
BEKEN-Mesh工程手册

BK3633 Mesh Provisioner User Guide V1.0.1

Beken Corporation
Building 41, Capital of Tech Leaders, 1387 Zhangdong Road,
Zhangjiang High-Tech Park, Pudong New District, Shanghai, China
Tel: (86)21 51086811
Fax: (86)21 60871089

This document contains information that may be proprietary to, and/or secrets of, Beken Corporation. The contents of this document should not be disclosed outside the companies without specific written permission.

Disclaimer: Descriptions of specific implementations are for illustrative purpose only, actual hardware implementation may differ.

目录

1. 目的	4
1.1. 支持 Mesh 特性	4
1.2. BK3633 的软件架构	4
1.3. Provisioner 协议说明	4
2. 开发者指南	6
2.1. 初始化	6
2.2. 元素注册与模型添加	7
2.3. Provisioner 配置	8
2.4. Provisioning	9
2.5. Provisioned device 配置	9
2.6. 日志打印	10
3. 工程演示	12
3.1. 应用准备	12
3.2. Mesh Light	12
3.3. Mesh Provisioner	13
3.4. Provisioning	14

版本	发布/更新日期	更新人员	重要变更内容
V1.0.0	2021/03/15	韩宇航	初始版本
V1.0.1	2021/03/24	韩宇航	更新了工程演示等章节

1. 目的

本文档的目的是简要阐述基于博通BK系列芯片软件开发套件 (BK3633 Alios SDK) 的Mesh Provisioner特性开发，便于初开发人员加快对工程的理解以及方便添加自定义的Mesh应用功能。

1.1. 支持 Mesh 特性

BK3633 Alios SDK允许开发者添加provisioner角色的应用代码，及符合《SIG Mesh Model specification》定义的通用模型如：Light client models等。此外，SDK提供相应的APIs到开发者以创建基于特定设备特性的自定义模型。

1.2. BK3633 的软件架构

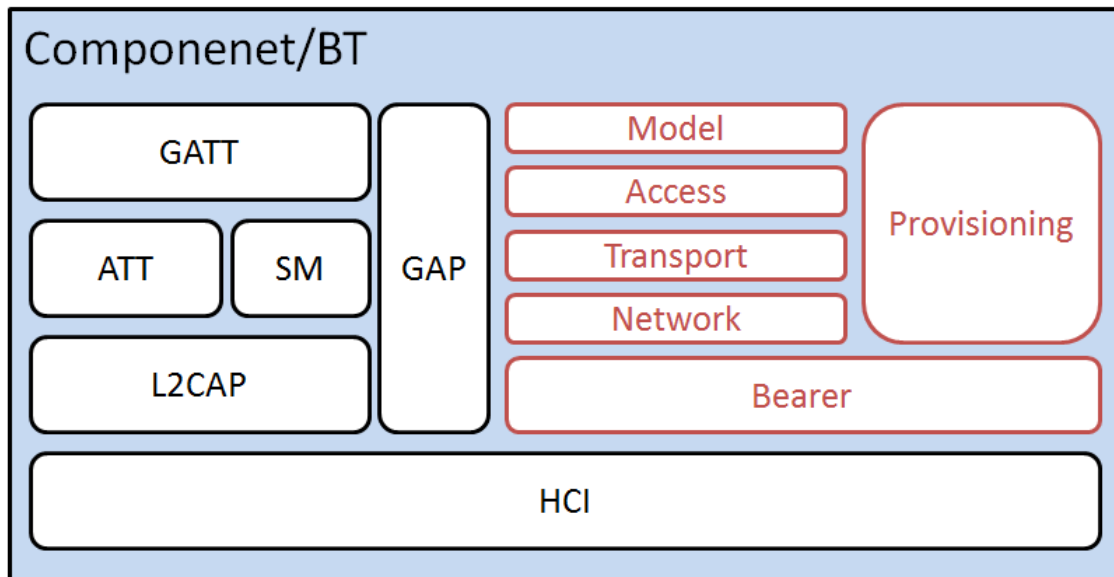
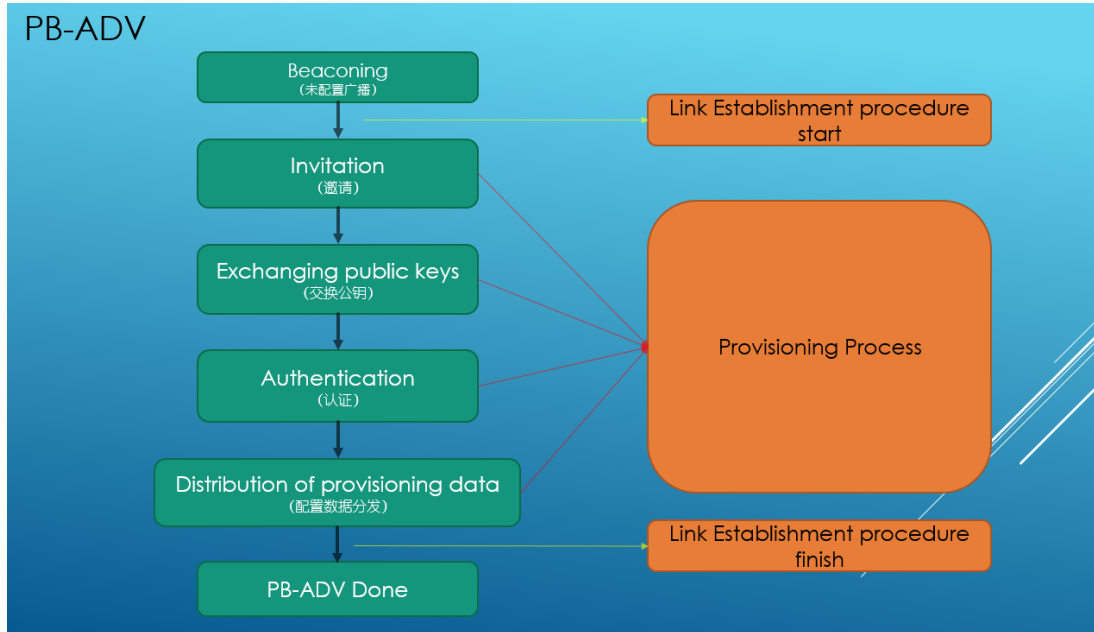


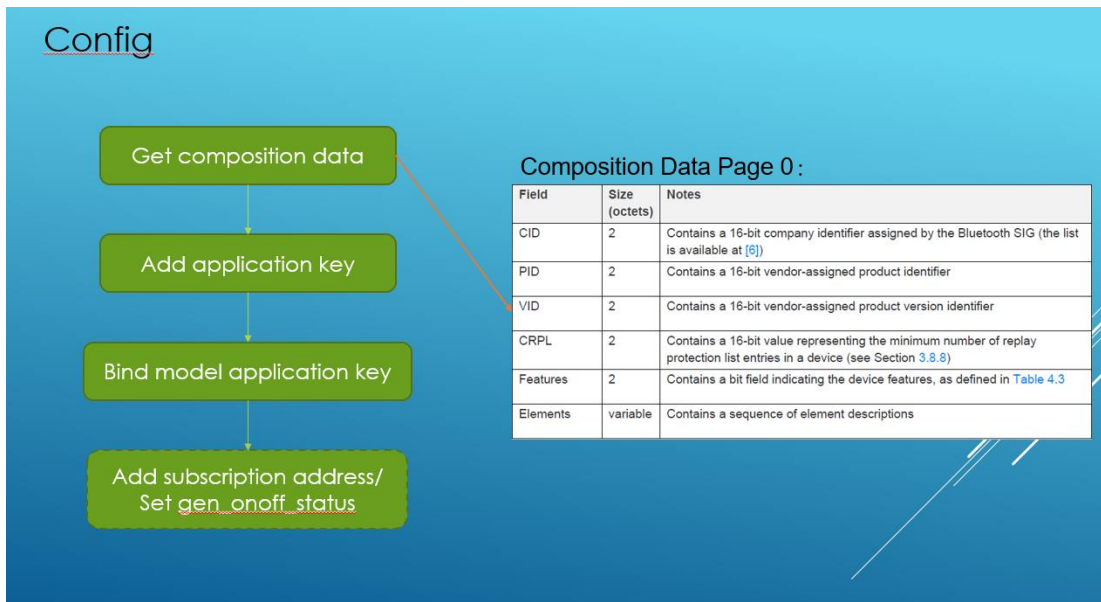
图1 BK3633软件架构中涵盖的Mesh协议栈

1.3. Provisioner 协议说明

配网者的provisioning流程为：PB-ADV，其相应流程如下图：



配网者的config流程如下图所示：



2. 开发者指南

本章节主要描述开发者如何创建自己的Mesh Provisioner应用程序。

2.1. 初始化

系统调用初始化：在Alios初始化完成后，在application_start()中调用provisioner_init()，如图2。

```
int application_start(int argc, char **argv)
{
    /* provisioner initilize */
    provisioner_init();

    BT_INFO("PROVISIONER_BUILD_TIME:%s", __DATE__, "__TIME__");

    //aos_loop_run();

    return 0;
}
```

图2 系统调用初始化

Mesh Provisioner的初始化：对mac地址、composition data、provisioner配置参数等的相应初始化，如图3。

```
void provisioner_init(void)
{
    int ret;

    BT_DBG(">>>init provisioner<<<");
    provisioner_mac_init();

    comp.cid = CONFIG_CID;
    comp.pid = 0;
    comp.vid = 1;
    comp.elem = elements;
    comp.elem_count = provisioner_get_vendor_element_num();

    hci_driver_init();

    ret = bt_enable(_provisioner_ready);
    if (ret) {
        BT_ERR("init err %d", ret);
    }
}
```

图3 Mesh Provisioner的初始化

API	功能说明
provisioner_mac_init()	mac地址初始化
comp	CompositionDataPage0的初始化
hci_driver_init()	Hci驱动注册
_provisioner_ready()	Provisioner的参数配置
bt_enable()	BT的初始化
bt_mesh_init()	Mesh的初始化
bt_mesh_provisioner_enable()	Provisioner使能

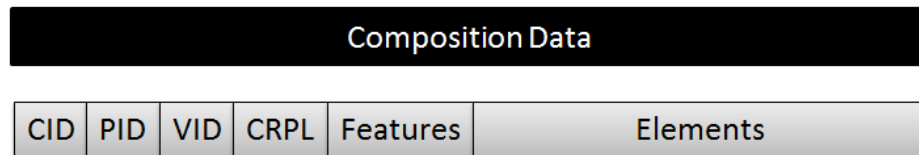


图4 Composition Data Page 0

```
static void _provisioner_ready(int err)
{
    if (err) {
        BT_ERR("BT init err %d", err);
        return;
    }

    err = bt_mesh_init(NULL, &comp, &provisioner);
    if (err) {
        BT_ERR("mesh provisioner init err %d", err);
        return;
    }

    bt_mesh_provisioner_set_dev_uuid_match(0x00,16,temp_uuid,1);
    bt_mesh_provisioner_enable(BT_MESH_PROV_GATT | BT_MESH_PROV_ADV);
    BT_DBG(">>> provisioner enable <<<");
    //create provisioner config thread.
    k_thread_create(&provisioner_config_thread_data, provisioner_config_thread_stack,
        K_THREAD_STACK_SIZEOF(provisioner_config_thread_stack), provisioner_config_thread,
        NULL, NULL, NULL, CONFIG_BT_MESH_PROVISIONER_CONFIG_PRIO, 0, K_NO_WAIT);
}
```

图5 _provisioner_ready()

2.2. 元素注册与模型添加

对于provisioner，需要注册elements，添加configuration client models、SIG Mesh client models，如light client model用于控制灯设备、自定义的vendor model。

```
static struct bt_mesh_elem elements[] = {  
    BT_MESH_ELEM(0, root_models, vnd_models, 0),  
};
```

图6 elements注册

```
static struct bt_mesh_model root_models[] = {  
#ifdef CONFIG_BT_MESH_CFG_CLI  
    BT_MESH_MODEL_CFG_CLI(&cfg_cli),  
    BT_MESH_MODEL(BT_MESH_MODEL_ID_GEN_ONOFF_CLI, NULL, NULL, &onoff_cli),  
#endif  
};
```

图7 root_models配置

```
static struct bt_mesh_model vnd_models[] = {  
//    BT_MESH_MODEL_VND(_company, _id, _op, _pub, _user_data), //example.  
// #ifdef CONFIG_MESH_MODEL_VENDOR_SRV  
//    MESH_MODEL_VENDOR_SRV(&g_elem_state[0]),  
// #endif  
};
```

图8 vendor_models配置

2.3. Provisioner 配置

Provisioner作为mesh网络的发起者，需要配置自身的security materials，如 network keys、application keys、IV index、unicast address。

Security materials	功能说明
network keys	由SDK自身随机生成， 具体可看provisioner_upper_init(*)
application keys	由开发者自己设置， bt_mesh_temp_prov_app_idx_set(*) bt_mesh_provisioner_bind_local_model_app_idx(*)
IV index	
unicast address	


```
static struct bt_mesh_provisioner provisioner = {
    .prov_uuid           = 0,
    .prov_unicast_addr   = 1,
    .prov_start_address   = 2,
    .prov_attention       = 0,
    .prov_algorithm       = 0,
    .prov_pub_key_oob     = 0,
    .prov_pub_key_oob_cb  = 0,
    .prov_static_oob_val  = 0,
    .prov_static_oob_len  = 0,
    .prov_input_num       = provisioner_input_num,
    .prov_output_num      = provisioner_output_num,
    .flags                = 0,
    .iv_index            = 0,
    .prov_link_open       = provisioner_link_open,
    .prov_link_close      = provisioner_link_close,
    .prov_complete        = provisioner_complete,
};
```

图9 provisioner参数配置

2.4. Provisioning

Provisioner会扫描unprovisioned device的UUID，可以选择其中一个来开启provisioning流程。

通过bt_mesh_provisioner_set_dev_uuid_match(offset, len, uuid, prov_flag)来对目标设备的uuid进行设置。

2.5. Provisioned device 配置

当设备完成provisioned后，需要provisioner对其进行configure，provisioner需要获取provisioned device的composition data，来知道设备所支持的models。

当provisioner接收到OP_DEV_COMP_DATA_STATUS后，会提供application key和为必要的设备模型bind application key，以实现后续模型交互数据时的解密。

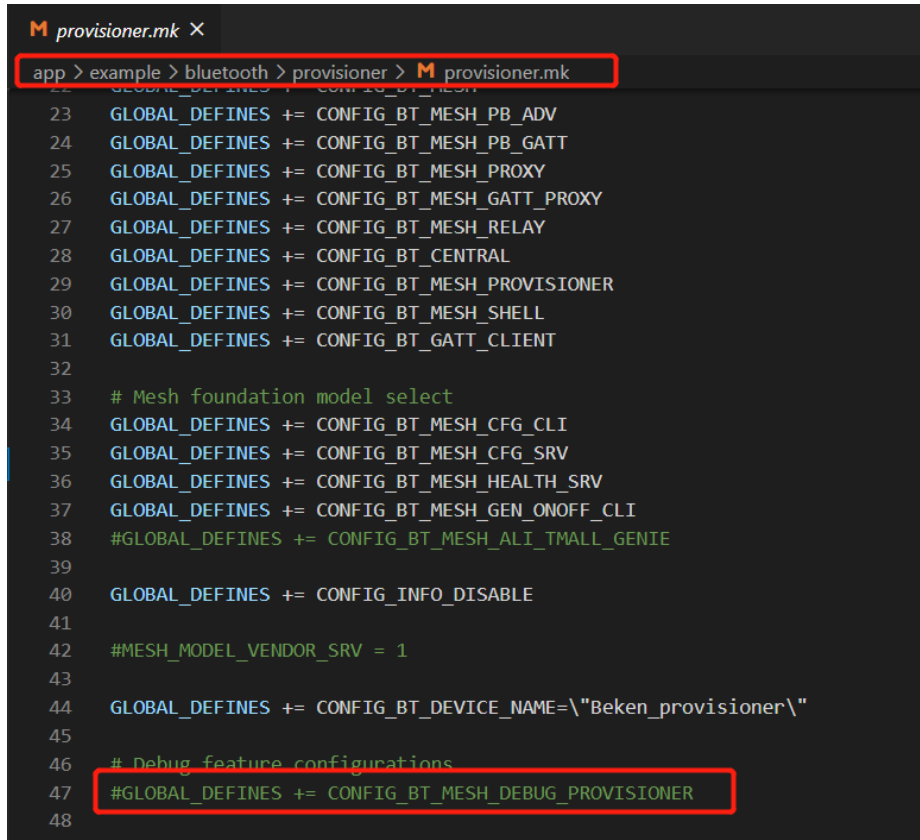
API	功能说明
provisioner_config_comp_data_get(*)	获取composition data
provisioner_config_app_key_add(*)	添加application key
provisioner_config_mod_app_bind(*)	为model绑定application key
provisioner_config_mod_sub_add(*)	为model订阅组播地址

2.6. 日志打印

对于整个配网者工程的日志打印，是划分为两个区域：一个是应用层的日志打印（**application.c**），另外一个为协议层的打印。

应用层日志打印的宏定义如下所示，默认关闭：

CONFIG_BT_MESH_DEBUG_PROVISIONER：application.c上的应用打印。



```

M provisioner.mk
app > example > bluetooth > provisioner > M provisioner.mk
23 GLOBAL_DEFINES += CONFIG_BT_MESH_PB_ADV
24 GLOBAL_DEFINES += CONFIG_BT_MESH_PB_GATT
25 GLOBAL_DEFINES += CONFIG_BT_MESH_PROXY
26 GLOBAL_DEFINES += CONFIG_BT_MESH_GATT_PROXY
27 GLOBAL_DEFINES += CONFIG_BT_MESH_RELAY
28 GLOBAL_DEFINES += CONFIG_BT_MESH_RELAY
29 GLOBAL_DEFINES += CONFIG_BT_MESH_RELAY
30 GLOBAL_DEFINES += CONFIG_BT_MESH_RELAY
31 GLOBAL_DEFINES += CONFIG_BT_MESH_RELAY
32
33 # Mesh foundation model select
34 GLOBAL_DEFINES += CONFIG_BT_MESH_CFG_CLI
35 GLOBAL_DEFINES += CONFIG_BT_MESH_CFG_SRV
36 GLOBAL_DEFINES += CONFIG_BT_MESH_HEALTH_SRV
37 GLOBAL_DEFINES += CONFIG_BT_MESH_GEN_ONOFF_CLI
38 #GLOBAL_DEFINES += CONFIG_BT_MESH_ALI_TMALL_GENIE
39
40 GLOBAL_DEFINES += CONFIG_INFO_DISABLE
41
42 #MESH_MODEL_VENDOR_SRV = 1
43
44 GLOBAL_DEFINES += CONFIG_BT_DEVICE_NAME="Beken_provisioner\"
45
46 # Debug feature configurations
47 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_PROVISIONER
48

```

协议层日志打印的宏定义如下所示，默认关闭：

CONFIG_BT_MESH_DEBUG_PROVISIONER_BEACON:

provisioner_beacon.c的日志打印，主要涉及到配网者的beacon包的广播与扫描；

CONFIG_BT_MESH_DEBUG_PROVISIONER_MAIN:

provisioner_main.c的日志打印，主要涉及到配网者的一些本地应用处理；

CONFIG_BT_MESH_DEBUG_PROVISIONER_PROV:

provisioner_prov.c的日志打印，主要涉及到配网者的provisioning流程；

CONFIG_BT_MESH_DEBUG_PROVISIONER_PROXY:

provisioner_proxy.c的日志打印，主要涉及到配网者的proxy功能的应用；

```
network > bluetooth > bt_mesh > M bt_mesh.mk
56 GLOBAL_INCLUDES += ./inc/ \
57 | | | | ./inc/api
58 | bekencorp, 8 months ago • first commit
59 |
60 GLOBAL_DEFINES += CRC16_ENABLED
61
62 ## BLE Mesh subsystem debug log control macro
63 ## Enable below macros if component-specific debug needed
64 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG
65 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_ACCESS
66 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_TRANS
67 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_NET
68 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_PROV
69 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_PROXY
70 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_FRIEND
71 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_LOW_POWER
72 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_ADV
73 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_BEACON
74 GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_MODEL
75 GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_VENDOR_MODEL
76 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_CRYPTO
77 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_PROVISIONER_BEACON
78 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_PROVISIONER_MAIN
79 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_PROVISIONER_PROV
80 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_PROVISIONER_PROXY
```

本章节主要描述开发者如何使用Mesh Provisioner和Mesh Light这两个应用工程来实现建立Mesh网络。

```

app > example > bluetooth > provisioner > C application > C application.c
411     BT_DBG("Set app key index end!");
412     bt_mesh_temp_prov_app_idx_set(default_app_key, net_index, &app_index, &status);
413     BT_DBG("Bind local model app key index!");
414     bt_mesh_provisioner_bind_local_model_app_idx(provisioner.prov_unicast_addr, BT_MESH_MODEL_ID_GEN_ONOFF_CLI,
415                                                  0xffff, app_index);
416     while(1) {
417         k_sem_take(&provisioner_config_sem, -1);
418         switch (prov_msg.msg_type) {
419             case PROVISIONER_COMP_DATA_GET:
420
421                 BT_DBG("provisioning config start!");
422                 BT_DBG("Get composition data");
423                 provisioner_config_comp_data_get(prov_msg.comp_get_t.netkey_idx, prov_msg.comp_get_t.unicast_addr);
424
425                 BT_DBG("Add app key");
426                 provisioner_config_app_key_add(prov_msg.comp_get_t.netkey_idx, prov_msg.comp_get_t.unicast_addr);
427
428                 BT_DBG("Bind model app key");
429                 provisioner_config_mod_app_bind(prov_msg.comp_get_t.netkey_idx, prov_msg.comp_get_t.unicast_addr,
430                                                  app_index);
431
432                 BT_DBG("Subscrip group addr");
433                 provisioner_config_mod_sub_add(prov_msg.comp_get_t.netkey_idx, prov_msg.comp_get_t.unicast_addr);
434
435                 // BT_DBG("Set node's onoff model!");
436                 // bt_mesh_cfg_cli_gen_onoff_set(net_index, provisioner.prov_unicast_addr, prov_msg.comp_get_t.unicast
437
438                 extern char temp_uuid[16];
439                 temp_uuid[0x0c]++;
440                 bt_mesh_provisioner_set_dev_uuid_match(0x00,16,temp_uuid,1); //for provisioning test.
441                 BT_DBG("provisioning config end, temp_uuid[0x0c]: 0x%x !!", temp_uuid[0x0c]);
442                 break;
443
444             default:
445                 break;
446
447         }

```

分别准备两个**bk3633**开发板。

编译light工程并烧录在其中一个开发板上，可从genie_tri_tuple_get_uuid() api中找到相应的light uuid:

```

genie_app > base > C tri_tuple.c > genie_tri_tuple_get_uuid(void)
72 uint8_t *genie_tri_tuple_get_uuid(void) Yulong, 8 months ago • Add the bk3633 Ali
73 {
74     int i;
75     uint32_t off_set = 0x000;
76     uint8_t addr[6] = {0};
77     uint8_t dummy_addr[6] = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff};
78
79     static_partition_write_addr_head(STATIC_SECTION_MAC);
80
81     if(static_partition_read(STATIC_SECTION_MAC, addr, sizeof(addr)) == 0 &&
82        memcmp(addr, dummy_addr, sizeof(addr)) != 0)
83     {
84         memcpy(g_mac, addr, 6);
85         printf("read sec, addr:%02x : %02x : %02x : %02x : %02x : %02x\n",
86            addr[0], addr[1], addr[2], addr[3], addr[4], addr[5]);
87     }
88     else
89     {
90         memcpy(addr, DEFAULT_MAC, 6);
91         memcpy(g_mac, DEFAULT_MAC, 6);
92         printf("-----\n");
93     }
94
95     printf("%s, addr:%02x : %02x : %02x : %02x : %02x : %02x\n",
96        __func__, addr[0], addr[1], addr[2], addr[3], addr[4], addr[5]);

```

3.3. Mesh Provisioner

对于配网者，用户需要添加应用代码的可在provisioner文件夹下的application.c中添加相应的应用代码，而具体应用则需要根据用户想在provisioning哪个环节中进行添加，下图为provisioning过程中所涉及的环节的相应回调：

```

app > example > bluetooth > provisioner > C application.c > provisioner
209 static struct bt_mesh_provisioner provisioner = {
210     .prov_uuid = 0,
211     .prov_unicast_addr = 1,
212     .prov_start_address = 2,
213     .prov_attention = 0,
214     .prov_algorithm = 0,
215     .prov_pub_key_oob = 0,
216     .prov_pub_key_oob_cb = 0,
217     .prov_static_oob_val = 0,
218     .prov_static_oob_len = 0,
219     .prov_input_num = provisioner_input_num,
220     .prov_output_num = provisioner_output_num,
221     .flags = 0,
222     .iv_index = 0,
223     .prov_link_open = provisioner_link_open,
224     .prov_link_close = provisioner_link_close,
225     .prov_complete = provisioner_complete,
226 };

```

```
app > example > bluetooth > provisioner > C application.c > temp_uuid
71 char temp_uuid[16] = {0x72, 0x62, 0x12, 0x4D, 0x23, 0xDC,
72 | | | | | 0x02, 0x15, 0x64, 0x66, 0x76, 0x6F,
73 | | | | | 0x61, 0x72, 0x63, 0x61}; You, 3 months ago

app > example > bluetooth > provisioner > C application.c > _provisioner_ready(int)
339 static void _provisioner_ready(int err)
340 {
341     if (err) {
342         BT_ERR("BT init err %d", err);
343         return;
344     }
345
346     err = bt_mesh_init(NULL, &comp, &provisioner);
347     if (err) {
348         BT_ERR("mesh provisioner init err %d", err);
349         return;
350     }
351
352     bt_mesh_provisioner_set_dev_uuid_match(0x00,16,temp_uuid,1);
353     bt_mesh_provisioner_enable(BT_MESH_PROV_GATT | BT_MESH_PROV_ADV);
354     BT_DBG(">>> provisioner enable <<<");
355     //create provisioner config thread.
356     k_thread_create(&provisioner_config_thread_data, provisioner_config_thread_stack,
357 | | | | | K_THREAD_STACK_SIZEOF(provisioner_config_thread_stack), provisioner_config_thread,
358 | | | | | NULL, NULL, NULL, CONFIG_BT_MESH_PROVISIONER_CONFIG_PRIO, 0, K_NO_WAIT);
359 | | | | | You, 4 months ago * 1. get a new-built provisioner project;
```

```
network > bluetooth > bt_mesh > M bt_mesh.mk
59
60 GLOBAL_DEFINES += CRC16_ENABLED
61
62 ## BLE Mesh subsystem debug log control macro
63 ## Enable below macros if component-specific debug needed
64 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG
65 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_ACCESS
66 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_TRANS
67 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_NET
68 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_PROV
69 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_PROXY
70 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_FRIEND
71 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_LOW_POWER
72 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_ADV
73 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_BEACON
74 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_MODEL
75 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_CRYPTOT
76 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_PROVISIONER_BEACON
77 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_PROVISIONER_MAIN
78 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_PROVISIONER_PROV
79 #GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_PROVISIONER_PROXY
```

3.4. Provisioning

便于用户观看provisioning的整个过程，我们先给配网者上电，让其先初始化完成，我们可以看到相应的初始化日志如下：

```
ble_handler start!

Initialize RW SW stack

***** appm_init *****
set ble bd addr: AB:23:4D:12:62:72

ble_handler |
ble driver start!

[000100]<V> PROVISIONER_BUILD_TIME:Jan 28 2021,15:28:25

[000140]<V> [_provisioner_ready]>>> provisioner enable <<< 配网者初始化成功

[000140]<V> [provisioner_config_thread]Set app key index end! 配网者自身配置
                                                                appkey成功
[000140]<V> [provisioner_config_thread]Bind local model app key index!
```

等待配网者完成初始化后，我们在给另外的light设备的开发板上电，待light设备完成相应的初始化后，我们便可以从配网者的日志打印上看到相应的provisioning&config过程的日志，具体的日志可实际观看，在此便不截图了。当出现如下日志打印时，表示配网者与light的provisioning&config的整个过程已经完成了：

```
app > example > bluetooth > provisioner > C application.c > ...
415         0xffff, app_index);
416     while(1) {
417         k_sem_take(&provisioner_config_sem, -1);
418         switch (prov_msg.msg_type) {
419             case PROVISIONER_COMP_DATA_GET:
420
421                 BT_DBG("provisioning config start!");
422                 BT_DBG("Get composition data");
423                 provisioner_config_comp_data_get(prov_msg.comp_get_t.netkey_idx, prov_msg.comp_get_t.unicast_addr);
424
425                 BT_DBG("Add app key");
426                 provisioner_config_app_key_add(prov_msg.comp_get_t.netkey_idx, prov_msg.comp_get_t.unicast_addr);
427
428                 BT_DBG("Bind model app key");
429                 provisioner_config_mod_app_bind(prov_msg.comp_get_t.netkey_idx, prov_msg.comp_get_t.unicast_addr,
430                                                 app_index);
431
432                 BT_DBG("Subscrip group addr");
433                 provisioner_config_mod_sub_add(prov_msg.comp_get_t.netkey_idx, prov_msg.comp_get_t.unicast_addr);
434
435                 // BT_DBG("Set node's onoff modell");
436                 // bt_mesh_cfg_cli_gen_onoff_set(net_index, provisioner.prov_unicast_addr, prov_msg.comp_get_t.unicast
437
438                 extern char temp_uuid[16];
439                 temp_uuid[0x0c]++;
440                 bt_mesh_provisioner_set_dev_uuid_match(0x00, 16, temp_uuid, 1); //for provisioning test.
441                 BT_DBG("provisioning config end, temp_uuid[0x0c]: 0x%2x !", temp_uuid[0x0c]);
442                 break;
443
444             default:
445                 break;
446         }
447     }
448 }
```

至此，两个应用工程的演示交互完成。