set successfully!
...
[inf] iotx_mc_init(1703): MQTT init success!
[inf] _ssl_client_init(175): Loading the CA root certificate ...
...
[inf] _TLSConnectNetwork(420):   . Verifying peer X.509 certificate..
[inf] _real_confirm(92): certificate verification result: 0x200
[inf] iotx_mc_connect(2035): mqtt connect success!
...
[dbg] iotx_mc_report_mid(2292): MID Report: topic name = '/sys/
    a1P1TlTjU9Q/LP-TEST-DEV-1/thing/status/update'
[dbg] iotx_mc_report_mid(2309): MID Report: finished, IOT_MQTT_Publish()
    = 0
[inf] iotx_mc_subscribe(1388): mqtt subscribe success,topic = /ota/device
    /upgrade/a1P1TlTjU9Q/LP-TEST-DEV-1!
mqtt_client|241 :: wait ota upgrade command....
mqtt_client|241 :: wait ota upgrade command....
```

新增固件

这里需要用户上传一个 bin 类型的测试固件，随意一个 bin 固件即可，演示例程只进行固件下载及校验，不会写入 Flash，所以也不会真正进行固件搬运升级。

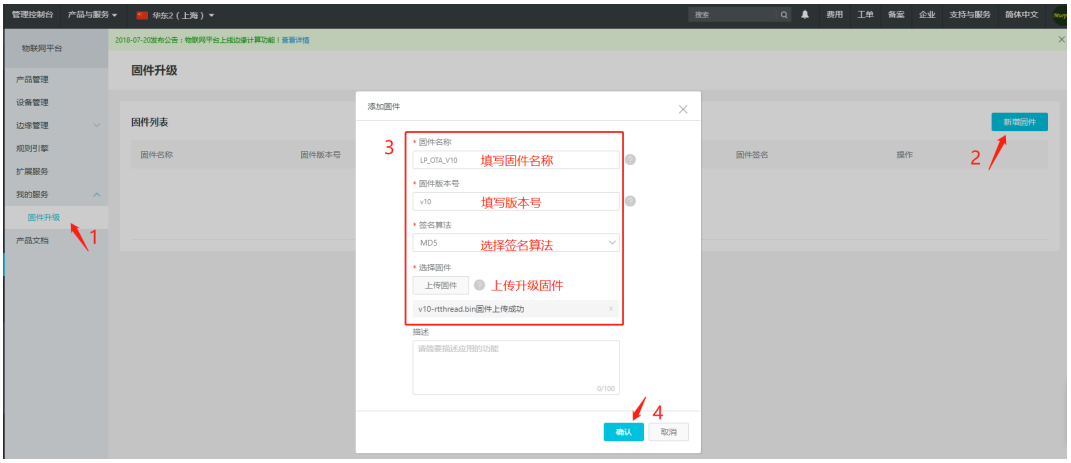

图 2.29: *LinkPlatform 平台新增 OTA 固件*

验证固件

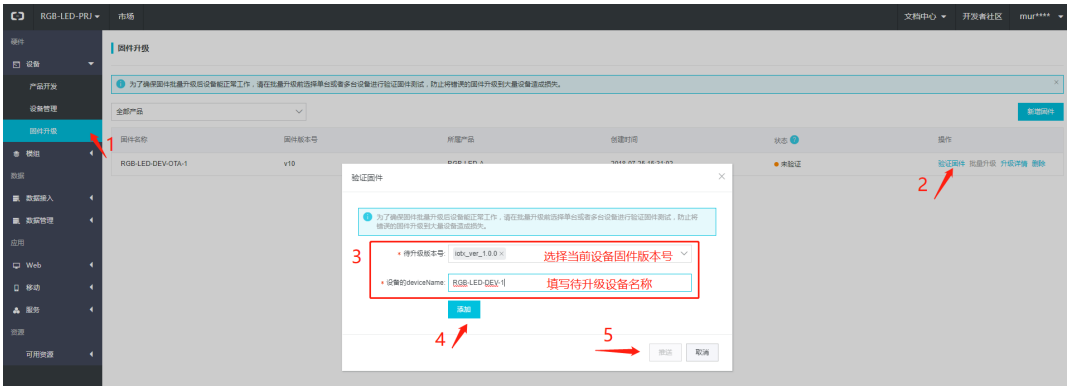

图 2.30: *LinkPlatform 验证 OTA 固件*

设备日志

推送成功后，设备开始下载固件，下载完成后自动进行固件完整性校验，设备端测试日志如下所示：

```
...
mqtt_client|254 :: Here write OTA data to file....
[dbg] IOT_OTA_Ioctl(457):
origin=e4e54df52a3b530c7e0544b2872f1305, now=
    e4e54df52a3b530c7e0544b2872f1305
mqtt_client|280 :: The firmware is valid!  Download firmware successfully
    .
mqtt_client|294 :: OTA FW version: v10
```

云端升级进度展示

设备升级过程中云端会显示设备下载固件的进度，固件下载完成并校验固件成功，设备 SDK 上报新的版本号到云端，云端会显示升级成功，如下图所示：

图 2.31: *升级进度*

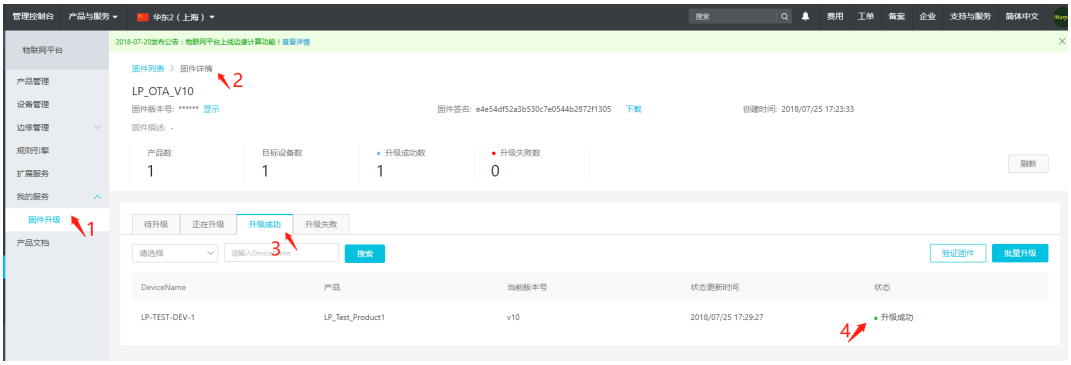

图 2.32: *升级成功*

退出 OTA 例程

升级成功或者升级失败会自动退出 OTA 例程，如果需要手动退出 OTA 例程，请使用 `ali_ota_test stop` 命令。

```
msh />ali_ota_test stop
msh />[dbg] iotx_mc_disconnect(2121): rc = MQTTDisconnect() = 0
[inf] _network_ssl_disconnect(514): ssl_disconnect
[inf] iotx_mc_disconnect(2129): mqtt disconnect!
[inf] iotx_mc_release(2175): mqtt release!
[err] LITE_dump_malloc_free_stats(594): WITH_MEM_STATS = 0
mqtt_client|340 :: out of sample!
```

2.3 注意事项

- 使用前请在 `menuconfig` 里配置自己的设备激活凭证（PRODUCT_KEY、DEVICE_NAME 和 DEVICE_SECRET）

- 使用 `menuconfig` 配置选择要接入的平台（**LinkDevelop** 或者 **LinkPlatform**）
- 开启 OTA 功能必须使能加密连接，默认选择（因为 OTA 升级**必须使用 HTTPS** 下载固件）

2.4 常见问题

- MbedTLS 返回 0x7200 错误

  通常是由于 MbedTLS 帧长度过小，请增加 MbedTLS 帧长度（**至少需要 8K 大小**）。

第 3 章

工作原理

iotkit SDK 为了方便设备上云封装了丰富的连接协议，如 MQTT、CoAP、HTTP、TLS，并且对硬件平台进行了抽象，使其不收具体的硬件平台限制而更加灵活。

通常用户并不需要关心 SDK 底层的实现机制，而只需要了解设备如何通过 SDK 与云端进行数据交互即可，方便用户理解如何使用应用层 API 接口进行业务逻辑编写。这里举例展示了 MQTT 和 OTA 应用的数据交互流程。

3.1 MQTT 数据交互流程

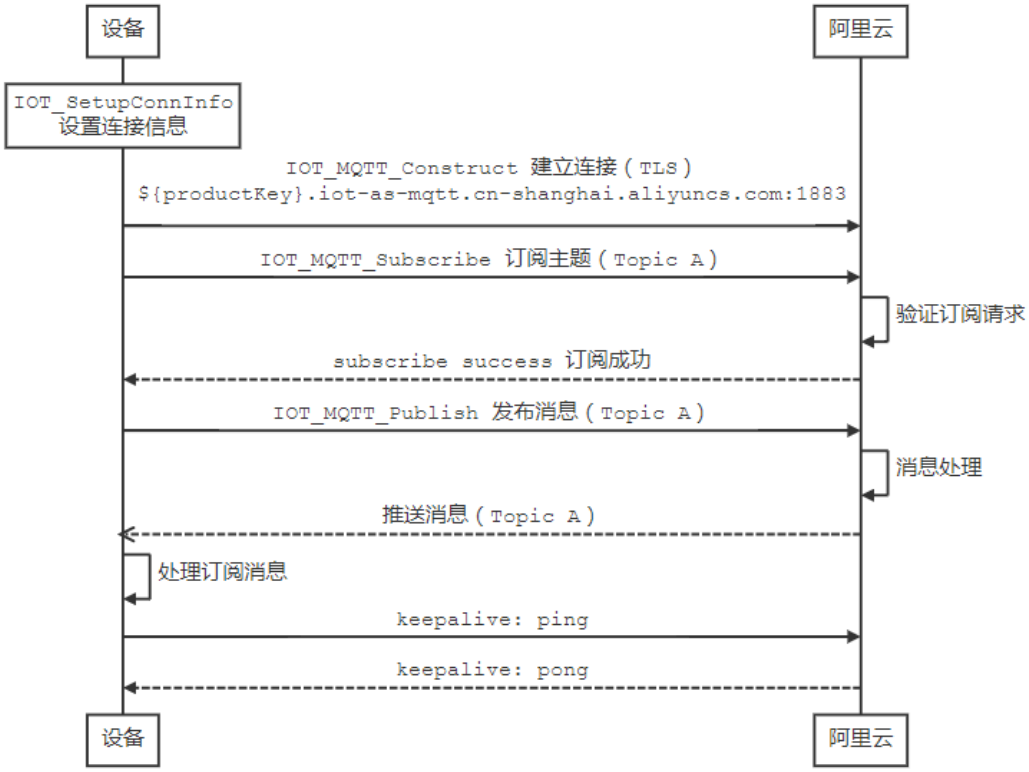

图 3.1: *MQTT 数据交互流程*

3.2 OTA 数据交互流程

以 MQTT 通道为例，固件升级流程如下图所示：

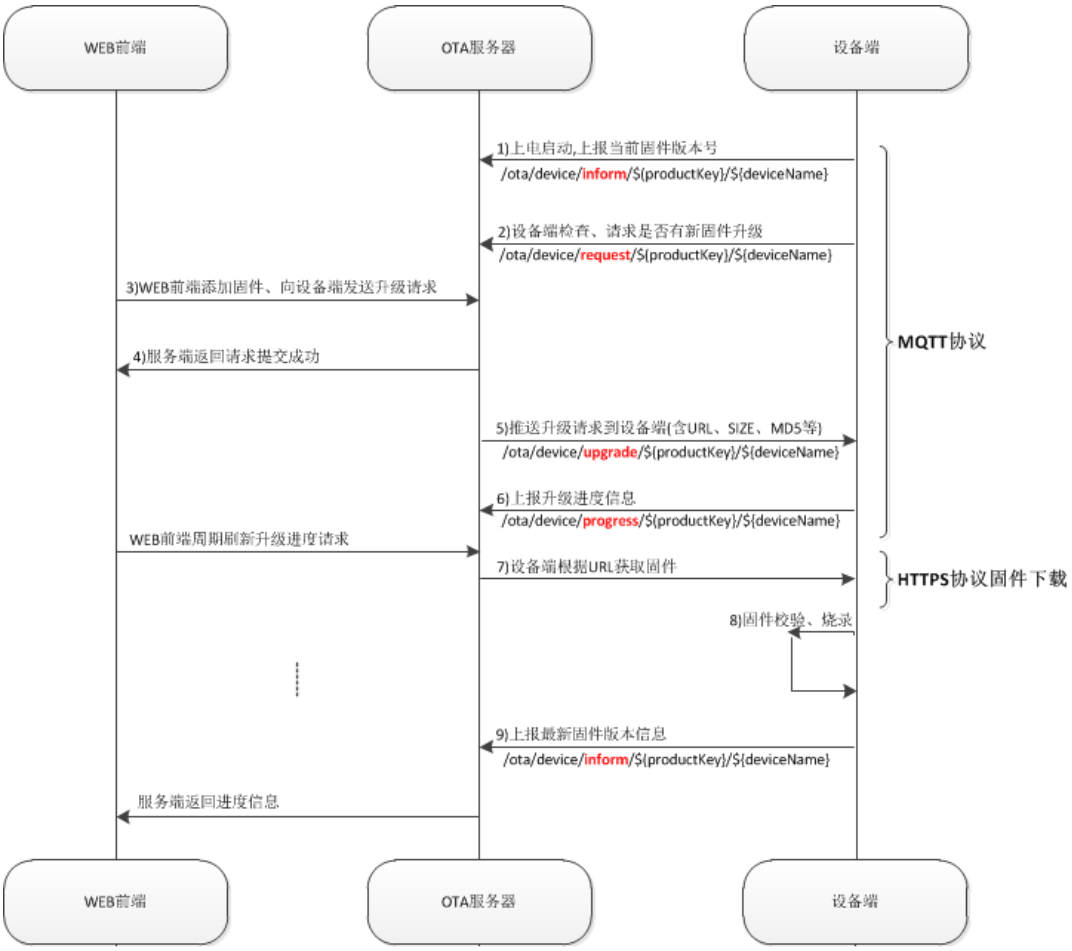

图 3.2: *OTA 固件升级流程*

第 4 章

使用指南

ali-iotkit 软件包封装了 HTTP、MQTT、CoAP 和 OTA 等应用层协议，方便了用户设备接入云平台，这里摘取部分做简要介绍。

4.1 MQTT 连接

目前阿里云支持 MQTT 标准协议接入，兼容 3.1.1 和 3.1 版本协议，具体的协议请参考 MQTT 3.1.1 和 MQTT 3.1 协议文档。

4.1.1 特征

- 支持 MQTT 的 PUB、SUB、PING、PONG、CONNECT、DISCONNECT 和 UNSUB 等报文
- 支持 cleanSession
- 不支持 will、retain msg
- 不支持 QOS2
- 基于原生的 MQTT topic 上支持 RRPC 同步模式，服务器可以同步调用设备并获取设备回执结果

4.1.2 安全等级

支持 TLSV1、TLSV1.1 和 TLSV1.2 版本的协议建立安全连接。

- TCP 通道基础＋芯片级加密（ID2 硬件集成）：安全级别高
- TCP 通道基础＋对称加密（使用设备私钥做对称加密）：安全级别中
- TCP 方式（数据不加密）：安全级别低

4.1.3 连接域名

- 华东 2 节点：***productKey***.iot-as-mqtt.cn-shanghai.aliyuncs.com:1883
- 美西节点：***productKey***.iot-as-mqtt.us-west-1.aliyuncs.com:1883
- 新加坡节点：***productKey***.iot-as-mqtt.ap-southeast-1.aliyuncs.com:1883

4.1.4 Topic 规范

默认情况下创建一个产品后，产品下的所有设备都拥有以下 Topic 类的权限：

- /***productKey***/***deviceName***/update pub
- /***productKey***/***deviceName***/update/error pub
- /***productKey***/***deviceName***/get sub
- /sys/***productKey***/***deviceName***/thing/# pub&sub
- /sys/***productKey***/***deviceName***/rrpc/# pub&sub
- /broadcast/***productKey***/# pub&sub

每个 Topic 规则称为 topic 类，topic 类实行设备维度隔离。每个设备发送消息时，将 deviceName 替换为自己设备的 deviceName ，防止 topic 被跨设备越权，topic 说明如下：

- pub：表示数据上报到 topic 的权限
- sub：表示订阅 topic 的权限
- /***productKey***/***deviceName***/xxx 类型的 topic 类：可以在物联网平台的控制台扩展和自定义
- /sys 开头的 topic 类：属于系统约定的应用协议通信标准，不允许用户自定义的，约定的 topic 需要符合阿里云 ALink 数据标准
- /sys/***productKey***/***deviceName***/thing/xxx 类型的 topic 类：网关主子设备使用的 topic 类，用于网关场景
- /broadcast 开头的 topic 类：广播类特定 topic
- /sys/***productKey***/***deviceName***/rrpc/request/${messageId}：用于同步请求，服务器会对消息 Id 动态生成 topic，设备端可以订阅通配符
- /sys/***productKey***/***deviceName***/rrpc/request/+：收到消息后，发送 pub 消息到 /sys/***productKey***/***deviceName***/rrpc/response/${messageId}，服务器可以在发送请求时，同步收到结果

4.1.5 建立 MQTT 连接

使用 IOT_MQTT_Construct 接口与云端建立 MQTT 连接。

如果要实现设备长期在线，需要程序代码中去掉 IOT_MQTT_Unregister 和 IOT_MQTT_Destroy 部分，使用 while 保持长连接状态。

示例代码如下：

```
while(1)
{
    IOT_MQTT_Yield(pclient, 200);
    HAL_SleepMs(100);
}
```

4.1.6 订阅 Topic 主题

使用 IOT_MQTT_Subscribe 接口订阅某个 Topic。

代码如下：

```
/* Subscribe the specific topic */
rc = IOT_MQTT_Subscribe(pclient, TOPIC_DATA, IOTX_MQTT_QOS1,
                        _demo_message_arrive, NULL);
if (rc < 0) {
    IOT_MQTT_Destroy(&pclient);
    EXAMPLE_TRACE("IOT_MQTT_Subscribe() failed, rc = %d", rc);
    rc = -1;
    goto do_exit;
}
```

4.1.7 发布消息

使用 IOT_MQTT_Publish 接口发布信息到云端。

代码如下：

```
/* Initialize topic information */
memset(&topic_msg, 0x0, sizeof(iotx_mqtt_topic_info_t));
strcpy(msg_pub, "message: hello! start!");
topic_msg.qos = IOTX_MQTT_QOS1;
topic_msg.retain = 0;
topic_msg.dup = 0;
topic_msg.payload = (void *)msg_pub;
topic_msg.payload_len = strlen(msg_pub);
rc = IOT_MQTT_Publish(pclient, TOPIC_DATA, &topic_msg);
EXAMPLE_TRACE("rc = IOT_MQTT_Publish() = %d", rc);
```

4.1.8 取消订阅

使用 IOT_MQTT_Unsubscribe 接口取消订阅云端消息

4.1.9 下行数据接收

使用 IOT_MQTT_Yield 数据接收函数接收来自云端的消息。

请在任何需要接收数据的地方调用这个 API。如果系统允许，请起一个单独的线程，执行该接口。

代码如下：

```
/* handle the MQTT packet received from TCP or SSL connection */
IOT_MQTT_Yield(pclient, 200);
```

4.1.10 销毁 MQTT 连接

使用 IOT_MQTT_Destroy 接口销毁 MQTT 连接，释放内存。

代码如下：

```
IOT_MQTT_Destroy(&pclient);
```

4.1.11 检查连接状态

使用 IOT_MQTT_CheckStateNormal 接口查看当前的连接状态。

该接口用于查询 MQTT 的连接状态。但是，该接口并不能立刻检测到设备断网，只会在有数据发送或是 keepalive 时才能侦测到 disconnect。

4.1.12 MQTT 保持连接

设备端在 keepalive_interval_ms 时间间隔内，至少需要发送一次报文，包括 ping 请求。

如果服务端在 keepalive_interval_ms 时间内无法收到任何报文，物联网平台会断开连接，设备端需要进行重连。

在 IOT_MQTT_Construct 函数可以设置 keepalive_interval_ms 的取值，物联网平台通过该取值作为心跳间隔时间。keepalive_interval_ms 的取值范围是 60000~300000。

示例代码：

```
iotx_mqtt_param_t mqtt_params;

memset(&mqtt_params, 0x0, sizeof(mqtt_params));
mqtt_params.keepalive_interval_ms = 60000;
mqtt_params.request_timeout_ms = 2000;

/* Construct a MQTT client with specify parameter */
pclient = IOT_MQTT_Construct(&mqtt_params);
```

4.2 CoAP 连接

- 支持 RFC 7252 Constrained Application Protocol 协议，具体请参考：RFC 7252
- 使用 DTLS v1.2 保证通道安全，具体请参考：DTLS v1.2
- 服务器地址 endpoint = ***productKey***.iot-as-coap.cn-shanghai.aliyuncs.com:5684
  其中 productKey 请替换为您申请的产品 Key

4.2.1 CoAP 约定

- 不支持？号形式传参数
- 暂时不支持资源发现
- 仅支持 UDP 协议，并且目前必须通过 DTLS
- URI 规范，CoAP 的 URI 资源和 MQTT TOPIC 保持一致，参考 MQTT 规范

4.2.2 应用场景

CoAP 协议适用在资源受限的低功耗设备上，尤其是 NB-IoT 的设备使用，基于 CoAP 协议将 NB-IoT 设备接入物联网平台的流程如下图所示：

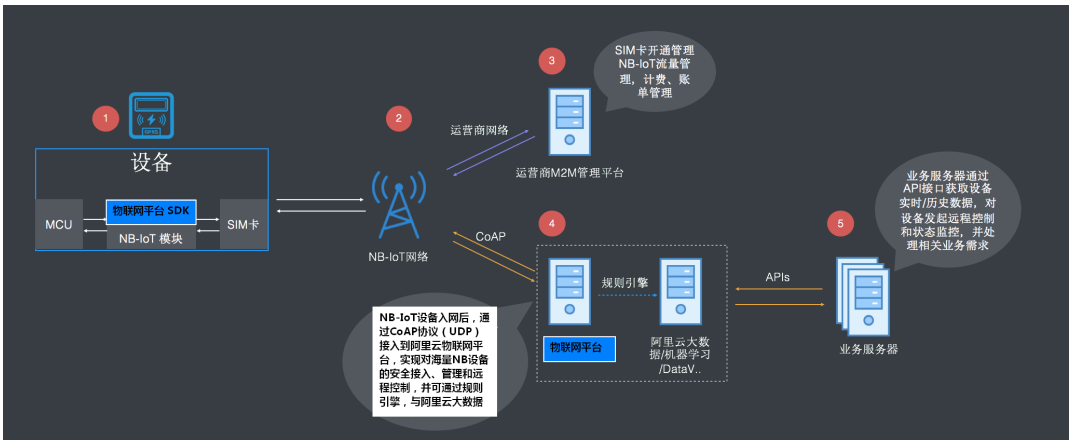

图 4.1: *CoAP 应用场景*

4.2.3 建立连接

使用 IOT_CoAP_Init 和 IOT_CoAP_DeviceNameAuth 接口与云端建立 CoAP 认证连接。

示例代码：

```
iotx_coap_context_t *p_ctx = NULL;
p_ctx = IOT_CoAP_Init(&config);
if (NULL != p_ctx) {
    IOT_CoAP_DeviceNameAuth(p_ctx);
    do {
        count ++;
        if (count == 11) {
            count = 1;
        }
    IOT_CoAP_Yield(p_ctx);
    } while (m_coap_client_running);
    IOT_CoAP_Deinit(&p_ctx);
} else {
    HAL_Printf("IoTx CoAP init failed\r\n");
}
```

4.2.4 收发数据

SDK 使用接口 IOT_CoAP_SendMessage 发送数据，使用 IOT_CoAP_GetMessagePayload 和 IOT_CoAP_GetMessageCode 接收数据。

示例代码：

```
/* send data */
static void iotx_post_data_to_server(void *param)
{
    char path[IOTX_URI_MAX_LEN + 1] = {0};
    iotx_message_t message;
    iotx_deviceinfo_t devinfo;
    message.p_payload = (unsigned char *)"{\"name\":\"hello world\"}";
    message.payload_len = strlen("{\"name\":\"hello world\"}");
    message.resp_callback = iotx_response_handler;
    message.msg_type = IOTX_MESSAGE_CON;
    message.content_type = IOTX_CONTENT_TYPE_JSON;
    iotx_coap_context_t *p_ctx = (iotx_coap_context_t *)param;
    iotx_set_devinfo(&devinfo);
```

```
    snprintf(path, IOTX_URI_MAX_LEN, "/topic/%s/%s/update/",
             (char *)devinfo.product_key,
             (char *)devinfo.device_name);
    IOT_CoAP_SendMessage(p_ctx, path, &message);
}

/* receive data */
static void iotx_response_handler(void *arg, void *p_response)
{
    int len = 0;
    unsigned char *p_payload = NULL;
    iotx_coap_resp_code_t resp_code;
    IOT_CoAP_GetMessageCode(p_response, &resp_code);
    IOT_CoAP_GetMessagePayload(p_response, &p_payload, &len);
    HAL_Printf("[APPL]: Message response code: 0x%x\r\n", resp_code);
    HAL_Printf("[APPL]: Len: %d, Payload: %s, \r\n", len, p_payload);
}
```

4.2.5 下行数据接收

使用 IOT_CoAP_Yield 接口接收来自云端的下行数据。

请在任何需要接收数据的地方调用这个 API，如果系统允许，请起一个单独的线程，执行该接口。

4.2.6 销毁 CoAP 连接

使用 IOT_CoAP_Deinit 接口销毁 CoAP 连接并释放内存。

4.3 OTA 升级

4.3.1 固件升级 Topic

- 设备端上报固件版本给云端

  /ota/device/inform/***productKey***/***deviceName***

- 设备端订阅该 topic 接收云端固件升级通知

  /ota/device/upgrade/***productKey***/***deviceName***

- 设备端上报固件升级进度

/ota/device/progress/***productKey***/***deviceName***

- 设备端请求是否固件升级

  /ota/device/request/***productKey***/***deviceName***

4.3.2 固件升级说明

- 设备固件版本号只需要在系统启动过程中上报一次即可，不需要周期循环上报
- 根据版本号来判断设备端 OTA 是否升级成功
- 从 OTA 服务端控制台发起批量升级，设备升级操作记录状态是待升级

  实际升级以 OTA 系统接收到设备上报的升级进度开始，设备升级操作记录状态是升级中。
- 设备离线时，接收不到服务端推送的升级消息

  当设备上线后，主动通知服务端上线消息，OTA 服务端收到设备上线消息，验证该设备是否需要升级，如果需要，再次推送升级消息给设备，否则，不推送消息。

4.3.3 OTA 代码说明

初始化

OTA 模块的初始化依赖于 MQTT 连接，即先获得的 MQTT 客户端句柄 pclient。

```
h_ota = IOT_OTA_Init(PRODUCT_KEY, DEVICE_NAME, pclient);
if (NULL == h_ota) {
    rc = -1;
    printf("initialize OTA failed\n");
}
```

上报版本号

在 OTA 模块初始化之后，调用 IOT_OTA_ReportVersion 接口上报当前固件的版本号，升级成功后重启运行新固件，并使用该接口上报新固件版本号，云端与 OTA 升级任务的版本号对比成功后，提示 OTA 升级成功。

示例代码如下:

```
if (0 != IOT_OTA_ReportVersion(h_ota, "version2.0")) {
    rc = -1;
    printf("report OTA version failed\n");
}
```

下载固件

MQTT 通道获取到 OTA 固件下载的 URL 后，使用 HTTPS 下载固件，边下载边存储到 Flash OTA 分区。

- IOT_OTA_IsFetching() 接口：用于判断是否有固件可下载
- IOT_OTA_FetchYield() 接口：用于下载一个固件块
- IOT_OTA_IsFetchFinish() 接口：用于判断是否已下载完成

示例代码：

```
// 判断是否有固件可下载
if (IOT_OTA_IsFetching(h_ota)) {
    unsigned char buf_ota[OTA_BUF_LEN];
    uint32_t len, size_downloaded, size_file;
    do {
        // 循环下载固件
        len = IOT_OTA_FetchYield(h_ota, buf_ota, OTA_BUF_LEN, 1);
        if (len > 0) {
            // 写入到Flash等存储器中（边下载边存储）
        }
    } while (!IOT_OTA_IsFetchFinish(h_ota)); // 判断固件是否下载完毕
}
```

上报下载状态

使用 IOT_OTA_ReportProgress 接口上报固件下载进度。

```
if (percent - last_percent > 0) {
    IOT_OTA_ReportProgress(h_ota, percent, NULL);
}
IOT_MQTT_Yield(pclient, 100);
```

判断下载固件是否完整

固件下载完成后，使用 IOT_OTA_Ioctl 接口校验固件的完整性。

```
int32_t firmware_valid;
IOT_OTA_Ioctl(h_ota, IOT_OTAG_CHECK_FIRMWARE, &firmware_valid, 4);
if (0 == firmware_valid) {
    printf("The firmware is invalid\n");
} else {
    printf("The firmware is valid\n");
}
```

销毁 OTA 连接

使用 IOT_OTA_Deinit 销毁 OTA 连接并释放内存。

4.4 参考

- 以上内容引自阿里云物联网平台使用文档，详细内容请访问阿里云物联网平台文档中心进行查阅
- 更多的 API 使用说明请参考 API 使用文档
- 更多的示例代码请参考示例程序及示例使用说明

第 5 章

API 说明

ali-iotkit 是 RT-Thread 移植的用于连接阿里云 IoT 平台的软件包。基础 SDK 是阿里提供的 **iotkit-embedded C-SDK**。

这里引用阿里 **iotkit-embedded** API 使用说明，内容如下。

注：以下的 API 描述信息来自阿里云，更多详细内容请参阅 iotkit-embedded **wiki**。

5.1 必选 API

序号	函数名	说明
1	IOT_OpenLog	开始打印日志信息 (log), 接受一个 const char * 为入参, 表示模块名字
2	IOT_CloseLog	停止打印日志信息 (log), 入参为空
3	IOT_SetLogLevel	设置打印的日志等级, 接受入参从 1 到 5, 数字越大, 打印越详细
4	IOT_DumpMemoryStats	调试函数, 打印内存的使用统计情况, 入参为 1-5, 数字越大, 打印越详细

5.2 MQTT 功能相关 API

序号	函数名	说明
1	IOT_SetupConnInfo	MQTT 连接前的准备, 基于`DeviceName + DeviceSecret + ProductKey`产生 MQTT 连接的用户名和密码等
2	IOT_SetupConnInfoSecure	MQTT 连接前的准备, 基于`ID2 + DeviceSecret + ProductKey`产生 MQTT 连接的用户名和密码等,ID2 模式启用
3	IOT_MQTT_CheckStateNormal	MQTT 连接后, 调用此函数检查长连接是否正常
4	IOT_MQTT_Construct	MQTT 实例的构造函数, 入参为`iotx_mqtt_param_t`结构体, 连接 MQTT 服务器, 并返回被创建句柄
5	IOT_MQTT_ConstructSecure	MQTT 实例的构造函数, 入参为`iotx_mqtt_param_t`结构体, 连接 MQTT 服务器, 并返回被创建句柄，ID2 模式启用
6	IOT_MQTT_Destroy	MQTT 实例的摧毁函数, 入参为`IOT_MQTT_Construct()`创建的句柄
7	IOT_MQTT_Publish	MQTT 会话阶段, 组织一个完整的`MQTT Publish`报文, 向服务端发送消息发布报文
8	IOT_MQTT_Subscribe	MQTT 会话阶段, 组织一个完整的`MQTT Subscribe`报文, 向服务端发送订阅请求
9	IOT_MQTT_Unsubscribe	MQTT 会话阶段, 组织一个完整的`MQTT UnSubscribe`报文, 向服务端发送取消订阅请求
10	IOT_MQTT_Yield	MQTT 会话阶段, MQTT 主循环函数, 内含了心跳的维持, 服务器下行报文的收取等

5.3 CoAP 功能相关 API

序号	函数名	说明
1	IOT_CoAP_Init	CoAP 实例的构造函数, 入参为`iotx_coap_config_t`结构体, 返回创建的 CoAP 会话句柄
2	IOT_CoAP_Deinit	CoAP 实例的摧毁函数, 入参为`IOT_CoAP_Init()`所创建的句柄
3	IOT_CoAP_DeviceNameAuth	基于控制台申请的`DeviceName`, `DeviceSecret`, `ProductKey`做设备认证
4	IOT_CoAP_GetMessageCode	CoAP 会话阶段, 从服务器的`CoAP Response`报文中获取`Respond Code`
5	IOT_CoAP_GetMessagePayload	CoAP 会话阶段, 从服务器的`CoAP Response`报文中获取报文负载
6	IOT_CoAP_SendMessage	CoAP 会话阶段, 连接已成功建立后调用, 组织一个完整的 CoAP 报文向服务器发送
7	IOT_CoAP_Yield	CoAP 会话阶段, 连接已成功建立后调用, 检查和收取服务器对`CoAP Request`的回复报文

5.4 HTTP 功能相关 API

序号	函数名	说明
1	IOT_HTTP_Init	Https 实例的构造函数, 创建一个 HTTP 会话的句柄并返回
2	IOT_HTTP_DeInit	Https 实例的摧毁函数, 销毁所有相关的数据结构
3	IOT_HTTP_DeviceNameAuth	基于控制台申请的`DeviceName`, `DeviceSecret`, `ProductKey`做设备认证
4	IOT_HTTP_SendMessage	Https 会话阶段, 组织一个完整的 HTTP 报文向服务器发送, 并同步获取 HTTP 回复报文

序号	函数名	说明
5	IOT_HTTP_Disconnect	Https 会话阶段, 关闭 HTTP 层面的连接, 但是仍然保持 TLS 层面的连接

5.5 OTA 功能相关 API

序号	函数名	说明
1	IOT_OTA_Init	OTA 实例的构造函数, 创建一个 OTA 会话的句柄并返回
2	IOT_OTA_Deinit	OTA 实例的摧毁函数, 销毁所有相关的数据结构
3	IOT_OTA_Ioctl	OTA 实例的输入输出函数, 根据不同的命令字可以设置 OTA 会话的属性, 或者获取 OTA 会话的状态
4	IOT_OTA_GetLastError	OTA 会话阶段, 若某个`IOT_OTA_*()`函数返回错误, 调用此接口可获得最近一次的详细错误码
5	IOT_OTA_ReportVersion	OTA 会话阶段, 向服务端汇报当前的固件版本号
6	IOT_OTA_FetchYield	OTA 下载阶段, 在指定的`timeout`时间内, 从固件服务器下载一段固件内容, 保存在入参 buffer 中
7	IOT_OTA_IsFetchFinish	OTA 下载阶段, 判断迭代调用`IOT_OTA_FetchYield()`是否已经下载完所有的固件内容
8	IOT_OTA_IsFetching	OTA 下载阶段, 判断固件下载是否仍在进行中, 尚未完成全部固件内容的下载
9	IOT_OTA_ReportProgress	可选 API, OTA 下载阶段, 调用此函数向服务端汇报已经下载了全部固件内容的百分之多少
10	IOT_OTA_RequestImage	可选 API，向服务端请求固件下载
11	IOT_OTA_GetConfig	可选 API，向服务端请求远程配置

5.6 云端连接 Cloud Connection 功能相关 API

序号	函数名	说明
1	IOT_Cloud_Connection_Init	云端连接实例的构造函数, 入参为`iotx_cloud_connection_param_pt`结构体, 返回创建的云端连接会话句柄
2	IOT_Cloud_Connection_Deinit	云端连接实例的摧毁函数, 入参为`IOT_Cloud_Connection_Init()`所创建的句柄
3	IOT_Cloud_Connection_Send_Me	发送数据给云端
4	IOT_Cloud_Connection_Yield	云端连接成功建立后，收取服务器发送的报文

5.7 CMP 功能相关 API

序号	函数名	说明
1	IOT_CMP_Init	CMP 实例的构造函数, 入参为`iotx_cmp_init_param_pt`结构体，只存在一个 CMP 实例
2	IOT_CMP_Register	通过 CMP 订阅服务
3	IOT_CMP_Unregister	通过 CMP 取消服务订阅
4	IOT_CMP_Send	通过 CMP 发送数据，可以送给云端，也可以送给本地设备
5	IOT_CMP_Send_Sync	通过 CMP 同步发送数据，暂不支持
6	IOT_CMP_Yield	通过 CMP 接收数据，单线程情况下才支持
7	IOT_CMP_Deinit	CMP 示例的摧毁函数
8	IOT_CMP_OTA_Start	初始化 ota 功能，上报版本
9	IOT_CMP_OTA_Set_Callback	设置 OTA 回调函数
10	IOT_CMP_OTA_Get_Config	获取远程配置
11	IOT_CMP_OTA_Request_Image	获取固件

序号	函数名	说明
12	IOT_CMP_OTA_Yield	通过 CMP 完成 OTA 功能

5.8 设备影子相关 (可选功能) API

序号	函数名	说明
1	IOT_Shadow_Construct	建立一个设备影子的 MQTT 连接, 并返回被创建的会话句柄
2	IOT_Shadow_Destroy	摧毁一个设备影子的 MQTT 连接, 销毁所有相关的数据结构, 释放内存, 断开连接
3	IOT_Shadow_Pull	把服务器端被缓存的 JSON 数据下拉到本地, 更新本地的数据属性
4	IOT_Shadow_Push	把本地的数据属性上推到服务器缓存的 JSON 数据, 更新服务端的数据属性
5	IOT_Shadow_Push_Async	和`IOT_Shadow_Push()`接口类似, 但是异步的, 上推后便返回, 不等待服务端回应
6	IOT_Shadow_PushFormat_Add	向已创建的数据类型格式中增添成员属性
7	IOT_Shadow_PushFormat_Finaliz	完成一个数据类型格式的构造过程
8	IOT_Shadow_PushFormat_Init	开始一个数据类型格式的构造过程
9	IOT_Shadow_RegisterAttribute	创建一个数据类型注册到服务端, 注册时需要`*PushFormat*()`接口创建的数据类型格式
10	IOT_Shadow_DeleteAttribute	删除一个已被成功注册的数据属性
11	IOT_Shadow_Yield	MQTT 的主循环函数, 调用后接受服务端的下推消息, 更新本地的数据属性

5.9 主子设备相关 (可选功能) API

序号	函数名	说明
1	IOT_Gateway_Construct	建立一个主设备，建立 MQTT 连接, 并返回被创建的会话句柄
2	IOT_Gateway_Destroy	摧毁一个主设备的 MQTT 连接, 销毁所有相关的数据结构, 释放内存, 断开连接
3	IOT_Subdevice_Login	子设备上线，通知云端建立子设备 session
4	IOT_Subdevice_Logout	子设备下线，销毁云端建立子设备 session 及所有相关的数据结构, 释放内存
5	IOT_Gateway_Yield	MQTT 的主循环函数, 调用后接受服务端的下推消息
6	IOT_Gateway_Subscribe	通过 MQTT 连接向服务端发送订阅请求
7	IOT_Gateway_Unsubscribe	通过 MQTT 连接向服务端发送取消订阅请求
8	IOT_Gateway_Publish	通过 MQTT 连接服务端发送消息发布报文
9	IOT_Gateway_RRPC_Register	注册设备的 RRPC 回调函数，接收云端发起的 RRPC 请求
10	IOT_Gateway_RRPC_Response	对云端的 RRPC 请求进行应答
11	IOT_Gateway_Generate_Message_	生成消息 id
12	IOT_Gateway_Get_TOPO	向 topo/get topic 发送包并等待回复（TOPIC_GET_REPLY 回复）
13	IOT_Gateway_Get_Config	向 conifg/get topic 发送包并等待回复（TOPIC_CONFIG_REPLY 回复）
14	IOT_Gateway_Publish_Found_List	发现设备列表上报

5.10 linkkit 功能相关 API

序号	函数名	说明
1	linkkit_start	启动 linkkit 服务，与云端建立连接并安装回调函数

序号	函数名	说明
2	linkkit_end	停止 linkkit 服务，与云端断开连接并回收资源
3	linkkit_dispatch	事件分发函数, 触发 linkkit_start 安装的回调
4	linkkit_yield	linkkit 主循环函数，内含了心跳的维持, 服务器下行报文的收取等; 如果允许多线程，请不要调用此函数
5	linkkit_set_value	根据 identifier 设置物对象的 TSL 属性，如果标识符为 struct 类型、event output 类型或者 service output 类型，使用点'.' 分隔字段；例如 “identifier1.identifier2” 指向特定的项
6	linkkit_get_value	根据 identifier 获取物对象的 TSL 属性
7	linkkit_set_tsl	从本地读取 TSL 文件, 生成物的对象并添加到 linkkit 中
8	linkkit_answer_service	对云端服务请求进行回应
9	linkkit_invoke_raw_service	向云端发送裸数据
10	linkkit_trigger_event	上报设备事件到云端
11	linkkit_fota_init	初始化 OTA-fota 服务，并安装回调函数 (需编译设置宏 SERVICE_OTA_ENABLED)
12	linkkit_invoke_fota_service	执行 fota 服务
13	linkkit_fota_init	初始化 OTA-cota 服务，并安装回调函数 (需编译设置宏 SERVICE_OTA_ENABLED SERVICE_COTA_ENABLED)
14	linkkit_invoke_cota_get_config	设备请求远程配置
15	linkkit_invoke_cota_service	执行 cota 服务