

配网操作手册

1. 名词解析.....	2
2. 云端控制台使用说明.....	3
2.1 创建项目	3
2.2 创建产品.....	3
2.3 创建设备	5
2.4 获取设备信息	5
2.5 调试设备	6
3. 配网操作说明.....	7
3.1 安装手机 APP	7
3.2 烧写设备端程序.....	8
3.3 一键配网.....	13
3.4 热点配网.....	22
3.5 零配.....	28
4. 常见问题.....	34

1. 名词解析

- **ProductKey**: 由平台颁发的产品唯一标识, 11 位长度的英文数字随机组合
- **ProductSecret**: 由平台颁发的产品加密密钥, 通常与 **ProductKey** 成对出现, 可用于一型一密的认证方案
- **DeviceName**: 用户注册设备时生成的设备唯一编号, 支持系统自动生成, 也可支持用户添加自定义编号, 产品维度内唯一
- **DeviceSecret**: 设备密钥, 与 **DeviceName** 成对出现, 可用于一机一密的认证方案
- **一机一密**: 平台为每台设备颁发一个密钥 (**DeviceSecret**), 设备量产时, 每台设备需要烧录唯一的设备信息 (**ProductKey**, **DeviceName** 和 **DeviceSecret**), 安全性高, 推荐使用
- **一型一密**: 平台为每个产品型号颁发一个密钥 (**ProductSecret**), 设备量产时, 同一个型号的设备, 仅需烧录同样的设备信息 (**ProductKey** 和 **ProductSecret**), 并采用设备自身的唯一标识符 (如 MAC 地址、SN 或 IMEI 号等) 作为 **DeviceName**。为了保障设备不会被黑客攻击和伪造, 平台要求一型一密的设备必须预注册每台设备的 **DeviceName**。当设备首次连云时, 平台会通过该 **DeviceName** 进行身份核对。
- **域名直连**: 直接连接云平台, 平台域名由 **ProductKey** 和站点 **RegionId** 拼接而成, `$(productKey).iot-as-mqtt.$(RegionId):1883`。其中, **productKey** 是平台颁发的产品唯一标识 **ProductKey**; **RegionId** 是设备要接入的站点地址。比如, 华东站点为 `cn-shanghai.aliyuncs.com`
- **鉴权再连接**: 首先使用 HTTPS 到 `iot-auth.$(RegionId):443/auth/devicename` 获取认证后, 再使用 MQTT 连接到 `/public.iot-as-mqtt.$(RegionId):1883`
- **一型一密设备连接**: 首先使用 HTTPS 到 `iot-auth.$(RegionId):443/auth/register/device` 获取设备的 **deviceSecret** 后, 再使用 MQTT 连接到 `$(productKey).iot-as-mqtt.$(RegionId):1883`

2. 云端控制台使用说明

2.1 创建项目

打开浏览器，输入 living.aliyun.com。在页面上输入用户名和密码登录飞燕平台。

1. 控制台左上角选择“中国站”或“国际站”



2. 填写项目名称

“数据节点”为当前中国站/国际站使用到的计算节点。



2.2 创建产品

产品对应真实的联网设备，一个产品对应一个型号。在添加产品的流程中，主要包括 4 个步骤：

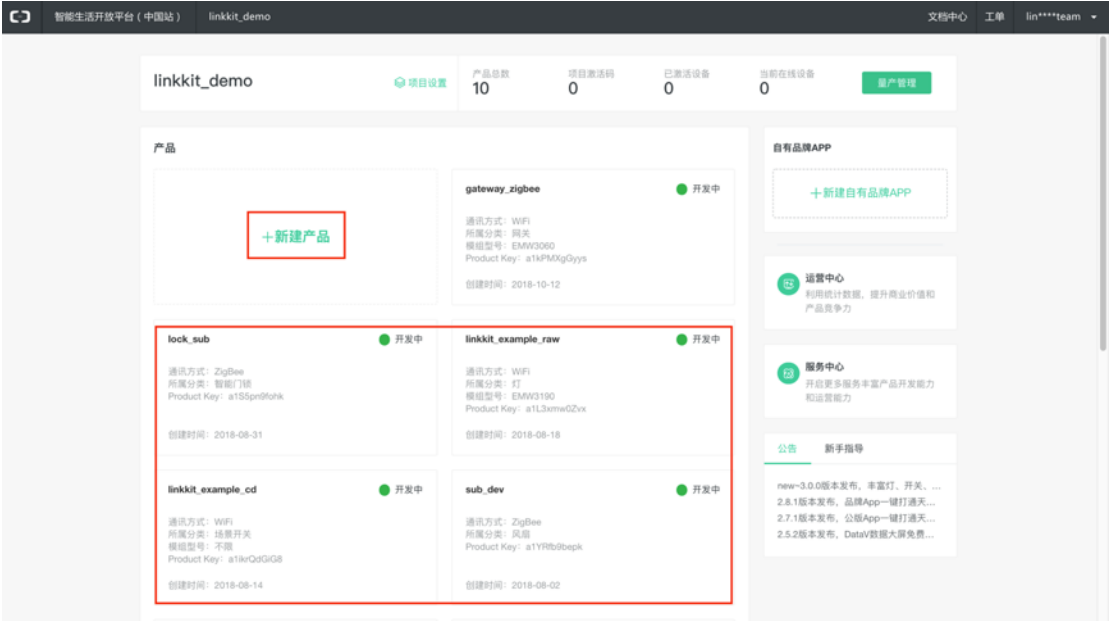
功能定义：定义设备的功能点，即设备需要真实上报的数据。

设备调试：提供了设备端的 IoT SDK 和 MCU SDK，也可以基于 AliOS 认证过的芯片，进行模组的适配和设备的开发；

人机交互：和 APP、语音相关的配置。

批量生产：发布产品，并购买和生成激活码，在量产时对每台设备进行烧录。

每个项目中，可以新建多个产品，一个产品对应一个真实的设备型号。



点击新建产品，先为产品选择一个分类，如电工照明-灯



填写产品的基本信息，如产品名称、节点、通讯方式和数据格式。



数据格式的区别

ICA 标准数据格式：设备需要按照平台标准的协议与云端的进行数据交换，采用 JSON 格式。

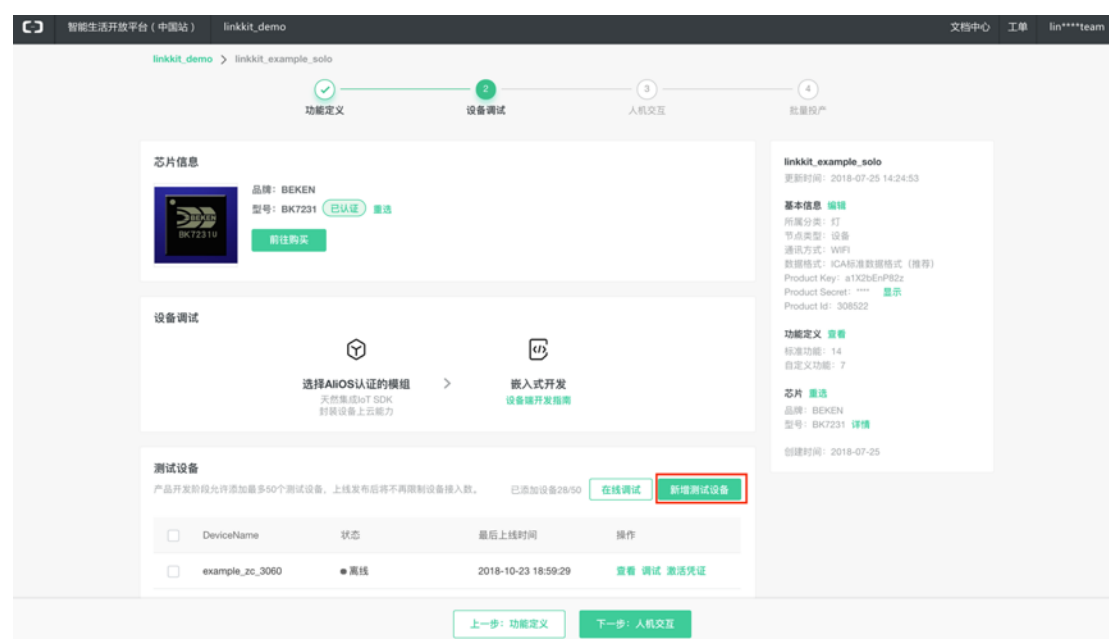
透传自定义格式：设备可以自定义数据格式，并在控制台中进行协议解析，转化为标准 JSON 格式上报云端。

2.3 创建设备

针对每一款产品，平台提供 50 个免费的激活码 (ProductKey, DeviceName 和 DeviceSecret)，用于设备的开发调试。

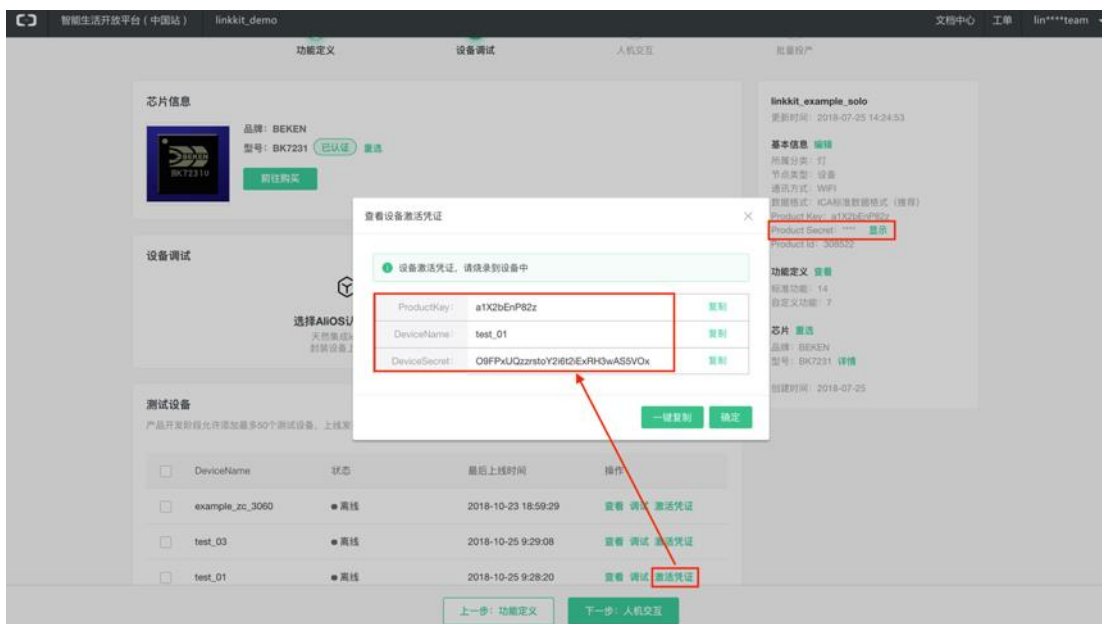
点击“新增测试设备”，填写 DeviceName，可以是自定义字符串、MAC 地址、IMEI 号或自定义 SN 等，须确保产品下唯一。如果不填写 DeviceName，将由系统自动生成。

平台生成设备激活码（三元组）之后，将三元组写入到设备中。当设备激活时，会上报到云端进行鉴权认证。



2.4 获取设备信息

获取设备信息 ProductKey, ProductSecret, DeviceKey 和 DeviceSecret。



2.5 调试设备

设备上线后，可以通过云端控制台查看设备上报的消息，也可以选择一个功能进行在线实时调试，支持分别对设备的属性、服务和事件进行调试，根据功能类型和读写类型，可以选择“设置”或“获取”方法进行调试。

调试属性

选择设备的某个属性，如果该属性支持“读写”或“可读”，可选择方法为“获取”，点击“发送指令”，将触发一个 Get 请求下发到设备端，下方编辑区中将以 JSON 格式显示该属性的最新数据值。



实时日志

设备上线后，页面的“实时日志”将实时刷新设备的上下行数据，包括设备上线、激活、数据上报、云端下发等日志。实时日志将默认自动刷新，您可以在调试过程中随时查看设备的状态，也可以取消勾选“自动刷新”，手动点击“刷新”按钮可以随时获取设备调试日志。点击“清屏”将清除当前屏幕中的日志信息。



3. 配网操作说明

3.1 安装手机 APP

手机 APP 的下载说明: <https://living.aliyun.com/doc#muti-app.html>。

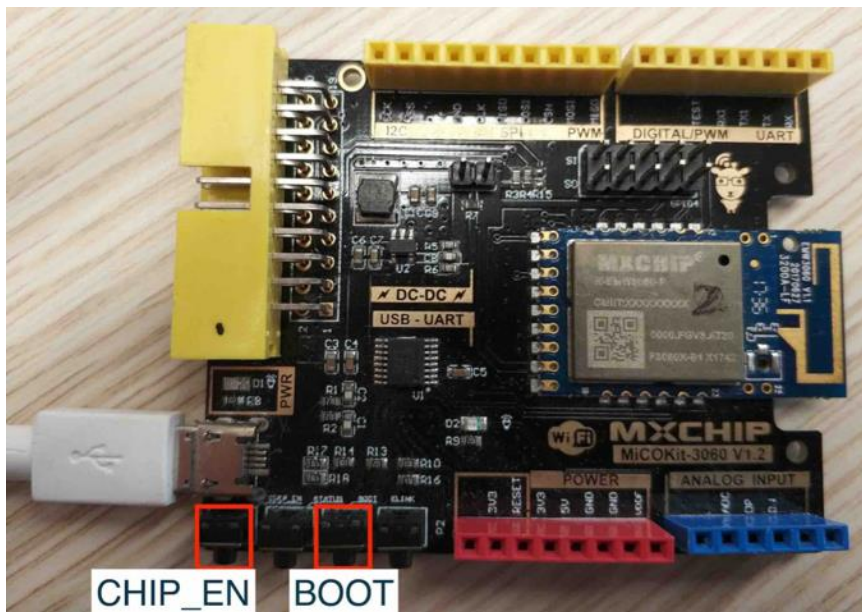
也可以通过扫描下面的二维码下载安装。

如果是已发布的产品，且使用的是设备厂商开发的 APP，那么请向设备厂商索取 APP。



3.2 烧写设备端程序

设备端以硬件 EMW3060 的开发板 MiCOKit-3060 V1.2 和软件 AliOS Things V2.1 中的 linkkitapp 为例。



1. 修改四元组

进入 AliOS Things 的工程目录，打开文件 `app/example/linkkitapp/linkkit_example_solo.c`，用我们自己产品的四元组替换相应的宏，如下所示：

```
#define PRODUCT_KEY        "a1X2bEnP82z"
#define PRODUCT_SECRET     "7jluWm1zql7bt8qK"
#define DEVICE_NAME        "test_01"
#define DEVICE_SECRET      "O9FPxUQzzrstoY2i6t2iExRH3wAS5VOx"
```


如果是正式的产品，四元组应该是存储在 FLASH 中，然后在配网之前，调用相应的 API 配置。

```
int HAL_SetProductKey(_IN_ char *product_key);
int HAL_SetProductSecret(_IN_ char *product_secret);
int HAL_SetDeviceName(_IN_ char *device_name);
int HAL_SetDeviceSecret(_IN_ char *device_secret);
```

2. 编译

在工程目录下面执行: `aos make clean`

然后执行: `aos make linkkitapp@mk3060`

如果没有安装 AliOS Things 的编译环境，请参考说明：
[https://github.com/alibaba/AliOS-Things/wiki/AliOS-Things-Linux-Environm
ent-Setup](https://github.com/alibaba/AliOS-Things/wiki/AliOS-Things-Linux-Environm
ent-Setup)。

3. 烧录

3.1 在 linux 下执行 `sudo minicom -s`，选择串口设置。

```
+-----[configuration]-----+
| Filenames and paths      |
| File transfer protocols  |
+-----+

Serial port setup
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
| Exit from Minicom        |
+-----+
```

3.2 然后选择串口号，以及设置波特率 921600 8N1。然后回车退出。

```
+-----+
| A -   Serial Device       : /dev/ttyUSB0          |
| B - Lockfile Location     : /var/lock              |
| C -   Callin Program      :                       |
| D -   Callout Program     :                       |
| E -   Bps/Par/Bits        : 921600 8N1            |
| F - Hardware Flow Control : No                    |
+-----+
```

```
| G - Software Flow Control : No |
| |
| Change which setting? |
+-----+
```

3.3 选择“Exit”退出。

3.4 同时按下开发板上“CHIP_EN”键和“BOOT”键，然后先松开“CHIP_EN”键，再松开“BOOT”键，进入烧录模式。

3.5 在串口中输入 write 0x13200。

```
=====
|          MOC108 BOOTLOADER MENU          |
|-----|
| Comamnd | Arguments      | Description      |
|-----+-----+-----|
| read    | <address> <size> | read flash      |
|-----+-----+-----|
| write   | <address>        | write flash     |
|-----+-----+-----|
| erase   | <address> <size> | erase flash     |
|-----+-----+-----|
| boot    | <mode>           | boot to APP, ATE or QC |
|-----+-----+-----|
| reboot  |                  | reboot          |
|-----+-----+-----|
=====

@ Author  : Snow Yang
@ Version : 1.1.0
@ Date    : Nov  3 2017 17:31:24
$ write 0x13200

Waiting for the file to be sent ... (press 'a' to abort)
CCCC
```

3.6 按 ctrl+a，松开后再按 z，进入菜单页。

```
+-----+
|          Minicom Command Summary          |
```

```

|
|
|      Commands can be called by CTRL-A <key>
|
|
|      Main Functions      Other Functions
|
|
| Dialing directory..D  run script (Go)....G | Clear Screen.....C |
| Send files.....S  Receive files.....R | cOnfigure Minicom..O |
| comm Parameters....P  Add linefeed.....A | Suspend minicom....J |
| Capture on/off....L  Hangup.....H | eXit and reset....X |
| send break.....F  initialize Modem...M | Quit with no reset.Q |
| Terminal settings..T  run Kermit.....K | Cursor key mode....I |
| lineWrap on/off....W  local Echo on/off..E | Help screen.....Z |
| Paste file.....Y  Timestamp toggle...N | scroll Back.....B |
| Add Carriage Ret...U
|
|
|      Select function or press Enter for none.
|
+-----+

```

3.7 按 S，选择 ymodem。

```

+--[Upload]--+
| zmodem  |
|
| ymodem
|
| xmodem  |
| kermit  |
| ascii   |
+-----+

```

3.8 选择要烧录的文件 linkkitapp@mk3060.bin，按回车开始烧录。

```

+-----[Select one or more files for upload]-----+
| Directory: ...johnny/project/linkkit/AliOS/out/linkkitapp@mk3060/binary |
|
| [..]
|
| [ota]
|
| link.2boot.opts
|
| link.opts
|
| linkkitapp@mk3060.2boot.elf
|

```

```
| linkkitapp@mk3060.2boot.hex |
| linkkitapp@mk3060.2boot.map |
| linkkitapp@mk3060.2boot.out |
| linkkitapp@mk3060.2boot.stripped.elf |
| linkkitapp@mk3060.2boot_map.csv |
linkkitapp@mk3060.bin
| linkkitapp@mk3060.bin.xz |
| linkkitapp@mk3060.elf |
| linkkitapp@mk3060.factory.bin |
| linkkitapp@mk3060.hex |
| linkkitapp@mk3060.map |
| linkkitapp@mk3060.out |
| linkkitapp@mk3060.stripped.elf |
| linkkitapp@mk3060_map.csv |
| mk3060.2boot.bin |
| |
| ( Escape to exit, Space to tag ) |
+-----+

```

3.9 等待烧录完成后，按回车键退出。

```
+-----[ymodem upload - Press CTRL-C to quit]-----+
|Sending: linkkitapp@mk3060.factory.bin |
|Bytes Sent: 646144 BPS:7280 |
|Sending: |
|Ymodem sectors/kbytes sent: 0/ 0k |
|Transfer complete |
| |
| READY: press any key to continue... |
+-----+

```

3.10 按“CHIP_EN”重启开发板。

3.3 一键配网

1. 开发板烧录完成并重启后, 处于 monitor 模式, 会搜索 2.4G 所有信道的 Wi-Fi 路由器。从设备端串口 log 中可以看到有搜到很多 Wi-Fi 的 SSID 以及它们的 MAC 地址。

```
Welcome to AliOS Things

[000008]<V> IOTX_AWSS_START

[crt] __awss_start

[dbg] zconfig_init(281): zconfig_init

Soft_AP_start

[crt] chan 1

[crt] [1] ssid:alibaba-guest, mac:141fba8c8221, chn:1, auth:open, none, none, rssi:-236, adha:0

[crt] [2] ssid:mt7628-2A45, mac:a09dc10d2a44, chn:2, auth:wpa2-psk, aes, tkip, rssi:-236, adha:0

[crt] [3] ssid:antbang_05E9FE, mac:001ca305e9fe, chn:2, auth:wpa2-psk, aes, aes, rssi:-236, adha:0
```

2. 生成配网二维码。利用二维码生成工具, 将文本 http://www.taobao.com?pk=a1X2bEnP82z&dn=test_01 生成二维码。其中“a1X2bEnP82z”和“test_01”分别是测试设备的 ProductKey 和 DeviceName。



3. 手机连上 Wi-Fi 路由器。
4. 打开手机 APP , 以测试版 APP 为例。选择“扫描”图标添加设备。



5. 输入 Wi-Fi 密码，点击“下一步”按钮。注意这里不要更改 Wi-Fi 的 SSID，因为待配网设备需要和手机连在同一个路由器下。



6. 按下开发板上的“BOOT”键,使能设备端配网。从 log 中可以看到 enable awss。

```
# [3279400]<V> awss config press 2
[3279400]<V> do_awss_active 0
[crt] enable awss
[3279400]<V> IOTX_AWSS_ENABLE
[dbg] awss_config_press(189): awss_config_press exit
```

7. 点击“我确认在闪烁”按钮,使能 APP 配网。



8. 设备端收集 APP 发的广播配网包。Log 中的 X+Y 表示该包的序列, X 表示组, Y 表示组内的包序号。例如: 0+4, 表示第 0 组第 4 包数据。

```
[3288780]<V> IOTX_AWSS_LOCK_CHAN
[crt] 0+0 [100] b30 F 4e0
[3288802]<V> IOTX_AWSS_LOCK_CHAN
[crt] 0+1 [100] b32 F 124
[crt] 0+2 [100] b33 F 18f
[crt] 0+3 [92] 258 T 20e
[crt] ssid auto-complete: antbang_05E9FE
[crt] 0+3 [100] b34 F 20e
[crt] ssid auto-complete: antbang_05E9FE
[crt] 0+4 [92] 259 T 290
[crt] 0+4 [100] b35 F 290
[crt] 0+5 [92] 25a T 341
```


可以通过 wireshark 抓取无线空口包，分析配网的过程。配网起始帧包含至少 3 个长度一致的数据包。数据长度在 1300 字节~1400 字节之间。根据加密方式和抓包方式，长度稍有不同。

No.	Time	Source	Destination	Protocol	Length	Info
406	2018-12-10 13:38:57.296610	XiaomiCo_4b:42:50	Broadcast	802.11	1374	QoS Data, SN=1661, FN=0, Flags=p.....TC
407	2018-12-10 13:38:57.296741	XiaomiCo_4b:42:50	Broadcast	802.11	1374	QoS Data, SN=1662, FN=0, Flags=p.....TC
409	2018-12-10 13:38:57.297810	XiaomiCo_4b:42:50	Broadcast	802.11	1374	QoS Data, SN=1664, FN=0, Flags=p.....TC
410	2018-12-10 13:38:57.297121	XiaomiCo_4b:42:50	Broadcast	802.11	1374	QoS Data, SN=1665, FN=0, Flags=p.....TC
411	2018-12-10 13:38:57.297280	XiaomiCo_4b:42:50	Broadcast	802.11	1374	QoS Data, SN=1666, FN=0, Flags=p.....TC
412	2018-12-10 13:38:57.297394	XiaomiCo_4b:42:50	Broadcast	802.11	1374	QoS Data, SN=1667, FN=0, Flags=p.....TC
413	2018-12-10 13:38:57.297581	XiaomiCo_4b:42:50	Broadcast	802.11	422	QoS Data, SN=1668, FN=0, Flags=p.....TC 0+1
416	2018-12-10 13:38:57.298171	XiaomiCo_4b:42:50	Broadcast	802.11	529	QoS Data, SN=1669, FN=0, Flags=p.....TC 0+2
420	2018-12-10 13:38:57.306320	XiaomiCo_4b:42:50	Broadcast	802.11	656	QoS Data, SN=1670, FN=0, Flags=p.....TC 0+3
428	2018-12-10 13:38:57.329654	XiaomiCo_4b:42:50	Broadcast	802.11	786	QoS Data, SN=1671, FN=0, Flags=p.....TC 0+4
432	2018-12-10 13:38:57.349453	XiaomiCo_4b:42:50	Broadcast	802.11	963	QoS Data, SN=1672, FN=0, Flags=p.....TC 0+5
450	2018-12-10 13:38:57.401861	XiaomiCo_4b:42:50	Broadcast	802.11	1353	Data, SN=798, FN=0, Flags=pm...F.C
451	2018-12-10 13:38:57.412989	XiaomiCo_4b:42:50	Broadcast	802.11	1353	Data, SN=799, FN=0, Flags=pm...F.C
452	2018-12-10 13:38:57.423939	XiaomiCo_4b:42:50	Broadcast	802.11	1353	Data, SN=800, FN=0, Flags=pm...F.C
453	2018-12-10 13:38:57.435162	XiaomiCo_4b:42:50	Broadcast	802.11	1353	Data, SN=801, FN=0, Flags=pm...F.C
454	2018-12-10 13:38:57.445994	XiaomiCo_4b:42:50	Broadcast	802.11	1353	Data, SN=802, FN=0, Flags=pm...F.C
455	2018-12-10 13:38:57.460457	XiaomiCo_4b:42:50	Broadcast	802.11	397	Data, SN=804, FN=0, Flags=pm...F.C
456	2018-12-10 13:38:57.464470	XiaomiCo_4b:42:50	Broadcast	802.11	584	Data, SN=805, FN=0, Flags=pm...F.C
457	2018-12-10 13:38:57.469781	XiaomiCo_4b:42:50	Broadcast	802.11	631	Data, SN=806, FN=0, Flags=pm...F.C
458	2018-12-10 13:38:57.477277	XiaomiCo_4b:42:50	Broadcast	802.11	761	Data, SN=807, FN=0, Flags=pm...F.C
477	2018-12-10 13:38:57.516050	XiaomiCo_4b:42:50	Broadcast	802.11	1180	QoS Data, SN=1673, FN=0, Flags=p.....TC 0+6
478	2018-12-10 13:38:57.516167	XiaomiCo_4b:42:50	Broadcast	802.11	1234	QoS Data, SN=1674, FN=0, Flags=p.....TC 0+7
479	2018-12-10 13:38:57.516340	XiaomiCo_4b:42:50	Broadcast	802.11	1344	QoS Data, SN=1675, FN=0, Flags=p.....TC 0+8
480	2018-12-10 13:38:57.516450	XiaomiCo_4b:42:50	Broadcast	802.11	1119	QoS Data, SN=1676, FN=0, Flags=p.....TC 8+0
481	2018-12-10 13:38:57.516630	XiaomiCo_4b:42:50	Broadcast	802.11	1119	QoS Data, SN=1677, FN=0, Flags=p.....TC 8+1
482	2018-12-10 13:38:57.516750	XiaomiCo_4b:42:50	Broadcast	802.11	447	QoS Data, SN=1678, FN=0, Flags=p.....TC 8+1
483	2018-12-10 13:38:57.516833	XiaomiCo_4b:42:50	Broadcast	802.11	592	QoS Data, SN=1679, FN=0, Flags=p.....TC 8+2
487	2018-12-10 13:38:57.525941	XiaomiCo_4b:42:50	Broadcast	802.11	713	QoS Data, SN=1680, FN=0, Flags=p.....TC 8+3
508	2018-12-10 13:38:57.614316	XiaomiCo_4b:42:50	Broadcast	802.11	1213	Data, SN=811, FN=0, Flags=pm...F.C
509	2018-12-10 13:38:57.625802	XiaomiCo_4b:42:50	Broadcast	802.11	1323	Data, SN=812, FN=0, Flags=pm...F.C
510	2018-12-10 13:38:57.634044	XiaomiCo_4b:42:50	Broadcast	802.11	1098	Data, SN=813, FN=0, Flags=pm...F.C
511	2018-12-10 13:38:57.643187	XiaomiCo_4b:42:50	Broadcast	802.11	1098	Data, SN=814, FN=0, Flags=pm...F.C
512	2018-12-10 13:38:57.646594	XiaomiCo_4b:42:50	Broadcast	802.11	426	Data, SN=815, FN=0, Flags=pm...F.C
513	2018-12-10 13:38:57.651364	XiaomiCo_4b:42:50	Broadcast	802.11	567	Data, SN=816, FN=0, Flags=pm...F.C
514	2018-12-10 13:38:57.657134	XiaomiCo_4b:42:50	Broadcast	802.11	688	Data, SN=817, FN=0, Flags=p....F.C
524	2018-12-10 13:38:57.705066	XiaomiCo_4b:42:50	Broadcast	802.11	829	QoS Data, SN=1681, FN=0, Flags=p...R...TC 8+4
525	2018-12-10 13:38:57.705196	XiaomiCo_4b:42:50	Broadcast	802.11	910	QoS Data, SN=1682, FN=0, Flags=p.....TC 8+5
526	2018-12-10 13:38:57.705362	XiaomiCo_4b:42:50	Broadcast	802.11	1043	QoS Data, SN=1683, FN=0, Flags=p.....TC 8+6
527	2018-12-10 13:38:57.705473	XiaomiCo_4b:42:50	Broadcast	802.11	1187	QoS Data, SN=1684, FN=0, Flags=p.....TC 8+7
528	2018-12-10 13:38:57.705621	XiaomiCo_4b:42:50	Broadcast	802.11	1303	QoS Data, SN=1685, FN=0, Flags=p.....TC 8+8
529	2018-12-10 13:38:57.705729	XiaomiCo_4b:42:50	Broadcast	802.11	1120	QoS Data, SN=1686, FN=0, Flags=p.....TC 16+0
530	2018-12-10 13:38:57.705888	XiaomiCo_4b:42:50	Broadcast	802.11	1120	QoS Data, SN=1687, FN=0, Flags=p.....TC 16+1
531	2018-12-10 13:38:57.706003	XiaomiCo_4b:42:50	Broadcast	802.11	424	QoS Data, SN=1688, FN=0, Flags=p.....TC 16+1
538	2018-12-10 13:38:57.717389	XiaomiCo_4b:42:50	Broadcast	802.11	551	QoS Data, SN=1689, FN=0, Flags=p.....TC 16+2

9. 收了所有的配网包后，解析出 SSID 和 PASSWORD。

```

24 0f 0e 10 41 4e 54 42
41 4e 47 3f 10 15 25 19
26 25 2b 18 22 3b 2b 22
3c 00 39 35 26 2f 3f 00
36 0b 16 25
[crt]
[crt] SSID1: [antbang_05E9FE]
[dbg] aes_decrypt_string(74): security level: 3
[crt] zconfig done. ssid:antbang_05E9FE, mac:001ca305e9fe
[dbg] zconfig_got_ssid_passwd_callback(120): ssid:antbang_05E9FE, bssid:001ca305e9fe, wpa2-psk, aes, 2
[3289664]<V> IOTX_AWSS_GOT_SSID_PASSWD

```

如果用的是 Android 版 APP，并且 SSID 和 PASSWORD 长度总和小于 26 字节，则只需通过一个管理帧广播，无需配网包序列。

10. 连接路由器。

获取 SSID 和密码后，就可以连接路由器了，获取 IP 地址。

```

[113320]<V> IOTX_AWSS_CONNECT_ROUTER

```

```

-----suspend station

sta_ip_down

Cancelling scan request

wpa_dInit

wpa_suppllicant_req_scan

Setting scan request: 2.100000 sec

MANUAL_SCAN_REQ

wpa_suppllicant_scan

wpa_drv_scan

wpa_send_scan_req

wpa_driver_scan_cb

wpa_get_scan_rst:1

wpa_suppllicant_connect

Cancelling scan request

wpa_driver_associate

get txpwrtab gain:11

rate:0, pwr_gain:20

add extral movement in test

-----SM_CONNECT_IND_ok

wpa_driver_assoc_cb

Cancelling scan request

hapd_intf_add_key CCMP

add sta_mgmt_get_sta

hapd_intf_add_key CCMP

add is_broadcast_ether_addr

ME_SET_CONTROL_PORT_REQ

sta_ip_start

[120826]<I> Got ip : 192.168.249.51, gw : 192.168.249.1, mask : 255.255.255.0

[120828]<I> Let's post GOT_IP event.

```

11. MQTT 建连

MQTT 与云端建立连接, 连接成功后可以从 log 中看到 mqtt connect success。

```

[inf] dm_client_open(37): CM Fd: 0

[156934]<V> IOTX_CONN_CLOUD

```

```
[inf] iotx_mc_init(2378): MQTT init success!
[156954]<I> Loading the CA root certificate ...
[156958]<I> ok (0 skipped)
[156958]<I> Connecting to /a1X2bEnP82z.iot-as-mqtt.cn-shanghai.aliyuncs.com/1883...
[157044]<I> ok
[157044]<I> . Setting up the SSL/TLS structure...
[157044]<I> ok
[157046]<I> Performing the SSL/TLS handshake...
[157510]<I> ok
[157510]<I> . Verifying peer X.509 certificate..
[157512]<I> certificate verification result: 0x00
[inf] iotx_mc_connect(2787): mqtt connect success!
```

12. 连接路由器广播

在 MQTT 建连的同时，设备也会创建 CoAP 的 Topic 并在本地注册用于设备绑定等。设备端接收到 CoAP 消息后，只处理在本地注册过的消息，否则直接丢弃。

```
[dbg] CoAPResource_register(137): CoAPResource_register, context:0x4214d0, new node
[dbg] CoAPResource_register(140): Register new resource /sys/a1X2bEnP82z/test_01/awss/device/switchap
success, count: 1
[dbg] CoAPResource_register(137): CoAPResource_register, context:0x4214d0, new node
[dbg] CoAPResource_register(140): Register new resource /sys/a1X2bEnP82z/test_01/awss/event/wifilist/get
CoAPResource_register, context:0x4214d0, new node
[dbg] CoAPResource_register(140): Register new resource /sys/awss/device/info/get success, count: 3
[dbg] CoAPResource_register(137): CoAPResource_register, context:0x4214d0, new node
[dbg] CoAPResource_register(140): Register new resource /sys/a1X2bEnP82z/test_01/awss/device/info/get
success, count: 4
[dbg] CoAPResource_register(137): CoAPResource_register, context:0x4214d0, new node
[dbg] CoAPResource_register(140): Register new resource /sys/device/info/notify success, count: 5
[dbg] CoAPResource_register(137): CoAPResource_register, context:0x4214d0, new node
[dbg] CoAPResource_register(140): Register new resource /sys/device/info/get success, count: 6
[dbg] CoAPResource_register(137): CoAPResource_register, context:0x4214d0, new node
[dbg] CoAPResource_register(140): Register new resource /sys/a1X2bEnP82z/test_01/device/info/get success,
count: 7
```

然后通过 CoAP 广播“连接路由器成功”的消息。

```
[dbg] CoAPServerPath_2_option(47): The uri is /sys/awss/event/connectap/notify
```

```
[dbg] awss_notify_dev_info(220): send notify success
```

```
[dbg] CoAPRequestMessage_handle(730): Request path is /sys/awss/event/connectap/notify
```

```
[dbg] awss_notify_dev_info(220): send notify success
```

```
[dbg] __awss_suc_notify(502): resp:0
```

13. 发送 token

MQTT 建连成功后，发送 token 给云端。

```
[dbg] awss_report_token_to_cloud(333): report token:{"id":"0", "version":"1.0",
```

```
"method":"thing.awss.enrollee.match", "params":{
```

```
"token":"BD2C495E355013585DD62D9C92B085E7"}}}
```

```
[dbg] awss_report_token_to_cloud(338): report token res:11
```

接收云端的回复，表示该 token 已被云端接受。

```
[dbg] awss_report_token_reply(97): awss_report_token_reply
```

14. token 广播

设备端收到云端 token 消息的回复后，通过 CoAP 广播包含 token 的“设备信息”。

```
[109266]<V> IOTX_AWSS_BIND_NOTIFY
```

在最新版的设备端 sdk 中，发送“设备信息”的 log 已被屏蔽。可以参考 wireshark 的抓包。

No.	Time	Source	Destination	Protocol	Length	Info
433	2018-12-10 17:27:30.028315	192.168.249.51	255.255.255.255	CoAP	357	CON, MID:9, POST, TKN:0c 03 02 01, /sys/awss/event/connectap/notify (application/json)
438	2018-12-10 17:27:31.492835	192.168.249.51	255.255.255.255	CoAP	482	CON, MID:11, POST, TKN:0e 03 02 01, /sys/device/info/notify (application/json)
441	2018-12-10 17:27:31.900125	192.168.249.130	192.168.249.255	CoAP	136	NON, MID:16, GET, TKN:f4 1a f7 76, /sys/device/info/get
442	2018-12-10 17:27:32.414141	192.168.249.51	255.255.255.255	CoAP	482	CON, MID:11, POST, TKN:0e 03 02 01, /sys/device/info/notify (application/json)
443	2018-12-10 17:27:32.514477	192.168.249.130	192.168.249.255	CoAP	146	NON, MID:116, GET, TKN:f5 1a f7 76, /dev/core/service/dev
450	2018-12-10 17:27:33.648461	192.168.249.51	255.255.255.255	CoAP	482	CON, MID:11, POST, TKN:0e 03 02 01, /sys/device/info/notify (application/json)
459	2018-12-10 17:27:35.693814	192.168.249.51	255.255.255.255	CoAP	482	CON, MID:11, POST, TKN:0e 03 02 01, /sys/device/info/notify (application/json)
818	2018-12-10 17:28:32.622758	192.168.249.130	192.168.249.255	CoAP	146	NON, MID:117, GET, TKN:f6 1a f7 76, /dev/core/service/dev

Offset	Hex	ASCII
0000	ff ff ff ff ff ff ff ff
0010	01 84 00 45 00 00 ff 11	...E...H...3...
0020	ff ff 16 33 16 33 01 70	...3:3:p u0...
0030	82 01 b3 73 79 73 86 64	...sys-d evic- in
0040	66 6f 0e 0e 6f 74 69 66	fo-notify 2{"id
0050	22 3a 22 31 31 22 2c 22	":"11"," version"
0060	3a 22 31 2e 30 22 2c 22	":"1.0"," method":
0070	22 64 65 76 69 63 65 2e	"device. info,not
0080	69 66 79 22 2c 22 70 61	ify","pa-rams":{"
0090	61 77 73 73 56 65 72 22	awssVer":{"smart
00a0	63 6f 6e 66 69 67 22 3a	config:"2.0","z
00b0	63 6f 6e 66 69 67 22 3a	config:"2.0","r
00c0	6f 75 74 65 72 22 3a 22	outer:" 2.0","ap
00d0	22 3a 22 32 2e 30 22 7d	":"2.0"},"produc
00e0	74 4b 65 79 22 3a 22 61	Key":{"a 1x2BnP8
00f0	32 7a 22 2c 22 64 65 76	2z","dev iceName"
0100	3a 22 74 65 73 74 5f 30	":"test_0 1","mac"
0110	3a 22 46 43 3a 45 45 3a	":"FC:EE: E6:05:72
0120	3a 38 38 22 2c 22 60 70	":"89","ip ":"192.1
0130	36 38 2e 32 34 39 2e 35	68,249.5 1","ciph
0140	65 72 54 79 70 65 22 3a	erType": 4,"token
0150	22 3a 22 36 39 44 44 36	":"69006 FB08353
0160	35 34 36 37 30 35 35 37	54678557 8A20B502
0170	43 31 41 22 2c 22 72 65	CIA","re mainTime
0180	22 3a 34 33 37 36 30 2c	":"43760, "type":0
0190	78 7d	}}

APP 通过该消息中的 token，进行设备绑定。但有时候 APP 还会下发“设备信息查询”消息来重新获取 token。

```
[dbg] CoAPRequestMessage_handle(730): Request path is /sys/device/info/get
[dbg] CoAPResourceByPath_get(176): Found the resource: /sys/device/info/get
[dbg] awss_process_get_devinfo(292): sending message to app: {"id":2, "code":200,
"data":{"awssVer":{"smartconfig":"2.0","zconfi
g":"2.0","router":"2.0","ap":"2.0"},"productKey":"a1X2bEnP82z","deviceName":"test_01","mac":"FC:EE:E6:05:72
:89","ip":"192.168.24
9.51","cipherType":4,"token":"50FCAB ...
[dbg] CoAPServerResp_send(268): Send a response message
```

15. 控制设备

设备绑定成功，可以在 APP 上控制设备。如 linkkit_example_solo 这个例子程序，可以打开开关，设置亮度等。





3.4 热点配网

1. 热点配网入口

在一键配网失败后，选择“试试另一种配网方式”，切换至热点配网模式。



请检查以下问题

- 设备与路由器的距离是否过远
- WIFI密码是否正确
- 设备是否处于待配网状态

重试

[试试另一种配网方式](#)

2. 设置热点

按照提示，设置手机热点 SSID：aha，密码：12345678。设置完成，并打开手机热点后，勾选“确认已完成上述指引的操作”，点击“操作完成，进入有下一步”。



3. 连接热点

按下待配网设备的“BOOT”键, 开发板会自动连上 aha 热点。

```
[crt] chan 6
[crt] [0] ssid:alibaba-guest, mac:141fba8c7a71, chn:6, auth:open, none, none, rssi:-236, adha:2
[crt] chan 8
[676498]<V> awss config press 2
[676498]<V> do_awss_active 0
[crt] enable awss
[676498]<V> IOTX_AWSS_ENABLE
[dbg] awss_config_press(189): awss_config_press exit
[crt] chan 10
[crt] [0] ssid:aha, mac:e646dad14250, chn:10, auth:wpa2-psk, aes, aes, rssi:-236, adha:3
[dbg] awss_rcv_callback_aha_ssid(84): found default ssid: aha
[dbg] zconfig_callback_channel_locked(139): channel lock @ 10
```



```
[676588]<V> IOTX_AWSS_LOCK_CHAN

[crt] zconfig done. ssid:aha, mac:e646dad14250

[dbg] zconfig_got_ssid_passwd_callback(120): ssid:aha, bssid:e646dad14250, invalid auth, invalid encry, 0

[676592]<V> IOTX_AWSS_GOT_SSID_PASSWD

[dbg] aws_main_thread_func(368): [channel scanning] 164084 ms

[dbg] aws_main_thread_func(377): final channel 10

[dbg] aws_main_thread_func(396): [channel recving] 164084 ms

[676790]<V> IOTX_AWSS_CONNECT_AHA

-----suspend station

sta_ip_down

Cancelling scan request

wpa_dInit

wpa_supPLICANT_req_scan

Setting scan request: 2.100000 sec

MANUAL_SCAN_REQ

wpa_supPLICANT_scan

wpa_drv_scan

wpa_send_scan_req

wpa_driver_scan_cb

wpa_get_scan_rst:1

wpa_supPLICANT_connect

Cancelling scan request

wpa_driver_associate

get_txpwrtab gain:2

rate:0, pwr_gain:29

add extral movement in test

-----SM_CONNECT_IND_ok

wpa_driver_assoc_cb

Cancelling scan request

hapd_intf_add_key CCMP

add_sta_mgmt_get_sta

hapd_intf_add_key CCMP

add_is_broadcast_ether_addr
```

```
ME_SET_CONTROL_PORT_REQ

sta_ip_start

[686534]<I> Got ip : 192.168.43.40, gw : 192.168.43.1, mask : 255.255.255.0

[686536]<I> In hotspot/router mode, do not post GOT_IP event here.

[686964]<I> AP connected

[dbg] __awss_start(72): awss connect ssid:aha success

[686964]<V> IOTX_AWSS_GOT_IP
```

4. 设备端 Wi-Fi 列表

设备端本地创建 CoAP 的 Topic 并注册。

```
[inf] CoAPResource_register(123): CoAPResource_register:Alread exist

[inf] CoAPResource_register(129): The resource /sys/a1X2bEnP82z/test_01/awss/device/switchap already exist,
re-write it

[inf] CoAPResource_register(123): CoAPResource_register:Alread exist

[inf] CoAPResource_register(129): The resource /sys/a1X2bEnP82z/test_01/awss/event/wifilist/get already exist,
re-write it

[inf] CoAPResource_register(123): CoAPResource_register:Alread exist

[inf] CoAPResource_register(129): The resource /sys/awss/device/info/get already exist, re-write it

[inf] CoAPResource_register(123): CoAPResource_register:Alread exist

[inf] CoAPResource_register(129): The resource /sys/a1X2bEnP82z/test_01/awss/device/info/get already exist,
re-write it

[inf] CoAPResource_register(123): CoAPResource_register:Alread exist

[inf] CoAPResource_register(129): The resource /sys/device/info/notify already exist, re-write it

[inf] CoAPResource_register(123): CoAPResource_register:Alread exist

[inf] CoAPResource_register(129): The resource /sys/device/info/get already exist, re-write it

[inf] CoAPResource_register(123): CoAPResource_register:Alread exist

[inf] CoAPResource_register(129): The resource /sys/a1X2bEnP82z/test_01/device/info/get already exist,
re-write it
```

收到 APP 的“查询 Wi-Fi 列表”消息，设备端回复本地扫描到的所有 Wi-Fi 路由器的 SSID。

```
[dbg] CoAPRequestMessage_handle(730): Request path is /sys/a1X2bEnP82z/test_01/awss/event/wifilist/get

[dbg] CoAPResourceByPath_get(176): Found the resource: /sys/a1X2bEnP82z/test_01/awss/event/wifilist/get

[dbg] wifimgr_process_get_wifilist_request(227): sending message to app: {"id":2, "code":200, "data":"success"}

[dbg] CoAPResourceByPath_get(176): Found the resource: /sys/a1X2bEnP82z/test_01/awss/event/wifilist/get
```

```

[dbg] CoAPObsServer_add(115): Create a observe node, cur have 1 nodes
[dbg] CoAPServerResp_send(268): Send a response message
get txpwrtab gain:17
rate:4, pwr_gain:14
add extral movement in test
[dbg] CoAPAckMessage_handle(567): The CON response message 13 receive ACK, remove it
[dbg] CoAPMessage_retransmit(870): Retansmit the message id 15 len 407
[dbg] CoAPMessage_retransmit(870): Retansmit the message id 15 len 407
[dbg] CoAPRequestMessage_handle(730): Request path is /sys/awss/device/info/notify
[dbg] awss_scan_cb(149): last_ap:0
[dbg] awss_scan_cb(149): last_ap:0
[dbg] awss_scan_cb(149): last_ap:0
[dbg] awss_scan_cb(149): last_ap:0
[dbg] awss_scan_cb(149): last_ap:0
[dbg] awss_scan_cb(149): last_ap:0
[dbg] awss_scan_cb(188): sending message to app: {"id":2, "code":200,
"data":{"awssVer":{"smartconfig":"2.0","zconfig":"2.0","ro
uter":"2.0","ap":"2.0"},
"wifiList":[{"ssid":"antbang_05E9FE","bssid":"00:1C:A3:05:E9:FE","rssi":"-15","auth":"5"},{"ssid":" 免
费安全共享 WiFi","bs ...

```

5. Wi-Fi SSID 和密码

设备端收到“切换路由器”消息，该消息内包含 Wi-Fi 路由器 SSID 和密码。

```

[dbg] CoAPRequestMessage_handle(730): Request path is /sys/a1X2bEnP82z/test_01/awss/device/switchap
[dbg] CoAPResourceByPath_get(176): Found the resource: /sys/a1X2bEnP82z/test_01/awss/device/switchap
[inf] CoAPRequestMessage_ack_send(702): Send Ack Response Message: 14
[dbg] wifimgr_process_switch_ap_request(332): switch ap, len:154,
{"id":3,"method":"awss.device.switchap","params":{"passwd":"
5B30ED0F8A6AAE8C80AD9199FE5A6928","ssid":"antbang_05E9FE","cipherType":"4"},"version":"1.0"}
[dbg] wifimgr_process_switch_ap_request(340): ssid, len:14,
antbang_05E9FE,"cipherType":"4"},"version":"1.0"}
[dbg] wifimgr_process_switch_ap_request(365): enr
[dbg] aes_decrypt_string(74): security level: 4
[dbg] wifimgr_process_switch_ap_request(423): Sending message to app: {"id":3, "code":200, "data":"success"}

```

6. 连接路由器及绑定

断开与当前热点的连接，连接路由器。后续 MQTT 建连及绑定流程与一键配网相同。

```
[dbg] wifmgr_process_switch_ap_request(424): switch to ap: 'antbang_05E9FE'  
[dbg] CoAPServerResp_send(268): Send a response message  
[dbg] wifmgr_process_switch_ap_request(431): sending succeeded.  
[dbg] wifmgr_process_switch_ap_request(444): connect 'antbang_05E9FE'  
[dbg] wifmgr_process_switch_ap_request(448): bssid: 00:1c:a3:05:e9:fe
```

3.5 零配

1. 准备新开发板

按照上文中的步骤，DeviceName 为 test_01 的开发板已配网并绑定成功。在这个测试环境的基础上，准备一块新的开发板 test_02，烧录程序。

例如，新板子的四元组如下：

```
#define PRODUCT_KEY      "a1X2bEnP82z"  
#define PRODUCT_SECRET   "7jluWm1zql7bt8qK"  
#define DEVICE_NAME      "test_02"  
#define DEVICE_SECRET    "QBIQNahUQIDbJdp3UCBzipGJ4RfQjwqtq"
```

2. 待配网设备信息通告

待配网的开发板 test_02 上电后，会进入 monitor 模式。同时它也会发送 probe request，里面携带了自己的 ProductKey 和 DeviceName。

No.	Time	Source	Destination	Protocol	Length	Info
1032	2018-12-11 13:41:21.302854	Shenzhen_0c:82:50	Formike_05:72:69	802.11	226	Probe Response, SN=3242, FN=0, Flags=...R...C, BI=200, SSID=alibab...
1033	2018-12-11 13:41:21.306576	Netgear_e9:9b:3e	Formike_05:72:69	802.11	413	Probe Response, SN=3727, FN=0, Flags=...R...C, BI=100, SSID=NETGEA...
1037	2018-12-11 13:41:21.316687	Netgear_e9:9b:3e	Formike_05:72:69	802.11	413	Probe Response, SN=3727, FN=0, Flags=...R...C, BI=100, SSID=NETGEA...
1038	2018-12-11 13:41:21.317288	Formike_05:72:69	Broadcast	802.11	75	Probe Request, SN=0, FN=0, Flags=.....C, SSID=adha
1040	2018-12-11 13:41:21.332360	Netgear_e9:9b:3e	Formike_05:72:69	802.11	413	Probe Response, SN=3727, FN=0, Flags=...R...C, BI=100, SSID=NETGEA...
1041	2018-12-11 13:41:21.334498	Shenzhen_0c:91:00	Formike_05:72:69	802.11	226	Probe Response, SN=4023, FN=0, Flags=...R...C, BI=200, SSID=alibab...
1042	2018-12-11 13:41:21.336489	Shenzhen_0c:91:00	Formike_05:72:69	802.11	226	Probe Response, SN=4023, FN=0, Flags=...R...C, BI=200, SSID=alibab...
1044	2018-12-11 13:41:21.347732	Shenzhen_0c:91:01	Formike_05:72:69	802.11	206	Probe Response, SN=2663, FN=0, Flags=...R...C, BI=200, SSID=alibab...
1046	2018-12-11 13:41:21.352041	Shenzhen_0c:91:01	Formike_05:72:69	802.11	206	Probe Response, SN=2663, FN=0, Flags=...R...C, BI=200, SSID=alibab...
1134	2018-12-11 13:41:21.737559	Formike_05:72:69	Broadcast	802.11	139	Probe Request, SN=1573, FN=0, Flags=...R...C, SSID=Wildcard (Broadca...
1162	2018-12-11 13:41:21.816527	Formike_05:72:69	Broadcast	802.11	75	Probe Request, SN=14, FN=0, Flags=...R...C, SSID=adha
1254	2018-12-11 13:41:22.323591	Formike_05:72:69	Broadcast	802.11	75	Probe Request, SN=20, FN=0, Flags=...R...C, SSID=adha
1347	2018-12-11 13:41:22.748208	Formike_05:72:69	Broadcast	802.11	139	Probe Request, SN=25, FN=0, Flags=...R...C, SSID=Wildcard (Broadca...
1373	2018-12-11 13:41:22.838865	Formike_05:72:69	Broadcast	802.11	75	Probe Request, SN=26, FN=0, Flags=...R...C, SSID=adha
1374	2018-12-11 13:41:22.841772	96:28:2e:ab:a0:e6	Formike_05:72:69	802.11	303	Probe Response, SN=1573, FN=0, Flags=...R...C, BI=100, SSID=adha
1399	2018-12-11 13:41:23.008403	Formike_05:72:69	Broadcast	802.11	139	Probe Request, SN=28, FN=0, Flags=...R...C, SSID=Wildcard (Broadca...
1408	2018-12-11 13:41:23.014188	Shenzhen_0c:9b:21	Formike_05:72:69	802.11	206	Probe Response, SN=1989, FN=0, Flags=...R...C, BI=200, SSID=alibab...

Supported Rates: 1(B) (0x82)
Supported Rates: 2(B) (0x84)
Supported Rates: 5.5(B) (0x8b)
Supported Rates: 11(B) (0x96)
Supported Rates: 6(B) (0x8c)
Supported Rates: 9(B) (0x92)
Supported Rates: 12(B) (0x98)
Supported Rates: 18(B) (0xa4)
Tag: Extended Supported Rates 24(B), 36, 48, 54, [Mbit/sec]
Tag Number: Extended Supported Rates (50)
Tag length: 4
Extended Supported Rates: 24(B) (0xb0)
Extended Supported Rates: 36 (0x48)
Extended Supported Rates: 48 (0x60)
Extended Supported Rates: 54 (0x6c)
Tag: Vendor Specific: Alibaba Cloud Computing Ltd.
Tag Number: Vendor Specific (221)
Tag length: 66
OUI: d8:9e:e0 (Alibaba Cloud Computi
Vendor Specific OUI Type: 170
Vendor Specific Data: aa0107746573745f3032000b6131583262456e5038327a10...

0030 00 00 00 01 08 02 84 b0 96 0c 92 98 a4 32 04 b0
0040 48 60 6c dd 42 d8 96 e0 a0 01 07 74 65 73 74 5f
0050 30 32 00 00 61 11 58 32 67 45 6e 50 38 32 7a 10
0060 6d 60 4c 75 fa 02 21 86 7f a0 71 1f e0 89 69 6f
0070 04 00 14 54 6d 1f 47 ba 87 ad 35 ed 64 14 12 4a
0080 ed ac 16 38 c4 c4 ce eb 11 c7 e1

包含待配网设备的
productkey:a1X2bEnP82z
devicename:test_02

3. 上报设备信息

已配网设备收到 probe request 之后, 会将待配网设备信息保存到本地的列表。

```
[dbg] enrollee_put(861): new enrollee[1] dev_name:test_02 time:78514
```

然后已配网设备会将本地发现的待配网设备, 通过 MQTT 上报给云端。

```
[dbg] awss_report_enrollee(636): topic:/sys/a1X2bEnP82z/test_01/thing/awss/enrollee/found
g/awss/enrollee/found, packet:{"id":"4", "version":"1.0", "method":"thing.awss.enrollee.found",
"params":{"awssVer":{"smartconfig":"2.0", "zconfig":"2.0", "router":"2.0", "ap":"2.0"}, "type":0, "ssid":"antbang_05
E9FE", "bssid" ...
```

```
[crt] enrollee report result:success, period:60000ms
```

```
[dbg] iotx_mc_cycle(1848): PUBACK
```

```
[dbg] iotx_mc_cycle(1866): PUBLISH
```

```
[dbg] iotx_mc_handle_rcv_PUBLISH(1621): Packet Ident : 00000000
```

```
[dbg] iotx_mc_handle_rcv_PUBLISH(1622): Topic Length : 56
```

```
[dbg] iotx_mc_handle_rcv_PUBLISH(1626): Topic Name :
```

```
/sys/a1X2bEnP82z/test_01/thing/awss/enrollee/found_reply
```

```
[dbg] iotx_mc_handle_rcv_PUBLISH(1629): Payload Len/Room : 169 / 1139
```

```
[dbg] iotx_mc_handle_rcv_PUBLISH(1630): Receive BufLen : 1200
```

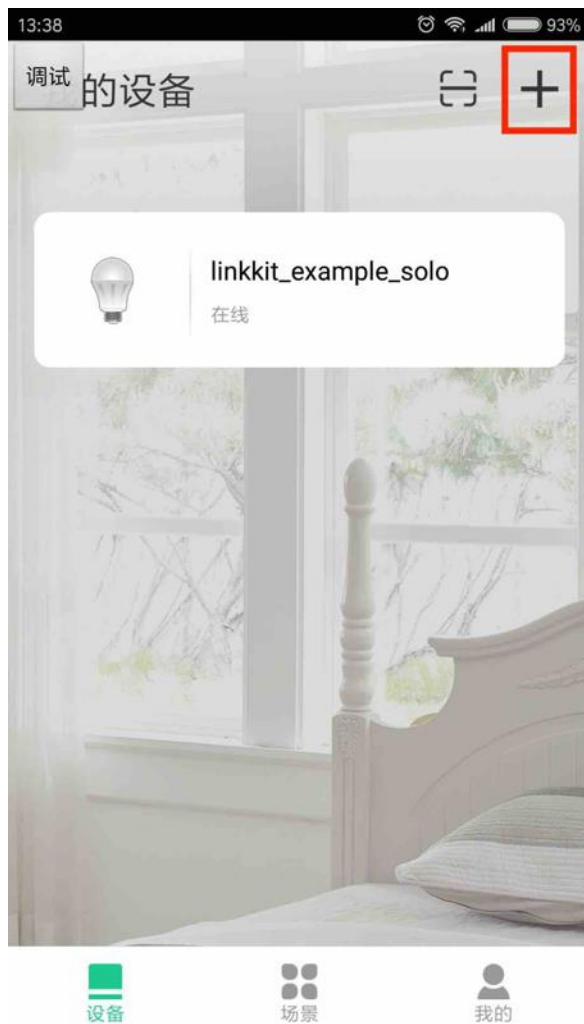
```
[dbg] iotx_mc_handle_rcv_PUBLISH(1641): delivering msg ...
```

```
[dbg] iotx_mc_deliver_message(1342): topic be matched
```

```
[dbg] awss_report_enrollee_reply(555): found
reply:{"code":200,"data":[{"deviceName":"test_02","productKey":"a1X2bEnP82z","timeout":300}],"id":"4","message":"success","method":"thing.awss.enrollee.found","version":"1.0"}
[dbg] awss_report_set_interval(464): key:a1X2bEnP82z, dev_name:test_02, interval:300
```

4. 添加设备

手机 APP 上点击“+”，APP 会从云端拉取设备列表。



选择“添加”，在下个页面点击“我确认在闪烁”，同时按下待配网的开发板上的“BOOT”键。



5. 设备信息同步

APP 将所选设备的信息同步给云端，云端会将该待配网设备的信息下发给已配网设备。从已配网设备的 log 里面可以看到“给待配网设备配网”指令。

```
[dbg] iotx_mc_cycle(1866): PUBLISH
```

```
[dbg] iotx_mc_handle_rcv_PUBLISH(1621):      Packet Ident : 00009746
[dbg] iotx_mc_handle_rcv_PUBLISH(1622):      Topic Length : 52
[dbg] iotx_mc_handle_rcv_PUBLISH(1626):      Topic Name :
/sys/a1X2bEnP82z/test_01/thing/awss/enrollee/checkin
[dbg] iotx_mc_handle_rcv_PUBLISH(1629):      Payload Len/Room : 150 / 1141
[dbg] iotx_mc_handle_rcv_PUBLISH(1630):      Receive Buflen : 1200
[dbg] iotx_mc_handle_rcv_PUBLISH(1641): delivering msg ...
[dbg] iotx_mc_deliver_message(1342): topic be matched
[dbg] awss_enrollee_checkin(160): checkin len:150,
payload:{"method":"thing.awss.enrollee.checkin","id":"203388548","params":{"productKey":"a1X2bEnP82z","deviceName":"test_02","timeout":120},"version":"1.0.0"}
```

已配网设备向云端查询待配网设备的 secret。

```
[dbg] iotx_mc_cycle(1866): PUBLISH
[dbg] iotx_mc_handle_rcv_PUBLISH(1621):      Packet Ident : 00000000
[dbg] iotx_mc_handle_rcv_PUBLISH(1622):      Topic Length : 47
[dbg] iotx_mc_handle_rcv_PUBLISH(1626):      Topic Name :
/sys/a1X2bEnP82z/test_01/thing/cipher/get_reply
[dbg] iotx_mc_handle_rcv_PUBLISH(1629):      Payload Len/Room : 188 / 1148
[dbg] iotx_mc_handle_rcv_PUBLISH(1630):      Receive Buflen : 1200
[dbg] iotx_mc_handle_rcv_PUBLISH(1641): delivering msg ...
[dbg] iotx_mc_deliver_message(1342): topic be matched
[dbg] awss_get_cipher_reply(358): cipher len:188,
payload:{"code":200,"data":{"deviceName":"test_02","productKey":"a1X2bEnP82z","secret":"bcb449cde069dd1720263009eb02ad4e"},"id":"5","message":"success","method":"thing.cipher.get","version":"1.0"}
[dbg] enrollee_enable_somebody_cipher(209): key:a1X2bEnP82z, dev_name:test_02,
cipher:bcb449cde069dd1720263009eb02ad4e
```

6. Wi-Fi SSID 和密码

已配网设备发送 probe request, 里面包含 Wi-Fi 路由器的 SSID 和密码 (密码经过特殊加密处理)。

```
[dbg] registrar_raw_frame_init(1020): dump registrar info:
40 00 00 00 ff ff ff ff ff fc ee e6 05 72 89
ff ff ff ff ff c0 79 00 00 01 08 82 84 8b 96
8c 92 98 a4 32 04 b0 48 60 6c dd 3d d8 96 e0 ab
```


01 14 54 6d 1f 47 ba 87 ad 35 ed 64 14 12 4a ed

ac 16 38 c4 c4 ce 01 0e 61 6e 74 62 61 6e 67 5f

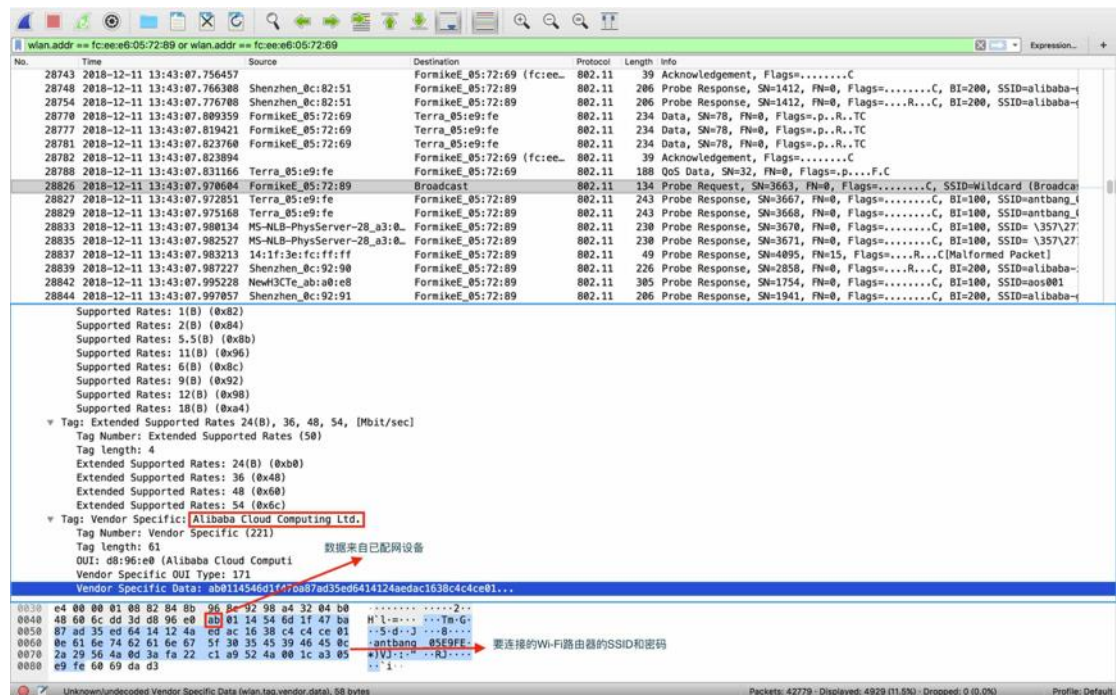
30 35 45 39 46 45 0c 2a 29 56 4a 0d 3a fa 22 c1

a9 52 4a 00 1c a3 05 e9 fe 3f 84 10 9e

[dbg] enrollee_checkin(442): registrar_raw_frame_send

[dbg] enrollee_checkin(442): registrar_raw_frame_send

从抓包文件中，也可以看到该数据包。



待配网设备接收 Wi-Fi 路由器 SSID 和密码。

[dbg] decrypt_ssid_passwd(197): Registrar ssid:antbang_05E9FE

[dbg] aes_decrypt_string(74): security level: 4

[dbg] decrypt_ssid_passwd(218): ssid:antbang_05E9FE

[dbg] zconfig_callback_channel_locked(139): channel lock @ 8

[089984]<V> IOTX_AWSS_LOCK_CHAN

[crt] zconfig done. ssid:antbang_05E9FE, mac:001ca305e9fe

[dbg] zconfig_got_ssid_passwd_callback(120): ssid:antbang_05E9FE, bssid:001ca305e9fe, wpa2-psk, aes, 10

[089984]<V> IOTX_AWSS_GOT_SSID_PASSWD

7. 连接路由器及绑定

连接路由器以及后续 MQTT 建连和绑定流程与一键配网相同。

4. 常见问题

1. 未配网的 Wi-Fi 设备上电后，没有扫描到附近 Wi-Fi 的 SSID。

原因：HAL 层函数适配不当。调用 HAL_Awss_Open_Monitor()函数会指定一个 callback。在 Wi-Fi 驱动程序的接收函数里面需要调用该 callback。示例伪代码如下：

```
awss_rcv_80211_frame_cb_t g_ieee80211_handler;

void monitor_data_handler(uint8_t *buf, int len, hal_wifi_link_info_t *info)
{
    /* 执行回调函数 */
    (*g_ieee80211_handler)((char *)buf, len, AWSS_LINK_TYPE_NONE, 0, info->rssi);
}

void HAL_Awss_Open_Monitor(_IN_ awss_rcv_80211_frame_cb_t cb)
{
    /* 获取指定网卡 */
    wifi_module_t *module = wifi_driver_get_module(if_name);

    /* 保存回调函数指针 */
    g_ieee80211_handler = cb;

    /* 将 monitor_data_handler 注册为 Wi-Fi 的接收处理函数 */
    wifi_driver_register_monitor_cb(module, monitor_data_handler);

    wifi_driver_start_wifi_monitor(module);

    HAL_Awss_Switch_Channel(6, 0, NULL);
}
```

其中回调函数的参数说明如下：

int (*cb)(char *buf, int length, enum AWSS_LINK_TYPE link_type, int with_fcs, signed char rssi)

- buf：包含 IEEE802.11 的 header 和 payload。
- length：IEEE802.11 的 header 长度加上 payload 长度，不包含最后 4 字节的 FCS。
- link_type：设为 AWSS_LINK_TYPE_NONE。
- with_fcs：设为 0。

- rssi: 当前数据包的信号强度。

2. 无法收全配网包

无法收全配网包，超时后重新开始扫描各个信道。

```
[crt] 8+6 [99] 33d T 395
[crt] 8+7 [100] 33e T 40d
[dbg] awss_rcv_callback_smartconfig(999): drop: 33e == 33e
[crt] 8+8 [100] 33f T 4b4
[dbg] aws_main_thread_func(374): channel rescanning...
[dbg] awss_rcv_callback_smartconfig(999): drop: 33f == 33f
[crt] chan 11
[crt] [20] ssid:NETGEAR09, mac:a040a062dbe7, chn:13, auth:wpa2-psk, aes, aes, rssi:-67, adha:0
```

原因 1：待配网设备、路由器、手机距离太远，造成 Wi-Fi 丢包严重。建议缩短三者的距离再测试。

原因 2：有些路由器会做优化：路由器上没有其他的设备，不转发广播包。再增加一个额外的设备或手机连接到该路由器下。

原因 3：检查待配网设备的 Wi-Fi 驱动，确保其能接收 from DS 的包，也能接收 to DS 的包。

3. Wi-Fi 密码解密出错

接收配网包完成，但是 Wi-Fi 密码解密出错。

```
[15:00:27:581]SSID1: [MW305R]
[15:00:27:582]security level: 3descrypted '123456-x'
[15:00:27:583]passwd err
[15:00:27:587][063325]<V> IOTX_AWSS_PASSWD_ERR
```

原因 1：Wi-Fi 密码是经过加密的，密钥是根据待配网设备的信息生成的。所以有可能 APP 添加设备的时候，选错了产品，ProductKey 不匹配，导致解密出错。

原因 2：HAL 层解密函数对接的有问题。可用以下代码验证：

```
/* 验证函数 */
unsigned char iv[16] = {0};
unsigned char key[16] = {207,12,174,46,47,250,168,12,131,204,2,12,221,180,137,174};
unsigned char src[20] =
{0xd7,0xf6,0x54,0xec,0xd6,0x19,0x50,0x75,0xff,0x4e,0x93,0xb4,0xab,0x36,0x9a,0x3,0x72,0xe7,0x59,0
x9d};
```

```

unsigned char dst[20] = {0};

p_HAL_Aes128_t aes_d_h;

aes_d_h = HAL_Aes128_Init(key, iv, HAL_AES_DECRYPTION);
HAL_Aes128_Cfb_Decrypt(aes_d_h, src, sizeof(src), dst);

/* 解密出来的数据 */
0x63 0x61 0x6F 0x68 0x61 0x69 0x62 0x6F
0x31 0x39 0x38 0x33 0x63 0x61 0x6F 0x68
0x61 0x69 0x62 0x6F

```

4. 连接路由器失败

原因 1：用户在 APP 上设置了错误的 Wi-Fi 密码，导致 APP 发给待配网设备的密码是错误的。

原因 2：HAL 层函数 HAL_Awss_Connect_Ap()实现时，使用 auth 和 encry 参数，而配网库调用该函数时，这两个参数未必是正确的。所以请使用 Wi-Fi 路由器的加密方式去连接路由器，忽略 auth 和 encry 参数。

```

int HAL_Awss_Connect_Ap(_IN_ uint32_t connection_timeout_ms,

    _IN_ char    ssid[HAL_MAX_SSID_LEN],

    _IN_ char    passwd[HAL_MAX_PASSWD_LEN],

    _IN_OPT_enum AWSS_AUTH_TYPE auth,

    _IN_OPT_enum AWSS_ENC_TYPE encry,

    _IN_OPT_uint8_t bssid[ETH_ALEN],

    _IN_OPT_uint8_t channel)

```

5. APP 端显示“需要授权绑定”

该设备已被其他账号绑定，需要先与其他账号解绑，才能重新绑定。

6. APP 端显示“Token not found”

原因 1：由于网络，账号等原因，MQTT 连接云端失败。

原因 2：设备发送了 token 给云端，没有收到云端的回复。

原因 3：设备在局域网广播了 token，手机 APP 没有收到。

原因 4：手机 APP 发送“查询设备信息”指令，设备没有收到；也有可能设备回复了设备信息，手机 APP 没有收到。

