
RT-THREAD ALI-IOTKIT User Manual

RT-THREAD Documentation Center

Copyright ©2019 Shanghai Ruiside Electronic Technology Co., Ltd.



WWW.RT-THREAD.ORG

Friday 28th September, 2018

Versions and Revisions

Date	Version	Author	Note
2018-07-21	v0.1	MurphyZhao	Initial version

[Table of contents](#)

Versions and Revisions	i
Table of contents	ii
1 Software Package Introduction	1
1.1 Software framework diagram.	2
1.2 Package directory structure.	2
1.2.1 iotkit-embedded package directory structure.	3
1.3 Features.	4
2 Sample Program	6
2.1 LinkDevelop platform.	6
2.1.1 Preparation.	6
2.1.2 MQTT Example.	11
2.1.3 OTA example.	15
2.2 LinkPlatform . . .	20
2.2.1 Preparation.	20
2.2.2 MQTT Example.	twenty four
2.2.3 OTA example.	27
2.3 Notes.	30
2.4 Frequently Asked Questions.	31
3 Working Principle	32
3.1 MQTT data interaction process.	33
3.2 OTA data interaction process.	33

4 Usage Guide	35
4.1 MQTT connection.	35
4.1.1 Features.	35
4.1.2 Security level.	35
4.1.3 Connecting domain name.	36
4.1.4 Topic Specifications	36
4.1.5 Establishing an MQTT connection	36
4.1.6 Subscribe to Topic.	37
4.1.7 Publish messages.	37
4.1.8 Cancellation of Subscription.	38
4.1.9 Downlink data reception.	38
4.1.10 Destroy the MQTT connection.	38
4.1.11 Check the connection status.	38
4.1.12 MQTT keep-alive.	38
4.2 CoAP connection.	39
4.2.1 CoAP Conventions	39
4.2.2 Application scenarios.	39
4.2.3 Establish a connection.	40
4.2.4 Sending and receiving data.	40
4.2.5 Downlink data reception.	41
4.2.6 Destroy the CoAP connection.	41
4.3 OTA upgrade.	41
4.3.1 Firmware Upgrade Topic	41
4.3.2 Firmware upgrade instructions.	42
4.3.3 OTA code description.	42
4.4 References.	44
5 API Description	45
5.1 Required API	45
5.2 MQTT function related API	45
5.3 CoAP function related API	46

- 5.4 HTTP function related API 47
- 5.5 OTA function related API 48
- 5.6 Cloud Connection API. 49
- 5.7 CMP function related API 49
- 5.8 Device shadow related (optional function) API 50
- 5.9 Master and slave device related (optional) API. 50
- 5.10 Linkkit function related API 51

Chapter 1

Software Package Introduction

ali-iotkit is a software package ported from RT-Thread for connecting to the Alibaba Cloud IoT platform. The basic SDK is [the iotkit-embedded C-SDK provided by Alibaba](#).

The IoT Suite provides the following capabilities:

- Quick access to embedded devices (device-side SDK) •
- Device management • Device and data information security • Bridge to other Alibaba Cloud products to store/compute device data



Figure 1.1: Alibaba Cloud IoT Platform Scenario Architecture Diagram

In the IoT platform scenario architecture diagram, the IoT device SDK on the left is the part that connects the embedded device to Alibaba Cloud.

1.1 Software Framework Diagram

In order to facilitate the connection between devices and the cloud, iotkit SDK encapsulates a variety of connection protocols, such as MQTT, CoAP, HTTP, and TLS, and abstracts the hardware platform to make it more flexible without being restricted by specific hardware platforms. In terms of code architecture, iotkit SDK is divided into three layers, as shown in the following figure:

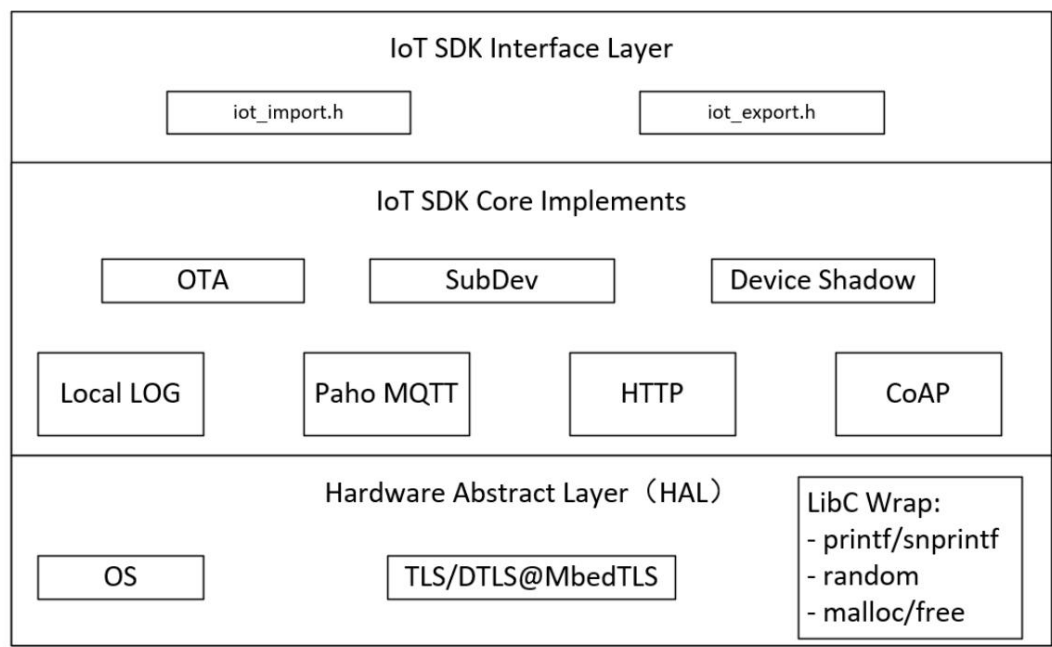


Figure 1.2: iotkit SDK Software framework diagram

- The bottom layer is called the Hardware Abstract Layer, also known as the HAL layer (Hardware Abstract Layer)

Abstracts the operating system's support functions for the SDK on different embedded target boards, including network sending and receiving, TLS/DTLS channel establishment and reading and writing, memory application and mutex locking and unlocking, etc.

- The middle layer is called the SDK Core Implements layer

The core implementation part of the IoT platform C-SDK, which completes the functional encapsulation of MQTT/CoAP channels based on the HAL layer interface, including MQTT connection establishment, message sending and receiving, CoAP connection establishment, message sending and receiving, OTA firmware status query and OTA firmware download, etc.

The encapsulation of the middle layer means that users do not need to worry about the internal implementation logic and can apply it without modification.

- The top layer is called the SDK interface declaration layer (IoT SDK Interface Layer)

The top layer provides APIs for applications, and users use the APIs in this layer to complete specific business logic.

1.2 Software package directory structure

The **ports** directory contains the porting files involved in porting the iotkit-embedded software package to RT-Thread, and is rebuilt using scons.

The **iotkit-embedded** package is the source code of Alibaba IoT platform C-SDK, which contains the necessary

Software package.

```

ali-iotkit

| README.md |                                // Software package instructions
SConscript                                     //RT-Thread default build script
+---docs
| +---figures || api.md ||                  // Document using images
introduction.md || ||                       // API usage instructions
principle.md || README.md ||               // Software package details
        LICENSE                             // License file
samples.md || user-guide.md |              // Implementation principle
+---version.md                             // Document structure description
                                           // Package Example
                                           // Instructions for use
                                           // Version description
+---ports | +---                           // Migrate files
rtthread | +---ssl +---                    // OS related porting files
samples                                    // MbedTLS related porting files

| +---mqtt | +---ota                        // Sample program for connecting MQTT channel to Alibaba Cloud
                                           // Alibaba Cloud OTA function demonstration program
+---iotkit-embedded                        //iotkit source code

```

1.2.1 iotkit-embedded software package directory structure

The iotkit-embedded package is the source code of Alibaba IoT platform C-SDK, which is unmodified and includes the necessary software package for cloud IoT. After RT-Thread is ported, the default Makefile in iotkit-embedded is not used. Build scripts, but rebuild them using scons.

The directory structure of the iotkit-embedded software package is as follows:

```

+-- LICENSE                               : Software License, Apache-2.0 Software License
+-- make.settings                         : Function tailoring configuration , such as MQTT|CoAP, or tailoring such as OTA|Shadow
+-- README.md                            : Quick Start Guide
+-- sample || ||                         : Routine directory , demonstrating the use of communication modules and service modules
|   +-- mqtt                             : Demonstrates how to use the API of the communication module MQTT
|   +-- coap +--                         : Demonstrates how to use the API of the communication module CoAP
|   device-shadow : Demonstrates how to use the API of the service module DeviceShadow
|   +-- http                             : Demonstrates how to use the communication module HTTP API
|   +-- ota                             : Demonstrate how to use the service module OTA API
+-- src

```


<code>++ sdk-impl package</code>	: SDK interface layer, providing overall header files and some API interfaces
<code>++ sdk-tests</code>	: Unit tests for the SDK
<code>++ mqtt</code>	: Communication module, realize access via MQTT protocol
<code>++ coap</code>	: Communication module, to achieve access via CoAP protocol
<code>++ http</code>	: Communication module, to achieve access via HTTP protocol
<code>++ ota</code>	: Service module, implement firmware download based on MQTT CoAP+HTTP+TLS
<code>aisle</code>	
<code>++ shadow</code>	: Service module, implement device shadow
<code>++ platform ++</code>	: Hardware platform abstraction layer, needs to be ported and adapted
<code>import library</code>	: External input directory, storing header files / binary files provided by chip/module manufacturers
<code>++ configs pruning,</code> <code>etc.</code>	: Hardware platform compilation configuration, such as cross-compilation tool chain settings, functional modules
<code>++ scripts ++</code>	: The compilation process will reference external scripts, users do not need to pay attention to
<code>packages ++ log</code>	: External software modules referenced by SDK, users do not need to pay attention to them
	: Basic module, realizes running log
<code>++ system</code> <code>ID, etc.</code>	: Basic module, saves global information, such as TLS root certificate, device identification
<code>++ tls</code>	: Basic module, implements TLS/DTLS, from tailored open source software
<code>mbedtls</code>	
<code>++ utils</code> <code>Time</code>	: Basic module, implements tool functions, such as SHA1 summary calculation, NTP pair

1.3 Features

• Different network access

Provides device access solutions for different networks, such as 2/3/4G, NB-IoT, LoRa, etc., to solve heterogeneous enterprise networks

The pain points of device access management.

• Different protocol access

Provides device SDKs for multiple protocols, such as MQTT, CoAP, HTTP, etc., which can meet the needs of devices

Long connections ensure real-time requirements and can also meet the needs of devices that require short connections to reduce power consumption.

• Two-way communication

Providing uplink and downlink channels between devices and the cloud, it can stably and reliably support scenarios where devices report and send commands to devices.

• Device Shadow

Provides a device shadow cache mechanism to decouple devices from applications and solve communication problems when wireless networks are unstable.

Reliable pain points.

• Device authentication

Provides a one-machine-one-password device authentication mechanism to reduce the security risk of device hacking.

- Secure transmission

Provides TLS standard data transmission channel to ensure data confidentiality and integrity.

chapter 2

Sample Program

The **ali-iotkit** software package also supports Alibaba's existing **LinkDevelop** and **LinkPlatform** platform.

This article demonstrates sample programs for these two platforms respectively. Users can choose to use one of them according to their needs. one of.

2.1 LinkDevelop Platform

The LinkDevelop platform uses RGB_LED as an example to introduce how to perform two-way communication between the device and the cloud.

2.1.1 Preparation

- Register on [the LinkDevelop platform](#)

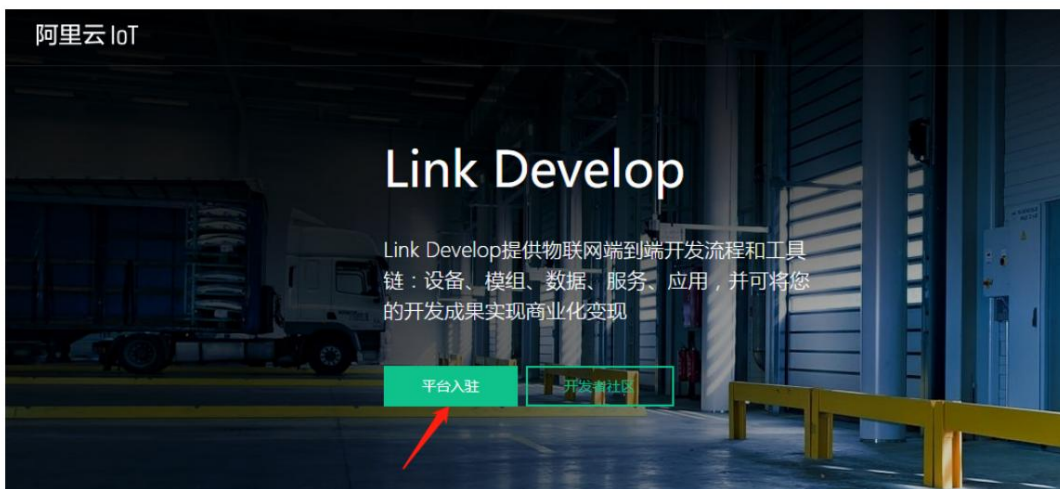


Figure 2.1: register *LinkDevelop* platform

- New Project



Figure 2.2: New Project

- New Products

When adding a new product, select the data format as needed. Here we use the **Alink** data format demonstration and select WiFi communication method.

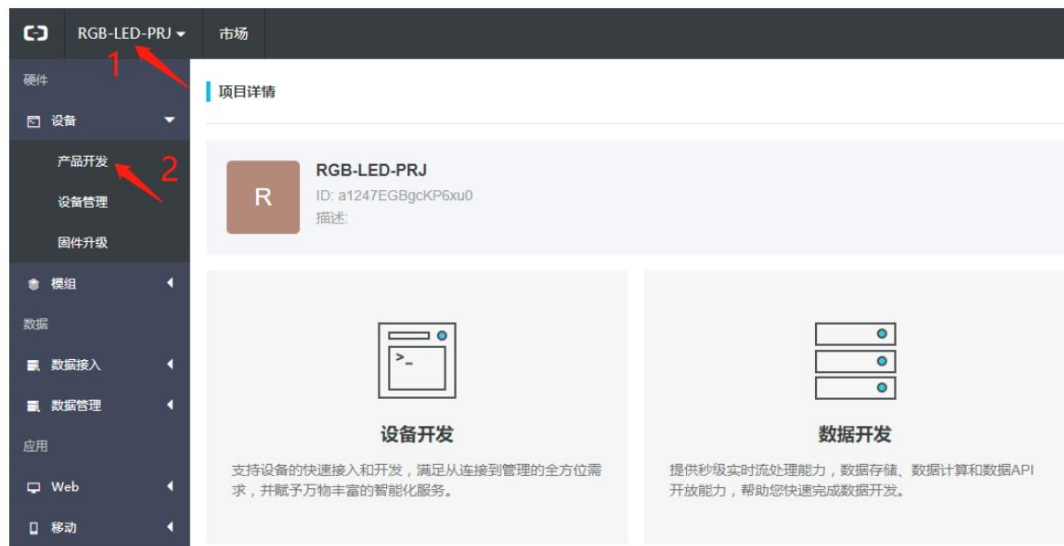


Figure 2.3: Open the product development page

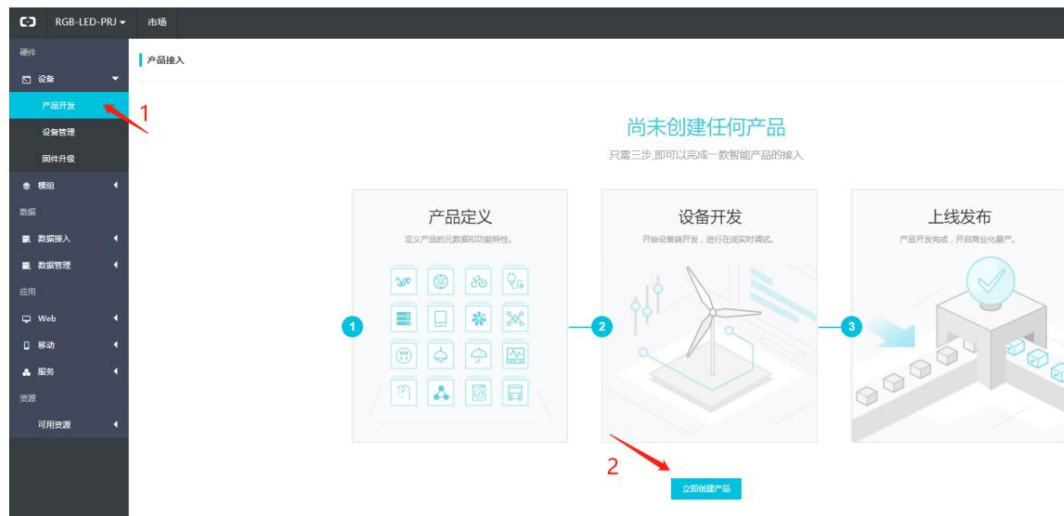


Figure 2.4: Create a product



Figure 2.5: Write product information

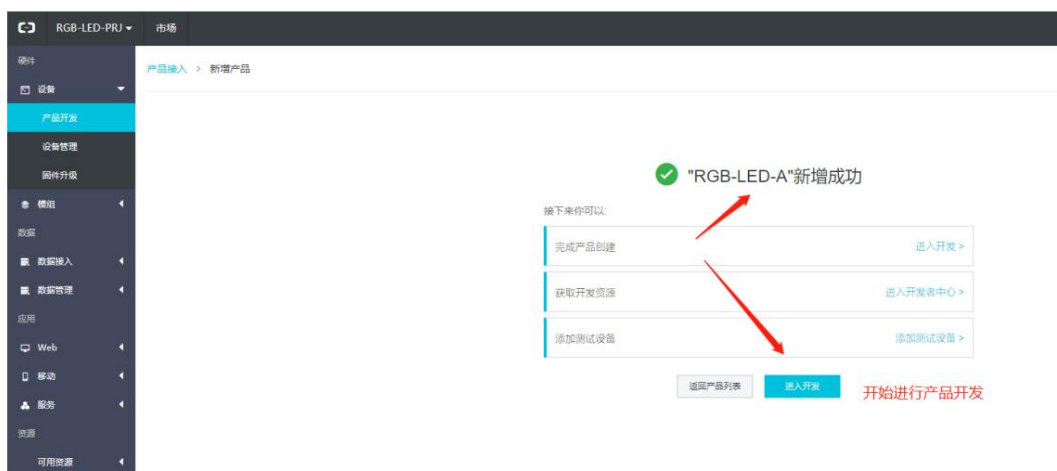


Figure 2.6: Enter product development

- add feature

Add RGB color adjustment function to the RGB LED demonstration product, as shown below:

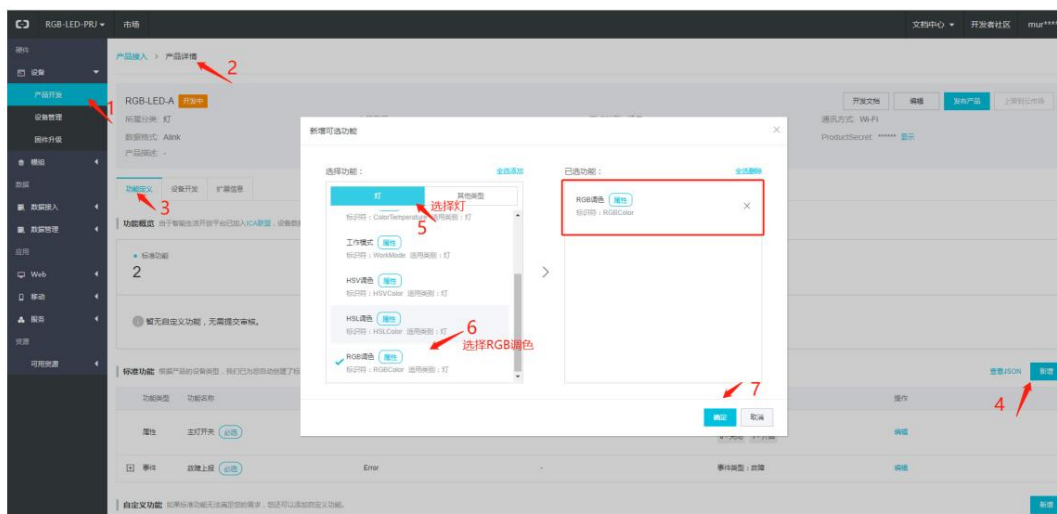


Figure 2.7: Add product features

- Adding a device

After creating a product, click View to enter the product details page, click Device Development, and add a new debugging device.

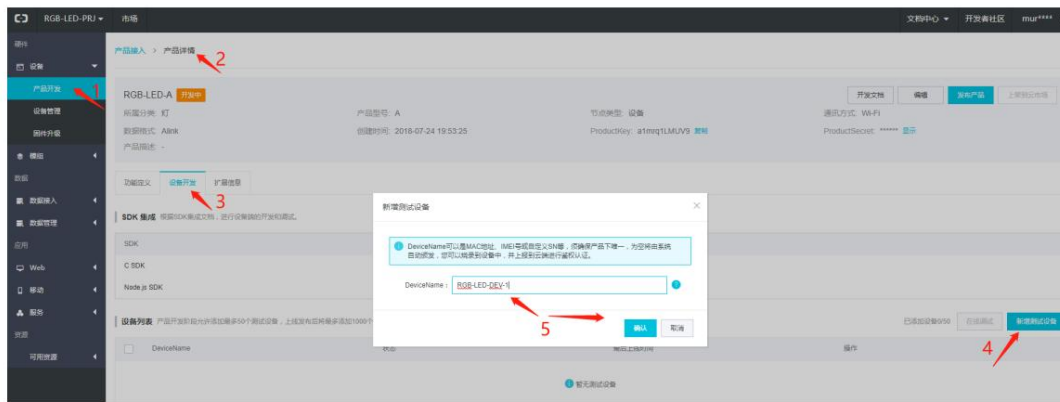


Figure 2.8: Add a device

After successfully creating the device, you can obtain the triplet required for device activation (ProductKey, DeviceName, DeviceSecret), which needs to be configured in the device SDK using **menuconfig** later.

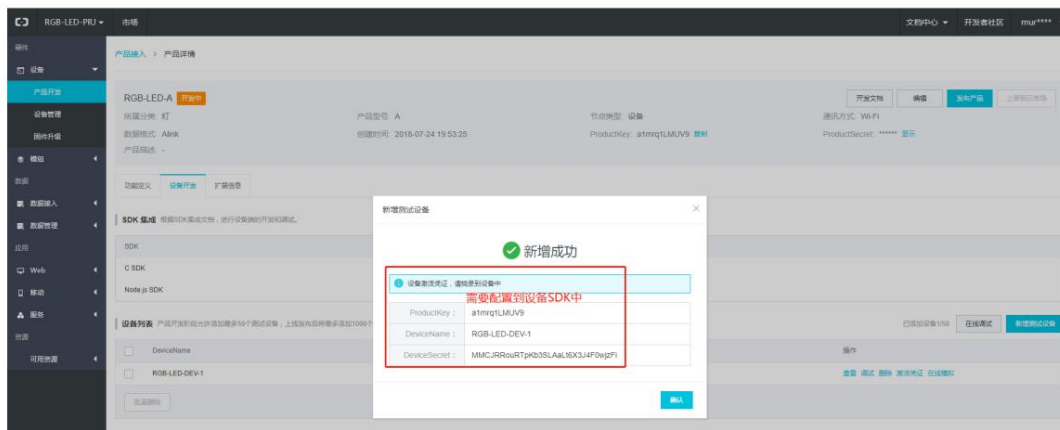


Figure 2.9: Obtain device activation certificate

- Get the software package

Open the ENV tool provided by RT-Thread and use **menuconfig** to configure the software package.

–Configure iotkit package

Configure and enable the iotkit software package and fill in the device activation credentials.

In **menuconfig**, select LinkDevelop for Alibaba Cloud platform and **MQTT** for **OTA channel** (with

Taking MQTT as an example), the detailed configuration is as follows:

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
(Ghuiyd9nmGowdZzjPqFtxhm3WUHEbIII) 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) ---->
```

– Increase `MBEDTLS` frame size

During the Alibaba TLS authentication process, the data packet is large, so the TLS frame size needs to be increased. During OTA, the size needs to be at least **8K**.

Open the ENV tool provided by RT-Thread and use `menuconfig` to configure the **TLS** frame size.

```
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
```

– Download the package using the `pkgs --update` command

2.1.2 MQTT Example

This MQTT sample program uses RGB-LED as an example to demonstrate how to use MQTT + TLS/SSL on the device

The channel establishes two-way communication with the Alibaba Cloud platform.

Sample Files

Sample program path	Verification Platform	illustrate
<code>samples/mqtt/mqtt-example.c</code>	LinkDevelop, LinkPlatform	Device and cloud two-way communication routine based on MQTT channel

Command List

In the example, use the MSH command to start the MQTT example. The command is as follows:

Order	illustrate
ali_mqtt_test start	Start the MQTT example
ali_mqtt_test pub open	Turn on the light and sync the light status to the cloud
ali_mqtt_test pub close	Turn off the lights and synchronize the light-off status to the cloud
ali_mqtt_test stop	Stop MQTT Example

Start MQTT

Use the `ali_mqtt_test start` command to start the MQTT example. After success, the device log shows that the subscription is successful.

The device log is as follows:

```
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/ a1HETIEuvri/RGB-LED-
DEV-1/thing/service/property/set! [inf] iotx_mc_subscribe(1388): mqtt
subscribe success,topic = / sys/
a1HETIEuvri/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!
```

Device release message

Use the `ali_mqtt_test pub open` command to send the LED status to the cloud. After success, the device log will display the success code.

200.

The device log is as follows:

```

msh />ali_mqtt_test pub open
...

[dbg] iotx_mc_cycle(1277): PUBLISH
[dbg] iotx_mc_handle_rcv_PUBLISH(1091): [dbg] Packet Ident : 00000000
iotx_mc_handle_rcv_PUBLISH(1092): [dbg] Topic Length : 57
iotx_mc_handle_rcv_PUBLISH(1096): Topic Name : /sys/
a1HETIEuvri/RGB-LED-DEV-1/thing/service/property/set
[dbg] iotx_mc_handle_rcv_PUBLISH(1099): [dbg] Payload Len/Room : 100 / 962
iotx_mc_handle_rcv_PUBLISH(1100): [dbg] Receive Buflen : 1024
iotx_mc_handle_rcv_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/a1HETIEuvri/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 ::----

```

View published messages in the cloud

In the operating status of the device details, you can view the message content reported to the cloud by the device.

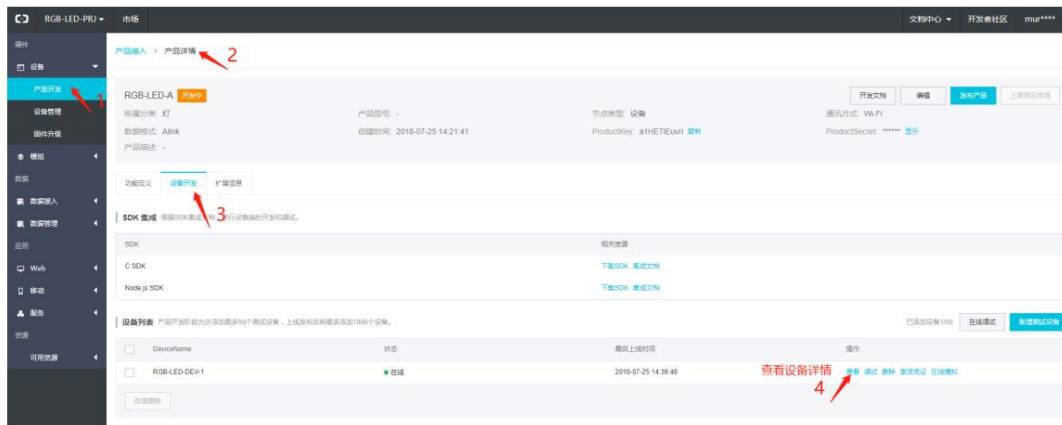


Figure 2.10: View device details

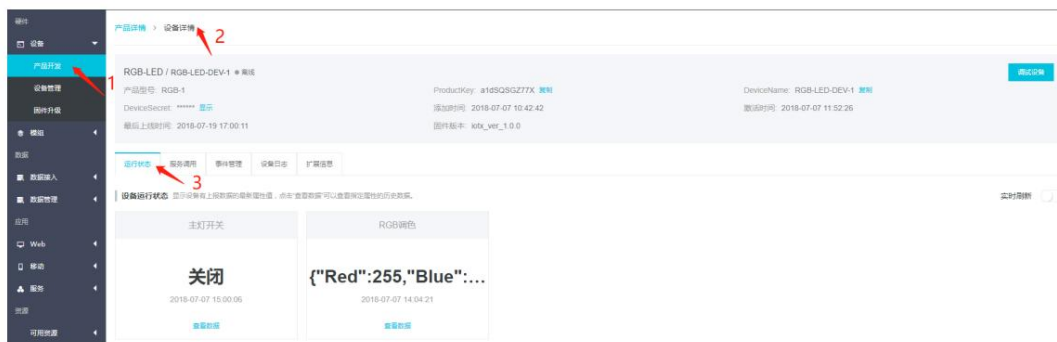


Figure 2.11: Check the device operation status

Push messages from the cloud to devices

Use the cloud-based debugging console to push messages to the device.

- Open the debug console

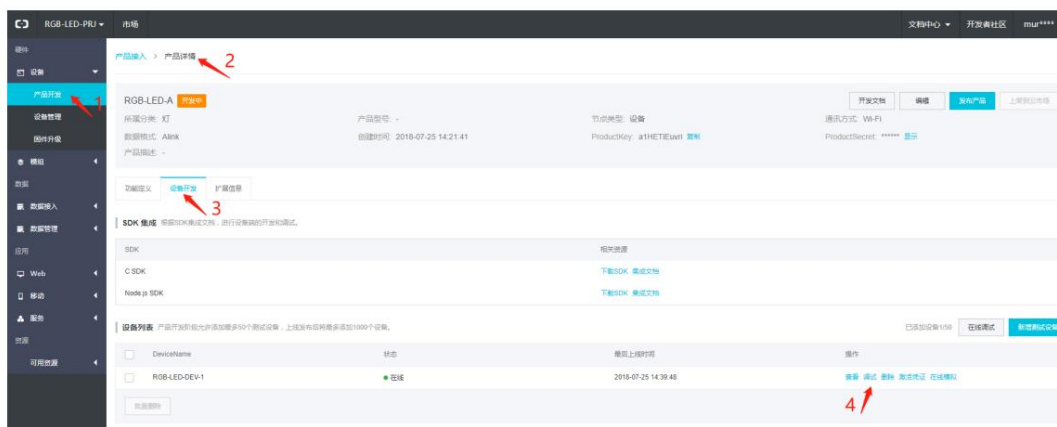


Figure 2.12: Open the debug console

- Send debug commands



Figure 2.13: Online debugging page

View device subscription logs

After sending commands using the debug console, the device can receive the commands and the log is as follows:

```
[dbg] iotx_mc_handle_rcv_PUBLISH(1091): [dbg] Packet Ident : 00000000
iotx_mc_handle_rcv_PUBLISH(1092): [dbg] Topic Length : 52
iotx_mc_handle_rcv_PUBLISH(1096): Topic Name : /sys/
    a1Ayv8xholl/RGB-DEV1/thing/service/property/set [dbg]
iotx_mc_handle_rcv_PUBLISH(1099): [dbg] Payload Len/Room : 100 / 967
iotx_mc_handle_rcv_PUBLISH(1100): [dbg] Receive Buflen : 1024
iotx_mc_handle_rcv_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/
a1Ayv8xholl/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 :: ----
```

Exit **MQTT** Example

Use the **ali_mqtt_test stop** command to exit the MQTT example. The device log is as follows:

```
msh />ali_mqtt_test stop [inf]
iotx_mc_unsubscribe(1423): mqtt unsubscribe success,topic = /sys/ a1HETIEuvri/RGB-LED-DEV-1/
    thing/event/property/post_reply! [inf] iotx_mc_unsubscribe(1423): mqtt unsubscribe
success,topic = /sys/
    a1HETIEuvri/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 Example

Firmware upgrade supports remote over-the-air upgrade of the device's firmware, enabling remote maintenance, function upgrades, problem repairs, and other scenarios. You can specify a new firmware for a product, verify the firmware, start batch upgrades after verification, and view the upgrade results in the firmware details.

Sample Files

Sample program path	Verification Platform	illustrate
samples/ota/ ota_mqtt-example.c	LinkDevelop, LinkPlatform	Devices based on MQTT channels OTA routines

Command List

In the example, use the MSH command to start the OTA example. The command is as follows:

Order	illustrate
ali_ota_test start	Launch OTA Example
ali_ota_test stop	Manually exit OTA example

Running the OTA Example

Use the `ali_ota_test start` command to start the OTA routine, and then wait for the cloud to send OTA instructions.

The device log is as follows:

```
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/a1HETIEuvri/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....
```

New firmware

Here, users are required to upload a bin-type test firmware. Any bin firmware will do. The demo program only performs

The firmware is downloaded and verified, but not written to Flash, so the firmware will not actually be moved and upgraded.



Figure 2.14: LinkDevelop New Platform OTA firmware

Verify the firmware

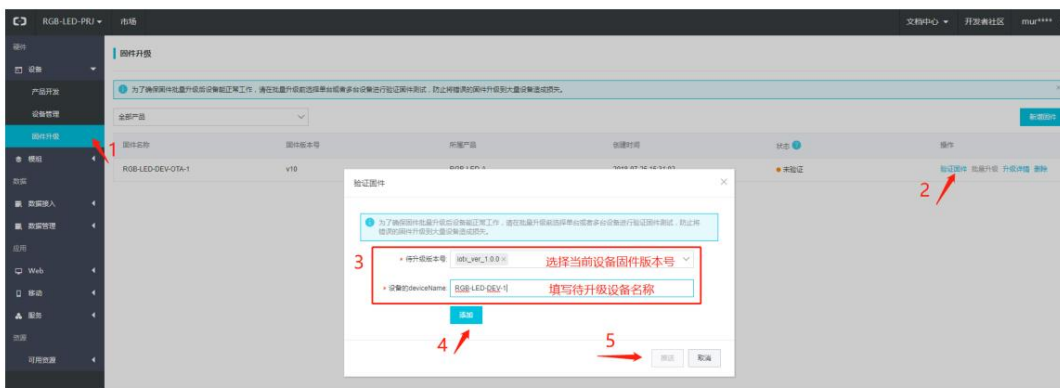


Figure 2.15: LinkDevelop Platform Verification OTA firmware

Device logs

After the push is successful, the device starts downloading the firmware. After the download is complete, the firmware integrity check is automatically performed.

The log is as follows:

```
...
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
```

Cloud upgrade progress display

During the device upgrade process, the cloud will display the progress of the device downloading firmware. When the firmware download is complete and the firmware verification is successful, the device

The SDK reports the new version number to the cloud, and the cloud will show that the upgrade is successful, as shown in the following figure:

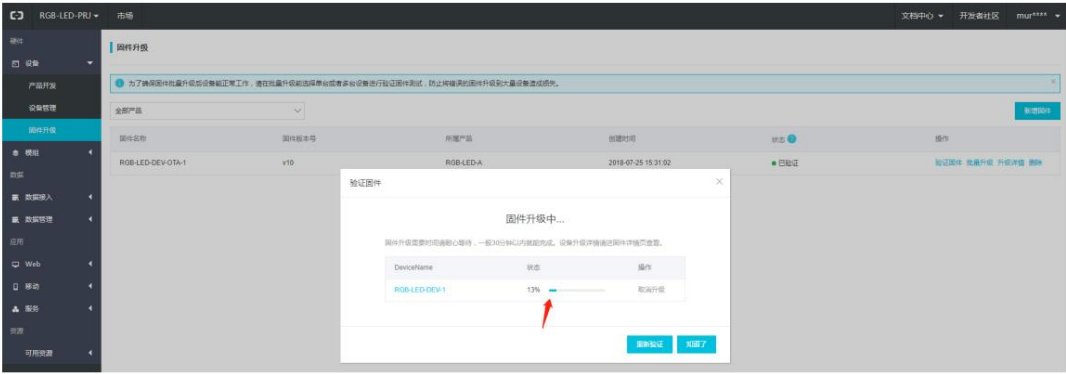


Figure 2.16: Upgrade Progress

After the upgrade progress reaches 100%, run the `ali_ota_test start` command again to upload the latest version number to the cloud. After the version number matches successfully, the cloud displays that the upgrade is successful, as shown in the following figure:

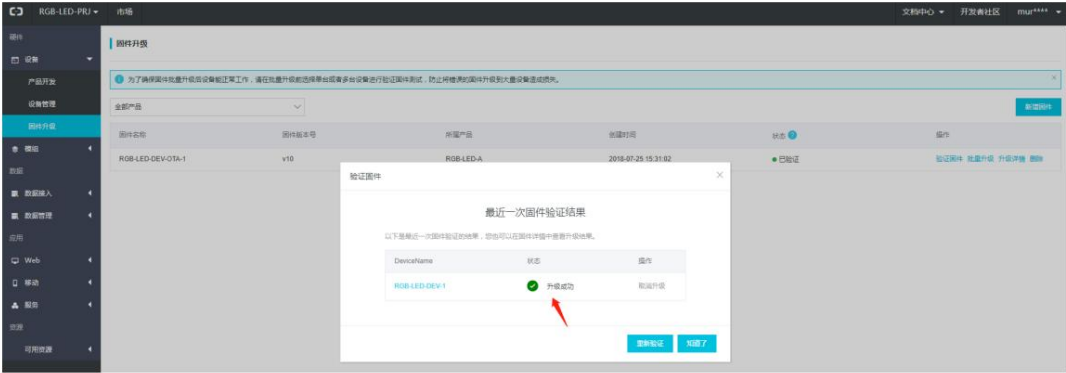


Figure 2.17: update succeeded

Exit OTA routine

The OTA routine will be automatically exited if the upgrade is successful or failed. If you need to exit the OTA routine manually, use

`ali_ota_test stop` command.

```
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

2.2.1 Preparation

- Register [LinkPlatform](#)



Figure 2.18: register *LinkPlatform* platform

- Create products



Figure 2.19: Create a product

- Adding a device

Under the Device Management menu, add a new test device and click View to enter the device details page.

After successfully creating the device, you can obtain the triplet required for device activation (ProductKey, DeviceName,

DeviceSecret), which needs to be configured in the device SDK using `menuconfig` later.

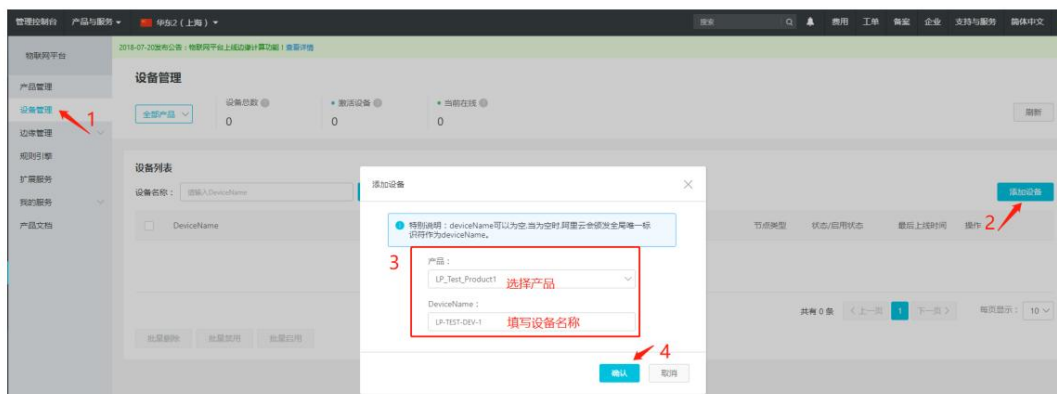


Figure 2.20: Add a device



Figure 2.21: Device activation certificate

- View the message topic list

Go to the device details page, and then in the **Topic** List option, view the default topic list assigned to the created device. and Topic permissions.

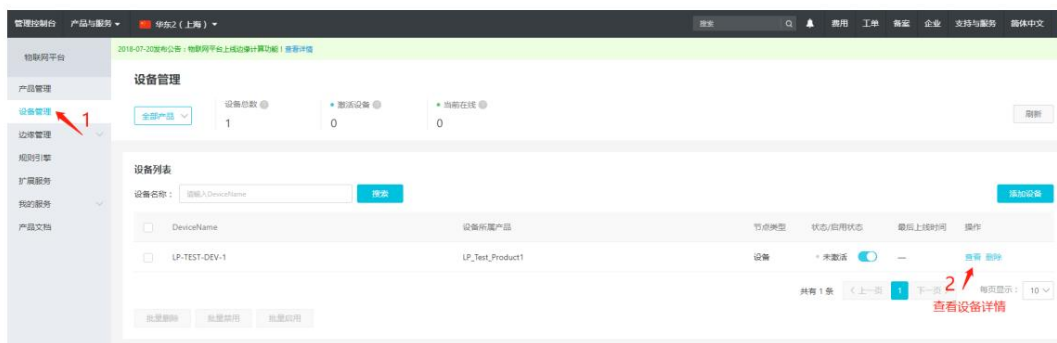


Figure 2.22: Go to the device details page

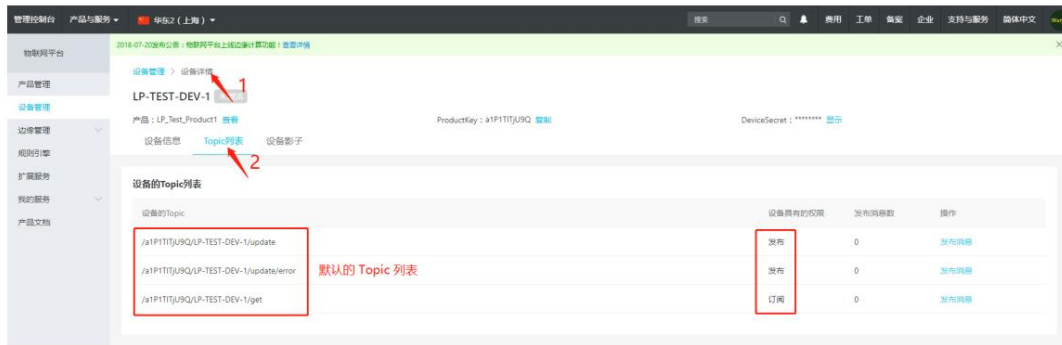


Figure 2.23: Check MQTT Topic List

• Custom Topics

The MQTT sample program uses a topic named **data**. The topic permissions are publish and subscribe, so You need to customize a **data** Topic, as shown below:



Figure 2.24: customize Topic

• Activate firmware upgrade service



Figure 2.25: Activate firmware upgrade service



Figure 2.26: Activate firmware upgrade service

- Get the software package

Open the ENV tool provided by RT-Thread and use **menuconfig** to configure the software package.

–Configure iotkit package

Configure and enable the iotkit software package and fill in the device

activation credentials. In **menuconfig**, select Alibaba Cloud platform as **LinkPlatform** and **OTA channel** as **MQTT** (taking MQTT as an example). The detailed configuration is as follows:

```
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
        (Ghuiyd9nmGowdZzjPqFtxhm3WUHEbIII) 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) --->
```

- Increase **mbdTLS** frame size

During the Alibaba TLS authentication process, the data packet is large, so the TLS frame size needs to be increased. During OTA, the size needs to be at least **8K**.

Open the ENV tool provided by RT-Thread and use **menuconfig** to configure the **TLS** frame size.

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

– Download the package using the `pkgs --update` command

2.2.2 MQTT Example

This MQTT sample program uses data **Topic** as an example to demonstrate how to use MQTT + TLS/SSL on a device.

The channel establishes two-way communication with the Alibaba Cloud platform.

Sample Files

Sample program path	Verification Platform	illustrate
<code>samples/mqtt/ mqtt-example.c</code>	LinkDevelop, LinkPlatform	Devices and Clouds Based on MQTT Channels Two-way communication routines

Command List

In the example, use the MSH command to start the MQTT example. The command is as follows:

Order	illustrate
<code>ali_mqtt_test start</code>	Start the MQTT example
<code>ali_mqtt_test pub open</code>	Turn on the light and sync the light status to the cloud
<code>ali_mqtt_test pub close</code>	Turn off the lights and synchronize the light-off status to the cloud
<code>ali_mqtt_test stop</code>	Stop MQTT Example

Start MQTT

Use the `ali_mqtt_test start` command to start the MQTT example. After success, the device log shows that the subscription is successful.

The device log is as follows:

```
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 = /
a1P1TITjU9Q/LP-TEST-DEV-1/data!
[dbg] iotx_mc_cycle(1269): SUBACK
event_handle|124 :: subscribe success, packet-id=0 [inf]
iotx_mc_heartbeat_sub(2226): send MQTT ping... [inf] iotx_mc_cycle(1295):
receive ping response!
```

Device release message

Run the `ali_mqtt_test pub open` command to send the message `data` Topic.

The device log is as follows:

```
msh />ali_mqtt_test pub open
ali_mqtt_test_pub|583 ::
publish message:
topic: /a1P1TITjU9Q/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_rcv_PUBLISH(1100): [dbg] Receive Buflen : 1024
iotx_mc_handle_rcv_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: '/'
a1P1TITjU9Q/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 ::----
```

View Cloud Logs

In the operating status of the device details, you can view the message content reported to the cloud by the device.

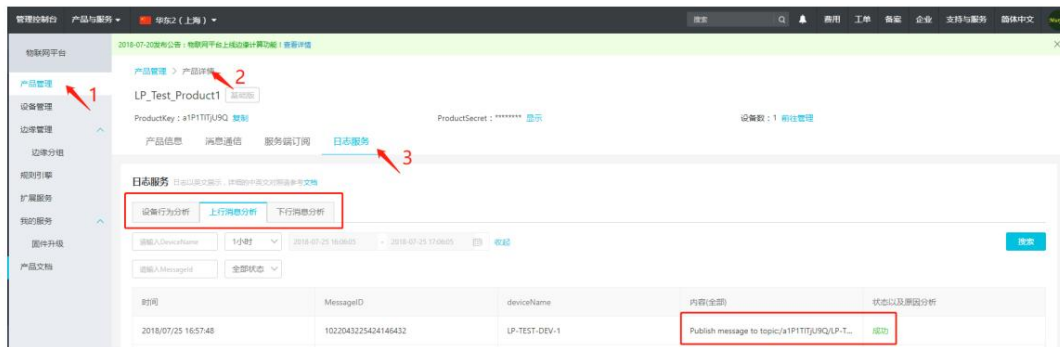


Figure 2.27: View Cloud Logs

Push messages from the cloud to devices



Figure 2.28: Publish messages from the cloud to devices

View device subscription logs

After sending commands using the debug console, the device can receive the commands and the log is as follows:

```
msh />[dbg] iotx_mc_cycle(1277): PUBLISH
[dbg] iotx_mc_handle_rcv_PUBLISH(1091): [dbg]
iotx_mc_handle_rcv_PUBLISH(1092): [dbg]
iotx_mc_handle_rcv_PUBLISH(1096):
    a1P1TITjU9Q/LP-TEST-DEV-1/data
[dbg] iotx_mc_handle_rcv_PUBLISH(1099): [dbg]
iotx_mc_handle_rcv_PUBLISH(1100):
```

Packet Idet : 00000000
Topic Length : 31
Topic Name : /
Payload Len/Room : 33 / 989
Receive Buflen : 1024

```
[dbg] iotx_mc_handle_rcv_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: '/'
a1P1TITjU9Q/LP-TEST-DEV-1/data' (
    Length: 31)
_demo_message_arrive|191 :: Payload: 'This message comes from the cloud'
    (Length: 33)
_demo_message_arrive|192 :: ----
```

Exit MQTT Example

Use the `ali_mqtt_test stop` command to exit the MQTT example. The device log is as follows:

```
msh />ali_mqtt_test stop msh /
>[inf] iotx_mc_unsubscribe(1423): mqtt unsubscribe success,topic = /
    a1P1TITjU9Q/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 Example

Firmware upgrade supports remote over-the-air upgrade of the device's firmware, enabling remote maintenance, function upgrades, problem repairs, and other scenarios. You can specify a new firmware for a product, verify the firmware, start batch upgrades after verification, and view the upgrade results in the firmware details.

Sample Files

Sample program path	Verification Platform	illustrate
samples/ota/ota_mqtt-example.c	LinkDevelop,	Devices based on MQTT channels
	LinkPlatform	OTA routines

Command List

In the example, use the MSH command to start the OTA example. The command is as follows:

Order	illustrate
ali_ota_test start	Launch OTA Example
ali_ota_test stop	Manually exit OTA example

Running the OTA Example

Use the `ali_ota_test start` command to start the OTA routine, and then wait for the cloud to send OTA instructions.

The device log is as follows:

```
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/
a1P1TITjU9Q/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/a1P1TITjU9Q/LP-TEST-DEV-1!
mqtt_client|241 :: wait ota upgrade command.... mqtt_client|241 ::
wait ota upgrade command....
```

New firmware

Here, users are required to upload a bin-type test firmware. Any bin firmware will do. The demo program only performs

The firmware is downloaded and verified, but not written to Flash, so the firmware will not actually be moved and upgraded.



Figure 2.29: LinkPlatform New Platform OTA firmware

Verify the firmware



Figure 2.30: LinkPlatform verify OTA firmware

Device logs

After the push is successful, the device starts downloading the firmware. After the download is complete, the firmware integrity check is automatically performed.

The log is as follows:

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

Cloud upgrade progress display

During the device upgrade process, the cloud will display the progress of the device downloading firmware. When the firmware download is complete and the firmware verification is successful, the device

The SDK reports the new version number to the cloud, and the cloud will show that the upgrade is successful, as shown in the following figure:

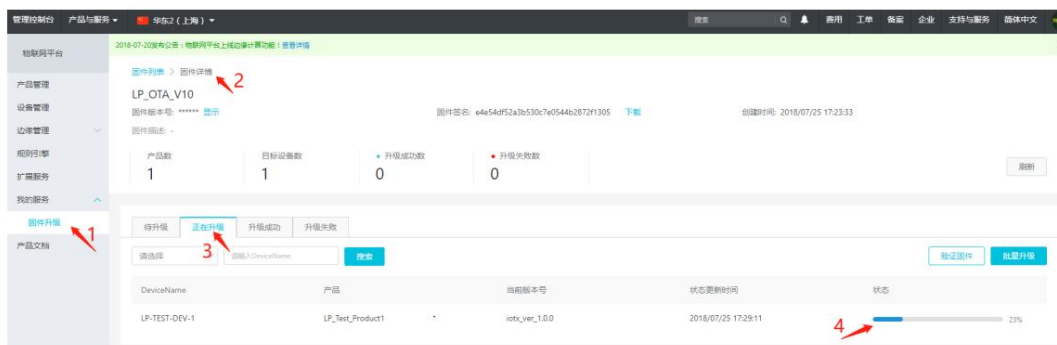


Figure 2.31: Upgrade Progress

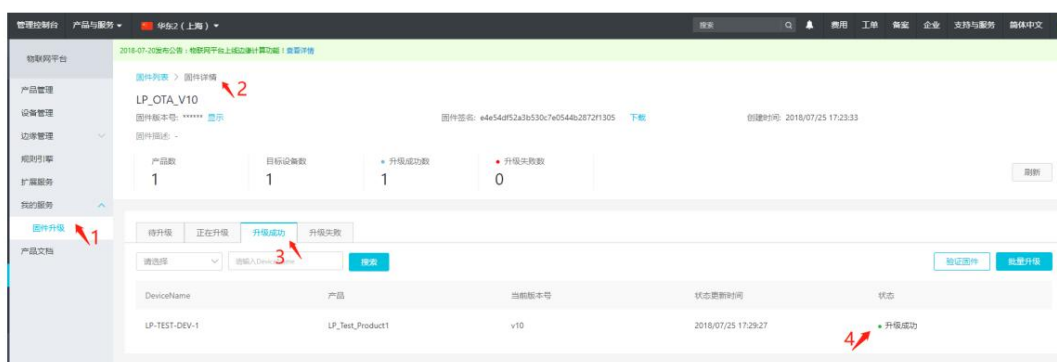


Figure 2.32: update succeeded

Exit OTA routine

The OTA routine will be automatically exited if the upgrade is successful or failed. If you need to exit the OTA routine manually, use

ali_ota_test stop command.

```
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_client340 :: out of sample!
```

2.3 Notes

- Please [configure](#) your device activation credentials (PRODUCT_KEY, DE-VICE_NAME and DEVICE_SECRET)

- Use [menuconfig](#) to select the platform to connect to (LinkDevelop or LinkPlatform) • To enable the OTA function, you must enable encrypted connection, which is selected by default (because OTA upgrades must be downloaded using HTTPS)

[Download Firmware](#))

2.4 Frequently Asked Questions

- MbedTLS returns 0x7200 error

This is usually because the MbedTLS frame length is too small. Please increase the MbedTLS frame length (it needs to be at least **8K** in size).

Chapter 3

working principle

In order to facilitate the connection between devices and the cloud, iotkit SDK encapsulates a variety of connection protocols, such as MQTT, CoAP, HTTP, and TLS. The hardware platform is abstracted to make it more flexible without being restricted by specific hardware platforms.

Usually, users do not need to care about the underlying implementation mechanism of the SDK, but only need to understand how the device interacts with the cloud through the SDK, so that users can understand how to use the application layer API interface to write business logic. Here is an example showing the data interaction process of MQTT and OTA applications.

3.1 MQTT data interaction process

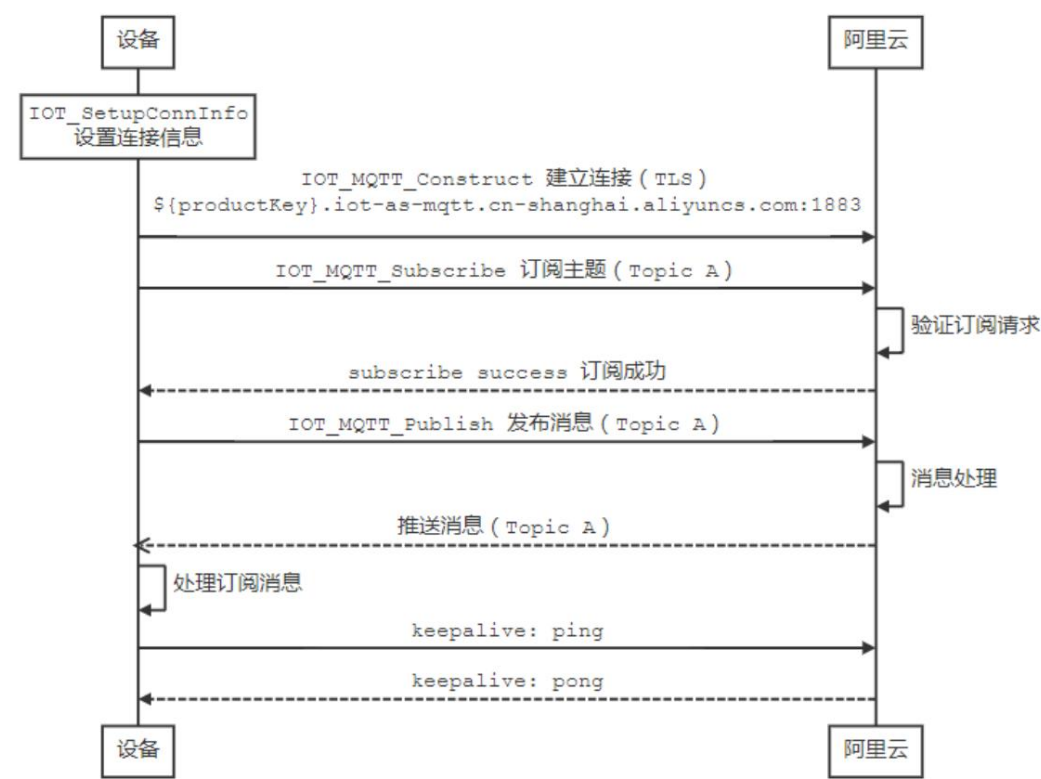


Figure 3.1: MQTT Data interaction process

3.2 OTA data interaction process

Taking the MQTT channel as an example, the firmware upgrade process is shown in the following figure:



Figure 3.2: OTA Firmware Upgrade Process

Chapter 4

user's guidance

The **ali-iotkit** software package encapsulates application layer protocols such as HTTP, MQTT, CoAP and OTA, making it convenient for users

The device is connected to the cloud platform. Here is a brief introduction of some parts.

4.1 MQTT connection

Currently, Alibaba Cloud supports MQTT standard protocol access, which is compatible with 3.1.1 and 3.1 protocol versions. For more information, see

Consider [MQTT 3.1.1](#) and [MQTT 3.1](#) Protocol documentation.

4.1.1 Features

- Support MQTT's PUB, SUB, PING, PONG, CONNECT, DISCONNECT and UNSUB
 - Support
- cleanSession • Do not support will
- and retain msg • Do not support QOS2 •
- Support RRPC
- synchronization mode based on native MQTT topic, the server can synchronously call the device and obtain
- Device receipt results

4.1.2 Security Level

Supports TLSV1, TLSV1.1 and TLSV1.2 versions of the protocol to establish a secure connection.

- TCP channel foundation + chip-level encryption (ID2 hardware integration): high security level • TCP
- channel foundation + symmetric encryption (using device private key for symmetric encryption): medium security level •
- TCP method (data is not encrypted): low security level

4.1.3 Connecting Domain Name

- East China 2 node: productKey.iot-as-mqtt.cn-shanghai.aliyuncs.com:1883 • US West Node: productKey.iot-as-mqtt.us-west-1.aliyuncs.com:1883 • Singapore Node: productKey.iot-as-mqtt.ap-southeast-1.aliyuncs.com:1883

4.1.4 Topic Specification

By default, after a product is created, all devices under the product have the following Topic class permissions:

```

• /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

```

Each Topic rule is called a topic class, and the topic class implements device-dimensional isolation. When each device sends a message, it Replace deviceName with the deviceName of your own device to prevent the topic from being overridden across devices. The topic description is as follows:

- pub: indicates the permission to report data to a topic • sub:

indicates the permission to subscribe to a topic • /

productKey/deviceName/xxx type topic class: can be expanded in the IoT platform console

and Custom

- Topic categories starting with /sys: These are application protocol communication standards agreed upon by the system and are not user-defined.

The specified topic must comply with Alibaba Cloud ALink data standards

- **/sys/productKey/deviceName/thing/xxx** type topic class: used by gateway master and child devices

Topic class, used in gateway scenarios

- Topics starting with /broadcast: broadcast-specific topics • /sys/productKey/

deviceName/rrpc/request/\${messageId}: used for synchronous requests, server

The topic will be dynamically generated for the message ID, and the device can subscribe to the wildcard

- **/sys/productKey/deviceName/rrpc/request/+: After receiving the message, send a** pub message to / sys/productKey/deviceName/rrpc/ response/\${messageId}. The server can receive the result synchronously when sending the request.

4.1.5 Establishing MQTT connection

Use the IOT_MQTT_Construct interface to establish an MQTT connection with the cloud.

If you want to achieve long-term online device, you need to remove IOT_MQTT_Unregister and

In the IOT_MQTT_Destroy part, use while to keep the long connection state.

The sample code is as follows:

```
while(1) {

    IOT_MQTT_Yield(pclient, 200);
    HAL_SleepMs(100);
}
```

4.1.6 Subscribe to Topic

Use the IOT_MQTT_Subscribe interface to subscribe to a Topic.

code show as below:

```
/* 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 Publishing Messages

Use the IOT_MQTT_Publish interface to publish information to the cloud.

code show as below:

```
/* 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 Unsubscribe

Use IOT_MQTT_Unsubscribe interface to unsubscribe from cloud messages

4.1.9 Downlink Data Reception

Use the IOT_MQTT_Yield data receiving function to receive messages from the cloud.

Please call this API wherever you need to receive data. If the system allows, please start a separate thread to execute this interface.

code show as below:

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

4.1.10 Destroy MQTT connection

Use the IOT_MQTT_Destroy interface to destroy the MQTT connection and release the memory.

code show as below:

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

4.1.11 Check the connection status

Use the IOT_MQTT_CheckStateNormal interface to check the current connection status.

This interface is used to query the connection status of MQTT. However, this interface cannot detect the device disconnection immediately, and can only detect disconnection when there is data sending or keepalive.

4.1.12 MQTT Keep Connected

The device needs to send a message at least once within the keepalive_interval_ms time interval, including a ping request.

If the server cannot receive any message within the keepalive_interval_ms time, the IoT platform will disconnect and the device needs to reconnect.

In the IOT_MQTT_Construct function, you can set the keepalive_interval_ms value.

This value is used as the heartbeat interval. The value range of keepalive_interval_ms is 60000~300000.

Sample code:

```

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 Connection

- Support RFC 7252 Constrained Application Protocol. For details, please refer to: RFC [7252](#)
- Use DTLS v1.2 to ensure channel security. For details, please refer to: DTLS [v1.2](#)
- Server address endpoint = productKey.iot-as-coap.cn-shanghai.aliyuncs.com:5684

Please replace productKey with the product key you applied for.

4.2.1 CoAP Conventions

- Not supported? Passing parameters
- in the form of numbers
- Resource discovery is not supported yet
- Only UDP protocol is supported, and currently DTLS must be used
- URI specification, CoAP URI resources are consistent with MQTT TOPIC, refer to [MQTT specification](#)

4.2.2 Application scenarios

The CoAP protocol is suitable for low-power devices with limited resources, especially NB-IoT devices.

The process of connecting NB-IoT devices to the IoT platform is shown in the following figure:

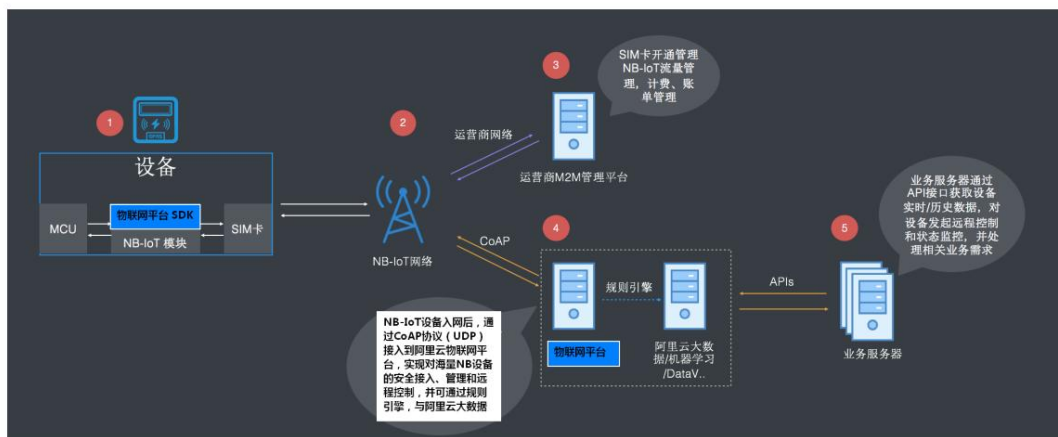


Figure 4.1: CoAP Application Scenario

4.2.3 Establishing a connection

Use the IOT_CoAP_Init and IOT_CoAP_DeviceNameAuth interfaces to establish a CoAP authentication connection with the cloud.

Sample code:

```
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 Sending and Receiving Data

SDK uses the interface IOT_CoAP_SendMessage to send data and IOT_CoAP_GetMessagePayload And IOT_CoAP_GetMessageCode to receive data.

Sample code:

```
/* 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 Downlink Data Reception

Use the IOT_CoAP_Yield interface to receive downlink data from the cloud.

Please call this API wherever you need to receive data. If the system allows, please start a separate thread to execute this interface.

4.2.6 Destroy CoAP connection

Use the IOT_CoAP_Deinit interface to destroy the CoAP connection and release the memory.

4.3 OTA Upgrade

4.3.1 Firmware Upgrade Topic

- The device reports the firmware version to the cloud

/ota/device/inform/productKey/deviceName

- The device subscribes to this topic to receive cloud firmware upgrade notifications

/ota/device/upgrade/productKey/deviceName

- The device reports the firmware upgrade progress

/ota/device/progress/productKey/deviceName

- The device requests whether to upgrade the firmware

/ota/device/request/productKey/deviceName

4.3.2 Firmware Upgrade Instructions

- The device firmware version number only needs to be reported once during system startup, and does not need to be reported periodically.
- Determine whether the device-side OTA upgrade is successful based on the version number
- Initiate a batch upgrade from the OTA server console, and the device upgrade operation record status is pending upgrade

The actual upgrade starts when the OTA system receives the upgrade progress reported by the device, and the device upgrade operation record status is Upgrading.

- When the device is offline, it cannot receive upgrade messages pushed by the server

When the device is online, it actively notifies the server of the online message. The OTA server receives the device online message and verifies whether the device needs to be upgraded. If necessary, it pushes the upgrade message to the device again. Otherwise, it does not push the message.

4.3.3 OTA code description

initialization

The initialization of the OTA module depends on the MQTT connection, that is, the MQTT client handle pclient is obtained first.

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

Report version number

After the OTA module is initialized, the IOT_OTA_ReportVersion interface is called to report the version number of the current firmware. After the upgrade is successful, the new firmware is restarted and the interface is used to report the new firmware version number. After the version number of the cloud and the OTA upgrade task are successfully compared, the OTA upgrade is successful.

The sample code is as follows:

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

Download Firmware

After the MQTT channel obtains the URL for OTA firmware download, it uses HTTPS to download the firmware and stores it in the Flash OTA partition while downloading.

- IOT_OTA_IsFetching() interface: used to determine whether there is firmware to download •
- IOT_OTA_FetchYield() interface: used to download a firmware block • IOT_OTA_IsFetchFinish()
- interface: used to determine whether the download is complete

Sample code:

```
// Determine whether there is firmware available for download
if (IOT_OTA_IsFetching(h_ota)) {
    unsigned char buf_ota[OTA_BUF_LEN]; uint32_t
    len, size_downloaded, size_file; do { // Loop download

        firmware len =
            IOT_OTA_FetchYield(h_ota, buf_ota, OTA_BUF_LEN, 1); if (len > 0) { // Write to
            Flash or other
                storage (download and store at the same time)
            }
        } while (!IOT_OTA_IsFetchFinish(h_ota)); // Check if the firmware is downloaded
    }
}
```

Report download status

Use the IOT_OTA_ReportProgress interface to report the firmware download progress.

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

Determine whether the downloaded firmware is complete

After the firmware is downloaded, use the IOT_OTA_Ioctl interface to verify the integrity of the firmware.

```
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");
}
}
```


Destroy **OTA** connection

Use `IOT_OTA_Deinit` to destroy the OTA connection and release the memory.

4.4 References

- The above content is quoted from the Alibaba Cloud IoT Platform documentation. For more information, please visit the Alibaba Cloud IoT Platform documentation.

Check

- For more API usage instructions, please refer to the API usage documentation. •

For more sample codes, please refer to the sample programs and sample usage instructions.

Chapter 5

API Description

ali-iotkit is a software package ported from RT-Thread for connecting to the Alibaba Cloud IoT platform.
Provided **iotkit-embedded C-SDK**.

Here is the usage guide of Alibaba **iotkit-embedded** API, the content is as follows.

Note: The following API description information comes from Alibaba Cloud. For more details, please refer to [the iotkit-embedded wiki](#).

5.1 Required APIs

Serial number	Function name	illustrate
1	IOT_OpenLog	Start printing log information (log), accept a const char * is the input parameter, indicating the module name
2	IOT_CloseLog	Stop printing log information (log), input parameter is empty
3	IOT_SetLogLevel	Set the log level to print, accept input parameters from 1 to 5, the larger the number, the more detailed the print
4	IOT_DumpMemoryStats	Debug function, print memory usage statistics The input parameter is 1-5. The larger the number, the more printing detailed

5.2 MQTT Function Related API

Serial number	Function name	illustrate
1	IOT_SetupConnInfo	Preparation before MQTT connection, basic In DeviceName + DeviceSecret + ProductKey generates the MQTT connection Username and password, etc.
2	IOT_SetupConnInfoSecure	Preparation before MQTT connection, based on ID2 + DeviceSecret + ProductKey generation Username and password for the MQTT connection etc., ID2 mode enabled
3	IOT_MQTT_CheckStateNormal	After MQTT is connected, call this function to check the long Is the connection normal?
4	IOT_MQTT_Construct	Constructor of MQTT instance, input parameters For the iotx_mqtt_param_t structure, connect MQTT server, and returns the created handle
5	IOT_MQTT_ConstructSecure	MQTT instance constructor, input parameter For the iotx_mqtt_param_t structure, connect MQTT server, and returns the created sentence Handle, ID2 mode enabled
6	IOT_MQTT_Destroy	The destruction function of the MQTT instance, input parameter Statements created for IOT_MQTT_Construct() handle
7	IOT_MQTT_Publish	MQTT session stage, organize a complete MQTT Publish message to the server Send a message to publish
8	IOT_MQTT_Subscribe	MQTT session stage, organize a complete MQTT Subscribe message to the server Send a subscription request
9	IOT_MQTT_Unsubscribe	MQTT session stage, organize a complete MQTT UnSubscribe message to the service The client sends an unsubscribe request
10	IOT_MQTT_Yield	MQTT session phase, MQTT main loop Function, which contains the heartbeat maintenance, server down Receiving of messages, etc.

5.3 CoAP Function Related API

Serial number	Function name	illustrate
1	IOT_CoAP_Init	Constructor of CoAP instance, input parameters For the <code>iotx_coap_config_t</code> structure, return Returns the created CoAP session handle
2	IOT_CoAP_Deinit	The destruction function of the CoAP instance, input parameter Handle created by IOT_CoAP_Init()
3	IOT_CoAP_DeviceNameAuth	Based on the DeviceName requested by the console, DeviceSecret, ProductKey for devices Certification
4	IOT_CoAP_GetMessageCode	CoAP session phase, from the server Obtained from the CoAP Response message Get Respond Code
5	IOT_CoAP_GetMessagePayload	CoAP session phase, from the server Get the message from the CoAP Response message load
6	IOT_CoAP_SendMessage	CoAP session phase, the connection has been successfully established After calling, organize a complete CoAP message Send the document to the server
7	IOT_CoAP_Yield	CoAP session phase, the connection has been successfully established After calling, checking and collecting server Reply message to CoAP Request

5.4 HTTP Function Related API

Serial number	Function name	illustrate
1	IOT_HTTP_Init	Https instance constructor, create a HTTP session handle and return
2	IOT_HTTP_DeInit	Https instance destruction function, destroy all Related data structures
3	IOT_HTTP_DeviceNameAuth	is based on the DeviceName requested by the console. DeviceSecret, ProductKey for devices Certification
4	IOT_HTTP_SendMessage	Https session stage, organize a complete HTTP message is sent to the server and synchronized Get HTTP response message

Serial number	Function name	illustrate
5	IOT_HTTP_Disconnect	Https session phase, close the HTTP layer connection, but still maintain TLS level connect

5.5 OTA function related API

Serial number	Function name	illustrate
1	IOT_OTA_Init	The constructor of the OTA instance creates an The handle of the OTA session is returned
2	IOT_OTA_Deinit	OTA instance destruction function, destroys all related Related data structures
3	IOT_OTA_Iocli	The input and output functions of the OTA instance are based on different The same command word can set the OTA session Properties, or get the status of the OTA session
4	IOT_OTA_GetLastError	During the OTA session, if IOT_OTA_*() function returns an error, calling This interface can be used to obtain the most recent detailed error code
5	IOT_OTA_ReportVersion	During the OTA session, report the current status to the server. Firmware version number
6	IOT_OTA_FetchYield	During the OTA download phase, Download the firmware from the firmware server within the timeout period. Download a firmware content and save it in the input parameter In the buffer
7	IOT_OTA_IsFetchFinish	OTA download stage, determine the iteration adjustment Has IOT_OTA_FetchYield() been used? Download all firmware content
8	IOT_OTA_IsFetching	OTA download phase, determine whether the firmware download is Still in progress, not all firmware has been completed Download
9	IOT_OTA_ReportProgress	Optional API, OTA download stage, call this The function reports to the server that all files have been downloaded. What percentage of firmware content
10	IOT_OTA_RequestImage	Optional API to request firmware download from the server
11	IOT_OTA_GetConfig	Optional API to request remote configuration from the server

5.6 Cloud Connection API

Serial number	Function name	illustrate
1	IOT_Cloud_Connection_Init Constructor of cloud connection instance, input parameter	<p>iotx_cloud_connection_param_pt</p> <p>Structure, returns the created cloud connection session</p> <p>Handle</p>
2	IOT_Cloud_Connection_Deinit Cloud connection instance destruction function, input parameter	<p>for IOT_Cloud_Connection_Init()</p> <p>The sentence created</p> <p>handle</p>
3	IOT_Cloud_Connection_Send_Message sends data to the cloud	
4	IOT_Cloud_Connection_Yield After the cloud connection is successfully established, the server will receive	<p>Messages sent</p>

5.7 CMP Function Related API

Serial number	Function name	illustrate
1	IOT_CMP_Init	<p>CMP instance constructor, input parameter</p> <p>It is the iotx_cmp_init_param_pt structure.</p> <p>There is only one CMP instance</p>
2	IOT_CMP_Register	Subscription Services through CMP
3	IOT_CMP_Unregister	Cancelling a service subscription through CMP
4	IOT_CMP_Send	<p>Send data through CMP to the cloud</p> <p>can also be sent to local devices</p>
5	IOT_CMP_Send_Sync	<p>Send data synchronously via CMP, not supported yet</p> <p>hold</p>
6	IOT_CMP_Yield	<p>Receiving data through CMP, single thread situation</p> <p>Support</p>
7	IOT_CMP_Deinit	Destroy function of the CMP example
8	IOT_CMP_OTA_Start	Initialize the ota function and report the version
9	IOT_CMP_OTA_Set_Callback sets the OTA callback function	
10	IOT_CMP_OTA_Get_Config Get remote configuration	
11	IOT_CMP_OTA_Request_Image Get firmware	

Serial number	Function name	illustrate
12	IOT_CMP_OTA_Yield	Complete OTA function through CMP

5.8 Device Shadow Related (Optional) API

Serial number	Function name	illustrate
1	IOT_Shadow_Construct	Establish an MQTT connection with a device shadow, And returns the created session handle
2	IOT_Shadow_Destroy	Destroys a device shadow's MQTT connection, Destroy all related data structures and release the internal Save, Disconnect
3	IOT_Shadow_Pull	The JSON data cached on the server Pull down to local and update local data attributes
4	IOT_Shadow_Push	Push local data attributes to the server cache Stored JSON data, update the server data <small>According to the attributes</small>
5	IOT_Shadow_Push_Async	Similar to IOT_Shadow_Push() interface, but It is asynchronous, and returns immediately after pushing, without waiting for the server to Server response
6	IOT_Shadow_PushFormat_Add	adds a format to the created data type format. <small>Member attributes</small>
7	IOT_Shadow_PushFormat_Finalize	completes the construction process of a data type format
8	IOT_Shadow_PushFormat_Init	starts the construction process of a data type format
9	IOT_Shadow_RegisterAttribute	creates a data type and registers it to the server. <small>When registering, you need to create the "PushFormat()" interface</small> Created data type format
10	IOT_Shadow_DeleteAttribute	Deletes a successfully registered data attribute
11	IOT_Shadow_Yield	The main loop function of MQTT, which receives The server pushes down messages to update local data <small>According to the attributes</small>

5.9 Master and slave device related (optional function) API

Serial number	Function name	illustrate
1	IOT_Gateway_Construct	Create a master device and establish an MQTT connection Connect and return the created session handle
2	IOT_Gateway_Destroy	Destroy a master's MQTT connection, Destroy all related data structures and release the internal Save, Disconnect
3	IOT_Subdevice_Login	The sub-device is online, notifying the cloud to create the sub-device session
4	IOT_Subdevice_Logout	The sub-device is offline, the cloud is destroyed and the sub-device is established session and all related data structures, Memory
5	IOT_Gateway_Yield	The main loop function of MQTT, which receives Push message from the server
6	IOT_Gateway_Subscribe	Send subscription to the server via MQTT connection Read Request
7	IOT_Gateway_Unsubscribe	Send the MQTT connection to the server. Unsubscribe Request
8	IOT_Gateway_Publish	Send messages via MQTT connection server Publish message
9	IOT_Gateway_RRPC_Register	Register the RRPC callback function of the device, Receive RRPC request initiated by the cloud
10	IOT_Gateway_RRPC_Response	responds to the RRPC request from the cloud
11	IOT_Gateway_Generate_Message_ID	generates message id
12	IOT_Gateway_Get_TOPO	Send a packet to topo/get topic and wait for a response Reply (TOPIC_GET_REPLY reply)
13	IOT_Gateway_Get_Config	Send a packet to the config/get topic and wait Reply (TOPIC_CONFIG_REPLY reply)
14	IOT_Gateway_Publish_Found_List	Report the list of discovered devices

5.10 Linkkit Function Related API

Serial number	Function name	illustrate
1	linkkit_start	Start the linkkit service and establish a connection with the cloud Connect and install callback function

Serial number	Function name	illustrate
2	linkkit_end	Stop the linkkit service and disconnect from the cloud Recycling
3	linkkit_dispatch	Event dispatch function, triggering linkkit_start Installed callbacks
4	linkkit_yield	Linkkit main loop function, including heartbeat Maintenance, receiving of server downlink messages, etc. If multithreading is enabled, do not call this function.
5	linkkit_set_value	Set the TSL of the object according to the identifier Attributes, if the identifier is a struct type, event output type or service Output type, use dot '.' to separate fields; For example, "identifier1.identifier2" refers to Specific items
6	linkkit_get_value	Get the TSL of the object according to the identifier Attributes
7	linkkit_set_tsl	Read TSL files from local, generate and add it to linkkit
8	linkkit_answer_service	Respond to cloud service requests
9	linkkit_invoke_raw_service	Sending raw data to the cloud
10	linkkit_trigger_event	Report device events to the cloud
11	linkkit_fota_init	Initialize OTA-fota service and install it back Call function (need to compile and set macro SERVICE_OTA_ENABLED)
12	linkkit_invoke_fota_service	Execute fota service
13	linkkit_cota_init	Initialize OTA-cota service and install it back Call function (need to compile and set macro SERVICE_OTA_ENABLED SERVICE_COTA_ENABLED)
14	linkkit_invoke_cota_get_config	Device requests remote configuration
15	linkkit_invoke_cota_service	Execute cota service