GigaDevice Semiconductor Inc.

GD32VW553 BLE 开发指南

应用笔记

AN152

1.2 版本

（2024 年 7 月）

目录

图索引

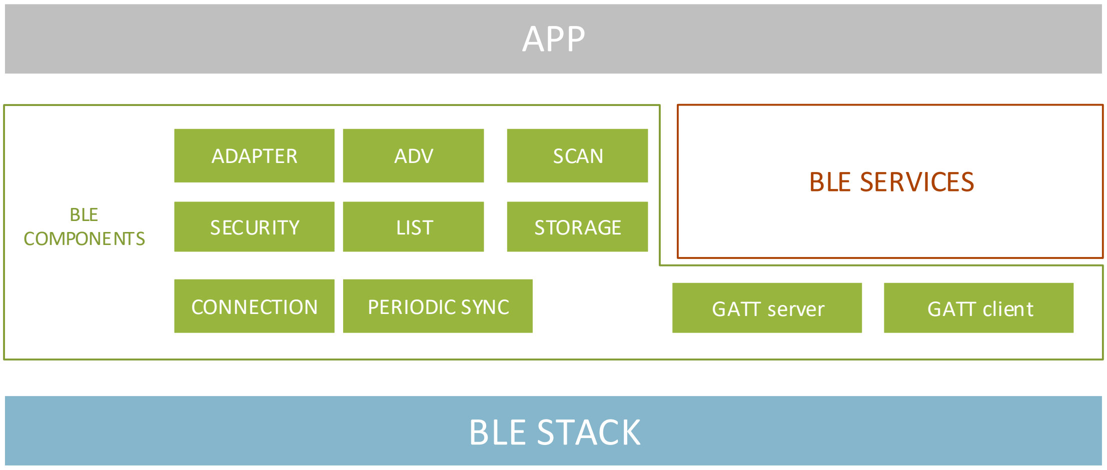
表索引

1. BLE SDK 概述

GD32VW553 系列芯片是以 RISC-V 为内核的 32 位微控制器（MCU），包含了 Wi-Fi4/ Wi-Fi6 及 BLE5.3 连接技术。GD32VW553 Wi-Fi+BLE SDK 集成 Wi-Fi 驱动、BLE 驱动、LwIP TCP/IP 协议栈、MbedTLS 等组件，可使开发者基于 GD32VW553 快速开发物联网应用程序。本应用指南描述了 BLE 软件框架及相关 API 接口，旨在帮助开发者熟悉并使用 BLE API 开发自己的应用程序，Wi-Fi 相关内容请参考《AN158 GD32VW553 Wi-Fi 开发指南》。

1.1. BLE 软件框架

图 1-1. BLE 软件框架图



如*图 1-1. BLE 软件框架图*所示，GD32VW553 BLE 软件部分由 BLE STACK、BLE COMPONENTS、BLE services 和 BLE APP 四个模块组成。

BLE STACK 是对 BLE 协议栈的实现，包含了 GAP、GATT、SMP、L2CAP、HCI 和 LL 等模块。BLE STACK 运行于一个单独的 task，与 BLE COMPONENTS 间通过 TASK message 进行交互，APP 需要通过 BLE COMPONENTS 对 STACK 进行操作。

BLE COMPONENTS 由多个组件构成，和 BLE service、BLE APP 运行在同一个 task，为 APP 提供对 STACK 的控制和状态通知等接口。需要注意的是，BLE 的大部分操作都是异步执行的，APP 需要向各个模块中注册 callback 处理函数，BLE COMPONENTS 会在 callback 函数中通知 APP 调用 API 的执行结果或者上报对端发起的操作请求等内容。各组件之间相互独立，APP 可根据需要选择不同的组件对其进行初始化并注册对应的 callback 函数。

BLE ADAPTER 模块主要提供对本地 BLE 相关属性进行配置和获取等操作的接口，***BLE adapter API*** 介绍了 ADAPTER 模块的 API 使用。

BLE ADV 模块主要提供创建/删除 advertising set，开启/停止发送 advertising packets 等操作的接口，***BLE advertising API*** 介绍了 ADV 模块的 API 使用，***BLE advertising data API*** 提供了一些在 advertising data 中查找特定的 AD type 数据的接口。

BLE SCAN 模块主要提供搜索 advertising set 的接口并且将搜索的结果上报给 APP，***BLE***

scan API介绍了SCAN模块的API使用。

BLE CONNECTION模块主要提供建立连线，获取对端设备信息，获取或设置连线参数等接口，***BLE connection API***介绍了CONNECTION模块的API使用。

BLE SECURITY模块主要提供pairing，authentication，encryption等过程中交互需要的接口，***BLE security API***介绍了SECURITY模块的API使用。

BLE LIST模块主要提供对FAL，RAL，PAL进行操作的接口，包括添加device到list，删除list中的device，清除list等，***BLE list API***介绍了LIST模块的API使用。

BLE PERIODIC SYNC模块主要提供同步periodic advertising，上报接收到的periodic advertising data等接口，***BLE periodic sync API***介绍了PERIODIC SYNC模块的API使用。

BLE STORAGE模块使用flash来存储并管理peer的bond信息和GATT信息等，其中bond信息包括peer_irk, peer_ltk, peer_csrk, local_irk, local_ltk和local_csrk等，***BLE storage API***介绍了STORAGE模块的API使用。

BLE GATT server模块主要提供注册/删除GATT service，向GATT CLIENT发送notification/indication等接口，***BLE gatts API***介绍了GATT server模块的API使用。

BLE GATT client模块主要提供发起GATT discovery，读写对端GATT server中的attribute等接口，***BLE gattc API***介绍了GATT client模块的API使用。

BLE services是基于GATT server、GATT client模块实现的不同service及profile，包括BAS、DIS等，用户也可以根据需要使用GATT server、GATT client接口实现私有service等。

BLE APP层是多个应用的集合，例如blue courier（蓝牙配网）及用户自定义应用等。APP根据不同的需要可以向不同的模块注册callback函数处理相应的消息。

2. BLE API

2.1. BLE adapter API

头文件 ble_adapter.h。

BLE adapter 模块主要提供对本地 BLE 相关属性进行配置和获取等操作的接口。

2.1.1. adapter 消息类型

APP 可以向 BLE adapter 模块注册 callback 函数，BLE 协议栈会通过 callback 函数发送以下的 event message 给 APP。

- BLE_ADP_EVT_ENABLE_CMPL_INFO

该消息会在 BLE adapter 初始化完成后发送，消息数据类型为 ble_adp_info_t，包含是否初始化成功，如果初始化成功也会上报 local version，local IRK 等本地属性。

APP 需在收到该消息且 status 表示初始化成功后才可以对 BLE 进行相关操作。

- BLE_ADP_EVT_RESET_CMPL_INFO

该消息会在 BLE adapter reset 完成后发送，消息数据类型为 uint16_t 表示是否 reset 成功。

- BLE_ADP_EVT_DISABLE_CMPL_INFO

该消息返回 APP 调用 ble_adp_disable API 将 BLE disable 的结果，消息数据类型为 uint16_t 表示是否 disable 成功。

- BLE_ADP_EVT_CHANN_MAP_SET_RSP

该消息返回 APP 调用 ble_adp_chann_map_set API 设置 channel map 的结果，消息数据类型为 uint16_t 表示 channel map 是否设置成功。

- BLE_ADP_EVT _LOC_IRK_SET_RSP

该消息返回 APP 调用 ble_adp_loc_irk_set API 设置 local IRK 的结果，消息数据类型为 uint16_t 表示 local IRK 是否设置成功。

- BLE_ADP_EVT_LOC_ADDR_INFO

该消息是在 localaddress 发生变化，例如 RPAtimeout 后通知给 APP 新的 address 信息，消息数据类型为 ble_gap_local_addr_info_t。

- BLE_ADP_EVT_NAME_SET_RSP

该消息返回 APP 调用 ble_adp_name_set API 设置 local name 的结果，消息数据为 uint16_t 的 status 表示 local name 是否设置成功。

- BLE_ADP_EVT_ADDR_RESLV_RSP

该消息返回 APP 调用 ble_adp_addr_resolve API 对传入的 RPA 进行 reslove 的结果，消息数据类型为 ble_gap_addr_resolve_rsp_t，包含是否 reslove 成功，如果 reslove 成功也会有 reslove 后的 address 以及对应的 IRK 信息。

- BLE_ADP_EVT_RAND_ADDR_GEN_RSP

该消息返回 APP 调用 ble_adp_none_resolvable_private_addr_gen API ，ble_adp_static_random_addr_gen API 或者 ble_adp_resolvable_private_addr_gen API 生成 random address 的结果，消息数据为ble_gap_rand_addr_gen_rsp_t，如果成功生成 random address，也提供对应的 address 信息。

- BLE_ADP_EVT_TEST_TX_RSP

该消息返回 APP 调用 ble_adp_test_tx API 的结果，消息数据类型为 uint16_t 表示 tx test 是否成功开始执行。

- BLE_ADP_EVT_TEST_RX_RSP

该消息返回 APP 调用 ble_adp_test_rx API 的结果，消息数据类型为 uint16_t 表示 rx test 是否成功开始执行。

- BLE_ADP_EVT_TEST_END_RSP

该消息返回 APP 调用 ble_adp_test_end API 的结果，消息数据类型为 ble_gap_test_end_rsp_t，包含是否成功结束 test。

- BLE_ADP_EVT_TEST_RX_PKT_INFO

该消息会在 test rx end 后通知 APP 成功接收到的 packet number，消息数据类型为 ble_gap_test_rx_pkt_info_t。

2.1.2. ble_adp_init

原型：ble_status_t ble_adp_init(void)

功能：初始化 BLE adapter 模块

输入参数：无

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.1.3. ble_adp_callback_register

原型：ble_status_t ble_adp_callback_register(ble_adp_evt_handler_t callback)

功能：注册处理 BLE adapter 消息的 callback 函数，adapter 消息说明见 ***adapter 消息类型***

输入参数：callback，callback 函数指针

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.1.4. ble_adp_callback_unregister

原型：ble_status_t ble_adp_callback_unregister(ble_adp_evt_handler_t callback)

功能：向 BLE adapter 模块取消注册的 callback 函数

输入参数：callback，需要取消的 callback 函数指针

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.1.5. ble_adp_reset

原型：ble_status_t ble_adp_reset(void)

功能：reset BLE 协议栈及各个模块

输入参数：无

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

reset 完成后会有 BLE_ADP_EVT_RESET_CMPL_INFO 消息通知 callback 函数

2.1.6. ble_adp_disable

原型：ble_status_t ble_adp_disable(void)

功能：disable BLE 协议栈及各个模块

输入参数：无

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

disable 完成会有 BLE_ADP_EVT_DISABLE_CMPL_INFO 消息通知 callback 函数

2.1.7. ble_adp_cfg

原型：ble_status_t ble_adp_cfg(ble_adp_config_t *p_adp_config)

功能：配置 BLE adapter

输入参数：p_adp_config，adapter config 结构体指针，可以配置设备的 role，privacy 等属性

如果 config 中 keys_user_mgr 置为 true，则需要 APP 进行 key 的保存和管理，

APP 可以调用 ***BLE storage API*** 节中提供的 storage API 存取或者使用自己需要的方式管理；否则由 ble security 模块管理，APP 不需要再进行相关信息，可以调用 ble_peer_data_bond_load 获取保存的 key 信息

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

配置完成后会有 BLE_ADP_EVT_ENABLE_CMPL_INFO 消息通知 callback 函数

2.1.8. ble_adp_chann_map_set

原型：ble_status_t ble_adp_chann_map_set(uint8_t *p_chann_map)

功能：设置 BLE 可用的 channel map

输入参数：p_chann_map，channel map 数组，长度为 5 bytes，有效 bit 位为低 37 bit，byte 0 的 bit 0 置位表示使用 channel index 0，byte 0 的 bit 1 置位表示使用 channel index 1，以此类推

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

设置完成后会有 BLE_ADP_EVT_CHANN_MAP_SET_RSP 消息通知 callback 函数

2.1.9. ble_adp_loc_irk_set

原型：ble_status_t ble_adp_loc_irk_set(uint8_t *p_irk)

功能：设置 local 使用的 IRK

输入参数：p_irk，需要设置的 IRK 指针，内容长度为 16 bytes

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

设置完成后会有 BLE_ADP_EVT_LOC_IRK_SET_RSP 消息通知 callback 函数

2.1.10. ble_adp_loc_irk_get

原型：ble_status_t ble_adp_loc_irk_get (uint8_t *p_irk)

功能：获取 BLE adapter 使用的 local IRK

输入参数：无

输出参数：p_irk，local IRK 指针，内容长度为 16 bytes，保存获取到的 local IRK 信息

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.1.11. ble_adp_identity_addr_get

原型：ble_status_t ble_adp_identity_addr_get (ble_gap_addr_t *p_id_addr)

功能：获取 BLE adapter 使用的 identity address

输入参数：无

输出参数：p_id_addr，identity address 指针，包括 address type，address 值

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.1.12. ble_adp_name_set

原型：ble_status_t ble_adp_name_set (uint8_t *p_name, uint8_t name_len)

功能：设置 BLE adapter 使用的 device name

输入参数：p_name，device name 指针

name_len，device name 长度

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

设置完成后会有 BLE_ADP_EVT_NAME_SET_RSP 消息通知 callback 函数

2.1.13. ble_adp_local_ver_get

原型：ble_status_t ble_adp_local_ver_get (ble_gap_local_ver_t *p_val)

功能：获取 BLE adapter 版本信息

输入参数：无

输出参数：p_val，local version 结构体指针，包括 hci version，lmp version 等

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.1.14. ble_adp_sugg_dft_data_len_get

原型：ble_status_t ble_adp_sugg_dft_data_len_get(ble_gap_sugg_dft_data_t *p_data)

功能：获取 BLE adapter 默认 transmit data 参数

输入参数：无

输出参数：p_data，suggest data 结构体指针，包括 max tx time，max tx octets

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.1.15. ble_adp_tx_pwr_range_get

原型：ble_status_t ble_adp_tx_pwr_range_get(ble_gap_tx_pwr_range_t *p_val)

功能：获取 BLE adapter transmit power 范围

输入参数：无

输出参数：p_val，tx power range 结构体指针，包括 min tx power，max tx power

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.1.16. ble_adp_max_data_len_get

原型：ble_status_t ble_adp_max_data_len_get(ble_gap_max_data_len_t *p_len)

功能：获取 BLE adapter max data length 信息

输入参数：无

输出参数：p_len，max data length 结构体指针，包括 max tx octets, max tx time,

max rx octets, max rx time

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.1.17. ble_adp_adv_sets_num_get

原型：ble_status_t ble_adp_adv_sets_num_get (uint8_t *p_val)

功能：获取 BLE adapter 支持的最大 advertising set 数目

输入参数：无

输出参数：p_val，advertising set number 指针

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.1.18. ble_adp_addr_resolve

原型：ble_status_t ble_adp_addr_resolve(uint8_t *p_addr, uint8_t *p_irk, uint8_t irk_num)

功能：依次使用提供的 IRK list 中的 key 去解输入的 RPA

输入参数：p_addr，待解的 resolvable private address

p_irk，IRK list 指针

irk_num，IRK list 中 key 的数量

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会 BLE_ADP_EVT_ADDR_RESLV_RSP 消息通知 callback 函数，如果提供的 address 可以解析，消息 data 中会包括解析后的 identity address 和使用的 IRK

2.1.19. ble_adp_static_random_addr_gen

原型：ble_status_t ble_adp_static_random_addr_gen(void)

功能：生成 static random address

输入参数：无

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_ADP_EVT_RAND_ADDR_GEN_RSP 消息通知 callback 函数

2.1.20. ble_adp_resolvable_private_addr_gen

原型：ble_status_t ble_adp_resolvable_private_addr_gen(void)

功能：生成 static resolvable private address

输入参数：无

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_ADP_EVT_RAND_ADDR_GEN_RSP 消息通知 callback 函数

2.1.21. ble_adp_none_resolvable_private_addr_gen

原型：ble_status_t ble_adp_none_resolvable_private_addr_gen(void)

功能：生成 static non-resolvable privatea ddress

输入参数：无

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_ADP_EVT_RAND_ADDR_GEN_RSP 消息通知 callback 函数

2.1.22. ble_adp_test_tx

原型：ble_status_t ble_adp_test_tx(uint8_t chann, uint8_t tx_data_len,

uint8_t tx_pkt_payload, uint8_t phy, int8_t tx_pwr_lvl)

功能：配置 BLE controller 进入 test mode，发送 test packet

输入参数：chann，tx rf channel index，范围：0x00~0x27

tx_data_len，tx 包的长度，范围：0x00~0xFF

tx_pkt_payload，tx 包的类型，范围：0x00~0x07

phy，tx 使用的 PHY，1：1M，2：2M，3：coded S=8，4：coded S=2

tx_pwr_lvl：tx power

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_ADP_EVT_TEST_TX_RSP 消息通知 callback 函数

2.1.23. ble_adp_test_rx

原型：ble_status_t ble_adp_test_rx(uint8_t chann, uint8_t phy, uint8_t modulation_idx)

功能：配置 BLE controller 进入 test mode，接收 test packet

输入参数：chann，rx 使用的 rf channel index，范围：0x00~0x27

phy，rx 使用的 PHY，1：1M，2：2M，3：coded

modulation_idx：BLE controller 是否有 stable modulation index

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_ADP_EVT_TEST_RX_RSP 消息通知 callback 函数

2.1.24. ble_adp_test_end

原型：ble_status_t ble_adp_test_end(void)

功能：配置 BLE controller 退出 test mode

输入参数：无

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_ADP_EVT_TEST_END_RSP 消息通知 callback 函数，如果退出的是 test rx mode，还会有 BLE_ADP_EVT_TEST_RX_PKT_INFO 消息通知 callback 函数成功接收的 packet number

2.2. BLE advertising API

头文件 ble_adv.h。

BLE advertising 模块主要提供创建/删除 advertising set，开启/停止发送 advertising packets 等操作的接口。

2.2.1. advertising 消息类型

■ BLE_ADV_EVT_STATE_CHG

该消息是在 advertising sets 的状态发生变化后通知 APP，advertising sets 的状态定义为 ble_adv_state_t，包括新的 state，state 变化的原因以及发生变化的 adv index。

■ BLE_ADV_EVT_DATA_UPDATE_RSP

该消息是对 APP 调用 ble_adv_data_update 更新正在使用的 advertising set 的 data 回复的 response，消息数据类型为 ble_adv_data_update_rsp_t，包含 update 的 advertising data 类型以及更新是否成功的 status。

■ BLE_ADV_EVT_SCAN_REQ_RCV

如果创建 advertising set 时 enable 了 scan request notification，在打 advertising 后如果收到 scan request packet 就会有该消息通知 APP，消息数据为 ble_adv_scan_req_rcv_t，包含发送 scan request 的设置 address。

2.2.2. ble_adv_init

原型：ble_status_t ble_adv_init(void)

功能：初始化 BLE adv 模块

输入参数：无

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.2.3. ble_adv_deinit

原型：ble_status_t ble_adv_deinit(void)

功能：释放 BLE adv 模块及使用的资源

输入参数：无

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.2.4. ble_adv_create

原型：ble_status_t ble_adv_create(ble_adv_param_t *p_param,

ble_adv_evt_handler_t hdlr, void *p_context)

功能：创建 BLE advertising set

输入参数：p_param，advertising 参数结构体指针，可配置 adv type，interval，phy 等参数

hdlr，注册该 adv 相关消息的处理函数，adv 消息的说明见 ***advertising 消息类型***。

p_context，可用于额外回传至消息处理函数中的参数

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

在该 advertising set create 成功后会有 BLE_ADV_EVT_STATE_CHG 消息发送到注册的消息处理函数，state 为 BLE_ADV_STATE_CREATE，同时在该消息中可以获得 adv index 在之后的 API 中使用

2.2.5. ble_adv_start

原型：ble_status_t ble_adv_start(uint8_t adv_idx, ble_adv_data_set_t *p_adv_data,

ble_adv_data_set_t *p_scan_rsp_data, ble_adv_data_set_t *p_per_adv_data)

功能：设置 advertising set 数据并开始发送 advertising packet

输入参数：adv_idx，advertising index

p_adv_data，advertising data 结构体指针，data 可以用配置参数由 ble adv 模块生成的方式也可以用调用者直接设置内容的方式

p_scan_rsp_data，scan response data 结构体指针，create 的 advertising set 为 scannable advertising 时需要设置

p_per_adv_data，periodic advertising data 结构体指针，create 的 advertising set 为 periodic advertising 时需要设置

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

调用该函数后会有 BLE_ADV_EVT_STATE_CHG 消息发送到 create advertising 时注册的消息处理函数中，根据设置的 advertising data 不同会有 state 为

BLE_ADV_STATE_ADV_DATA_SET，BLE_ADV_STATE_SCAN_RSP_DATA_SET 或者 BLE_ADV_STATE_PER_ADV_DATA_SET 的消息，最后还会一个 state 为 BLE_ADV_STATE_START 的消息

2.2.6. ble_adv_restart

原型：ble_status_t ble_adv_restart(uint8_t adv_idx)

功能：在 advertising set 被停止后重新开始发送 advertising packet

输入参数：adv_idx，advertising index

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

成功开始发送 advertising packet 后会有 BLE_ADV_EVT_STATE_CHG

消息发送到调用 ble_adv_create API 时注册的消息处理函数，

state 为 BLE_ADV_STATE_START

2.2.7. ble_adv_stop

原型：ble_status_t ble_adv_stop(uint8_t adv_idx)

功能：停止发送 advertising packet

输入参数：adv_idx，advertising index

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

advertising set 停止发送后会有 BLE_ADV_EVT_STATE_CHG 消息发送到

调用 ble_adv_create API 时注册的消息处理函数，

state 为 BLE_ADV_STATE_CREATE

2.2.8. ble_adv_remove

原型：ble_status_t ble_adv_remove(uint8_t adv_idx)

功能：删除不在发送 advertising packet 的 advertising set，

如果正在发送 advertising packet，即 state 为 BLE_ADV_STATE_START，

需先调用 ble_adv_stop 将其 stop 后再调用该函数将其 remove。

输入参数：adv_idx，advertising index

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

2.2.9. ble_adv_data_update

原型：ble_status_t ble_adv_data_update(uint8_t adv_idx, ble_adv_data_set_t *p_adv_data,

ble_adv_data_set_t *p_scan_rsp_data, ble_adv_data_set_t *p_per_adv_data)

功能：更新正在发送 advertising packet，即 state 为 BLE_ADV_STATE_START 的 advertising set 的 adv data、scan response data、periodic adv data

输入参数：adv_idx，advertising index

p_adv_data，advertising data 结构体指针

p_scan_rsp_data，scan response data 结构体指针

p_per_adv_data，periodic advertising data 结构体指针

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_ADV_EVT_DATA_UPDATE_RSP 消息通知到注册的 callback 函数

2.3. BLE advertising data API

头文件 ble_adv_data.h

BLE advertising data 模块主要提供从 advertising data 中查找指定的 ad type 的接口。

2.3.1. ble_adv_find

原型：uint8_t *ble_adv_find(uint8_t *p_data, uint16_t data_len, uint8_t ad_type,

uint8_t *p_len)

功能：在 advertising data 中寻找指定 ad type 的数据

输入参数：p_data，待查找的 advertising data 地址

data_len，待查找的 advertising data 长度

ad_type，需要查找的 ad type

输出参数：p_len，查找到的对应 type 的 data value 长度

返回值：查找到的对应 type 的 data value 地址，若未找到返回 NULL

2.3.2. ble_adv_cmpl_name_find

原型：bool ble_adv_cmpl_name_find(uint8_t *p_data, uint16_t data_len,

uint8_t *p_name, uint16_t name_len)

功能：在 advertising data 中寻找是否存在指定的 complete name

输入参数：p_data，待查找的 advertising data 地址

data_len，待查找的 advertising data 长度

p_name，需要查找的 complete name 的地址

name_len，需要查找的 complete name 的长度

输出参数：无

返回值：返回 true 表示在 advertising data 中可以找到指定的 complete name，否则返回 false

2.3.3. ble_adv_short_name_find

原型：bool ble_adv_short_name_find(uint8_t *p_data, uint16_t data_len,

uint8_t *p_name, uint16_t name_len_min)

功能：在 advertising data 中寻找是否存在指定的 short name

输入参数：p_data，待查找的 advertising data 地址

data_len，待查找的 advertising data 长度

p_name，需要查找的 short name 的地址

name_len_min，short name 需要匹配的最小长度

输出参数：无

返回值：返回 true 表示在 advertising data 中可以找到指定的 short name，否则返回 false

2.3.4. ble_adv_svc_uuid_find

原型：bool ble_adv_svc_uuid_find(uint8_t *p_data, uint16_t data_len, ble_uuid_t *p_uuid)

功能：在 advertising data 中寻找是否存在指定的 service uuid

输入参数：p_data，待查找的 advertising data 地址

data_len，待查找的 advtising data 长度

p_uuid，需要查找的 uuid 结构体指针，包括 uuid 长度及 uuid 内容

输出参数：无

返回值：返回 true 表示在 advertising data 中可以找到指定的 service uuid，否则返回 false

2.3.5. ble_adv_appearance_find

原型：bool ble_adv_appearance_find(uint8_t *p_data, uint16_t data_len,
uint16_t appearance)

功能：在 advertising data 中寻找是否存在指定的 appearance

输入参数：p_data，待查找的 advertising data 地址

data_len，待查找的 advertising data 长度

appearance，需要查找的 appearance 值

输出参数：无

返回值：返回 true 表示在 advertising data 中可以找到指定的 appearance，否则返回 false

2.4. BLE scan API

头文件 ble_scan.h。

BLE scan 模块主要提供搜索 advertising data 接口并将搜索的结果上报。

2.4.1. scan 消息类型

APP 可以向 BLE scan 模块注册 callback 函数，BLE 协议栈会通过 callback 函数发送以下的 event message 给 APP。

- BLE_SCAN_EVT_STATE_CHG

该消息会在 scan 状态发生变化时通知到 callback 函数，消息数据类型为 ble_scan_state_chg_t，包含当前的 scan 状态和变化的原因。

- BLE_SCAN_EVT_ADV_RPT

该消息会在 scan 到 advertising packet 后通知 APP 收到的 data 内容，消息数据类型为 ble_gap_adv_report_info_t，该结构体中包含收到的 advertising packet 类型，advertiser 的地址，advertising sid，data 等内容。

2.4.2. ble_scan_init

原型：ble_status_t ble_scan_init(ble_gap_local_addr_type_t own_addr_type)

功能：初始化 BLE scan 模块

输入参数：own_addr_type，scan 过程中使用的 local address type

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.4.3. ble_scan_reinit

原型：ble_status_t ble_scan_reinit(ble_gap_local_addr_type_t own_addr_type)

功能：重新初始化 BLE scan 模块

输入参数：own_addr_type，scan 过程中使用的 local address type

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.4.4. ble_scan_callback_register

原型：ble_status_t ble_scan_callback_register(ble_scan_evt_handler_t callback)

功能：注册处理 BLE scan 消息的 callback 函数

输入参数：callback，处理 BLE scan 消息的函数，scan 消息的说明见 ***scan 消息类型***

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.4.5. ble_scan_callback_unregister

原型：ble_status_t ble_scan_callback_unregister(ble_scan_evt_handler_t callback)

功能：向 BLE scan 模块取消注册的 callback 函数

输入参数：callback，需要取消的 callback 函数指针

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.4.6. ble_scan_enable

原型：ble_status_t ble_scan_enable(void)

功能：开启 BLE 扫描，扫到的设备会由 BLE_SCAN_EVT_ADV_RPT 消息通知 callback 函数

输入参数：无

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

enable 完成后会有 BLE_SCAN_EVT_STATE_CHG 消息通知 callback 函数，state 为 BLE_SCAN_STATE_ENABLED

2.4.7. ble_scan_disable

原型：ble_status_t ble_scan_disable(void)

功能：结束 BLE 扫描

输入参数：无

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

disable 完成后会有 BLE_SCAN_EVT_STATE_CHG 消息通知 callback 函数，state 为 BLE_SCAN_STATE_DISABLED

2.4.8. ble_scan_param_set

原型：ble_status_t ble_scan_param_set (ble_gap_scan_param_t *p_param)

功能：设置 BLE 扫描参数

输入参数：p_param，扫描参数结构体指针，包括 scan type，interval，window 等

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.5. BLE connection API

头文件 ble_conn.h。

BLE connection 模块主要提供建立连线，获取对端设备信息，获取或设置连线参数等接口。

2.5.1. connection 消息类型

APP 可以向 BLE connection 模块注册 callback 函数，BLE 协议栈会通过 callback 函数发送以下的 event message 给 APP。

- BLE_CONN_EVT_INIT_STATE_CHG

该消息会在主动建立连线过程中状态发生变化时通知 callback 函数，数据类型为 ble_init_state_chg_t，包含当前的 state，state 变化的原因以及是否使用 filter accept list。

- BLE_CONN_EVT_STATE_CHG

该消息会在连线状态发生变化后通知 callback 函数，数据类型为 ble_conn_state_chg_t，包

含新的 state，在 state 为 BLE_CONN_STATE_CONNECTED 时还会包含结构体为 ble_gap_conn_info_t 的连线相关信息，state 为 BLE_CONN_STATE_DISCONNECTD 还会包含结构体为 ble_gap_disconn_info_t 的断线相关信息。

■ BLE_CONN_EVT_DISCONN_FAIL

该消息会在主动发起断线失败时通知 callback 函数，数据类型为 ble_conn_disconn_fail_t，包含断线失败原因等。

■ BLE_CONN_EVT_PEER_NAME_GET_RSP

该消息返回 APP 调用 ble_conn_peer_name_get 获取对端 GATT database 中 name 信息的结果，消息数据类型为 ble_gap_peer_name_get_rsp_t，包含获取 attribute 的 status，如果 status 为 BLE_ERR_NO_ERROR，还包含 attribute handle，name 长度和 name 内容等。

■ BLE_CONN_EVT_PEER_VERSION_GET_RSP

该消息返回 APP 调用 ble_conn_peer_version_get 获取对端版本信息的结果，消息数据类型为 ble_gap_peer_ver_get_rsp_t，包含获取 version 的 status，如果 status 为 BLE_ERR_NO_ERROR，还包含 company id，lmp version，lmp subversion 等。

■ BLE_CONN_EVT_PEER_FEATS_GET_RSP

该消息返回 APP 调用 ble_conn_peer_feats_get 获取对端 supported features 信息的结果，消息数据类型为 ble_gap_peer_feats_get_rsp_t，包含获取的 status，如果 status 为 BLE_ERR_NO_ERROR，还包含对端支持的 feature 数组等。

■ BLE_CONN_EVT_PEER_APPEARANCE_GET_RSP

该消息返回 APP 调用 ble_conn_peer_appearance_get 获取对端 GATT database 中 appearance 信息的结果，消息数据类型为 ble_gap_peer_appearance_get_rsp_t，包含获取 attribute 的 status，如果 status 为 BLE_ERR_NO_ERROR，还包含 attribute handle，appearance 等内容。

■ BLE_CONN_EVT_PEER_SLV_PRF_PARAM_GET_RSP

该消息返回 APP 调用 ble_conn_peer_slave_prefer_param_get 获取对端 GATT database 中 slave preferred parameter 这个 attribute 的信息的结果，消息数据类型为 ble_gap_slave_prefer_param_get_rsp_t，包含获取 attribute 的 status，如果 status 为 BLE_ERR_NO_ERROR，还包含 attribute handle，slave preferred connection interval、latency 等内容。

■ BLE_CONN_EVT_PEER_ADDR_RESLV_GET_RSP

该消息返回 APP 调用 ble_conn_peer_addr_resolution_support_get 获取对端 GATT database 中 central address resolution support 这个 attribute 的信息的结果，消息数据类型为 ble_gap_peer_addr_resol_get_rsp_t，包含获取 attribute 的 status，如果 status 为 BLE_ERR_NO_ERROR，还包含 attribute handle，central address resolution support 等内容。

■ BLE_CONN_EVT_PEER_RPA_ONLY_GET_RSP

该消息返回 APP 调用 ble_conn_peer_rpa_only_get 获取对端 GATT database 中 resolvable private address only 这个 attribute 的信息的结果，消息数据类型为 ble_gap_peer_rpa_only_get_rsp_t，包含获取 attribute 的 status，如果 status 为 BLE_ERR_NO_ERROR，还包含 attribute handle，resolvable private address only 等内容。

- BLE_CONN_EVT_PEER_DB_HASH_GET_RSP

该消息返回 APP 调用 ble_conn_peer_db_hash_get 获取对端 GATT database 中 database hash 这个 attribute 信息的结果，消息数据类型为 ble_gap_peer_db_hash_get_rsp_t，包含获取 attribute 的 status，如果 status 为 BLE_ERR_NO_ERROR，还包含 attribute handle，database hash 等内容。

- BLE_CONN_EVT_PING_TO_VAL_GET_RSP

该消息返回 APP 调用 ble_conn_ping_to_get 获取 BLE link ping timeout 值的结果，消息数据类型为 ble_gap_ping_tout_get_rsp_t，包含获得的 status，如果 status 为 BLE_ERR_NO_ERROR，还包含 ping timeout 值。

- BLE_CONN_EVT_PING_TO_INFO

该消息是在 ping timeout 发生后主动通知 APP，消息数据类型为 ble_gap_ping_tout_info_t，包含发生 ping timeout 的 connection index。

- BLE_CONN_EVT_PING_TO_SET_RSP

该消息返回 APP 调用 ble_conn_ping_to_set 设置 ping timeout 值的结果，消息数据类型为 ble_gap_ping_tout_set_rsp_t，包含设置的 status 等内容。

- BLE_CONN_EVT_RSSI_GET_RSP

该消息返回 APP 调用 ble_conn_rssi_get 获取对应连线的最近成功收到的一笔包的 RSSI 的结果，消息数据类型为 ble_gap_rssi_get_rsp_t，包含获取的 status，如果 status 为 BLE_ERR_NO_ERROR，还包含 RSSI 等内容。

- BLE_CONN_EVT_CHANN_MAP_GET_RSP

该消息返回 APP 调用 ble_conn_chann_map_get 获取对应连线使用的 channel map 的结果，消息数据类型为 ble_gap_chann_map_get_rsp_t，包含获取的 status，如果 status 为 BLE_ERR_NO_ERROR，还包含 channel map 数组信息。

- BLE_CONN_EVT_NAME_GET_IND

该消息是在对端设备要获取本地 name 时通知 APP，消息数据类型为 ble_gap_name_get_ind_t，包含本次获取 name 的起始 offset 及最大的 name length，APP 可以调用 ble_conn_name_get_cfm 进行回复。

- BLE_CONN_EVT_APPEARANCE_GET_IND

该消息是在对端设备要获取本地 appearance 时通知 APP，消息数据类型为 ble_gap_appearance_get_ind_t，APP 可以调用 ble_conn_appearance_get_cfm 进行回复。

- BLE_CONN_EVT_SLAVE_PREFER_PARAM_GET_IND

该消息是在对端设备要获取本地的 slave preferred parameter 属性时通知 APP，消息数据类型为 ble_gap_slave_prefer_param_get_ind_t，APP 可以调用 ble_conn_slave_prefer_param_get_cfm 进行回复。

- BLE_CONN_EVT_NAME_SET_IND

该消息是在对端设备要设置本地 name 时通知 APP，消息数据类型为 ble_gap_name_set_ind_t，包含要设置的 name 长度及 name 内容，APP 可以调用 ble_conn_name_set_cfm 进行回复。

- BLE_CONN_EVT_APPEARANCE_SET_IND

该消息是在对端设备要设置本地 appearance 时通知 APP，消息数据类型为 ble_gap_appearance_set_ind_t，包含需要设置的 appearance 值，APP 可以调用 ble_conn_appearance_set_cfm 进行回复。

- BLE_CONN_EVT_PARAM_UPDATE_IND

该消息是在对端发起 connection parameter update 时通知 APP，消息数据类型为 ble_gap_conn_param_update_ind_t，包含对端希望更新的 connection interval，latency，supervision timeout 等参数，APP 可以调用 ble_conn_param_update_cfm 进行回复。

- BLE_CONN_EVT_PARAM_UPDATE_RSP

该消息返回 APP 调用 ble_conn_param_update_req 发起 connection parameter update 的结果，消息类型为 ble_gap_conn_param_update_rsp_t，包含 update 的 status。

- BLE_CONN_EVT_PARAM_UPDATE_INFO

该消息是在对端或者本地发起的 connection parameter update 完成后通知 APP，消息数据类型为 ble_gap_conn_param_info_t，包含 update 后使用的 connection interval，latency 和 supervision timeout 等内容。

- BLE_CONN_EVT_PKT_SIZE_SET_RSP

该消息返回 APP 调用 ble_conn_pkt_size_set 设置本地发送的数据包大小的结果，消息数据类型为 ble_gap_pkt_size_set_rsp_t，包含设置的 status。

- BLE_CONN_EVT_PKT_SIZE_INFO

该消息是在对端或者本地发起 packet size 更新完成后通知 APP，消息数据类型为 ble_gap_pkt_size_info_t，包含 max tx octets，max tx time，max rx octets，max rx time。

- BLE_CONN_EVT_PHY_GET_RSP

该消息返回 APP 调用 ble_conn_phy_get 获取连线使用的 PHY 信息的结果，消息数据类型为 ble_gap_phy_get_rsp_t，包含获取的 status。

- BLE_CONN_EVT_PHY_SET_RSP

该消息返回 APP 调用 ble_conn_phy_set 设置连线使用的 PHY 的结果，消息数据类型为 ble_gap_phy_set_rsp_t，包含设置的 status。

- BLE_CONN_EVT_PHY_INFO

该消息是在 APP 获取连线 PHY 信息，APP 或者对端设置连线 PHY 完成后通知 APP 当前使用的 PHY 的信息，消息数据类型为 ble_gap_phy_info_t，包含当前连线的 tx PHY，rx PHY 信息。

- BLE_CONN_EVT_LOC_TX_PWR_GET_RSP

该消息返回 APP 调用 ble_conn_local_tx_pwr_get 获取本地 transmit power 的结果，消息数据类型为 ble_gap_local_tx_pwr_get_rsp_t，包含获取的 status，如果 status 为 BLE_ERR_NO_ERROR，还包含获取的 PHY，对应 PHY 上当前使用的 transmit power 和最大的 transmit power。

- BLE_CONN_EVT_PEER_TX_PWR_GET_RSP

该消息返回 APP 调用 ble_conn_peer_tx_pwr_get 获取对端 transmit power 的结果，消息数据类型为 ble_gap_peer_tx_pwr_get_rsp_t，包含获取的 status，如果 status 为 BLE_ERR_NO_ERROR，还包含获取的 PHY，对应 PHY 上对端使用的 transmit power 及 power flags。

- BLE_CONN_EVT_TX_PWR_RPT_CTRL_RSP

该消息返回 APP 调用 ble_conn_tx_pwr_report_ctrl 设置 transmit power report 的结果，消息数据类型为 ble_gap_tx_pwr_report_ctrl_rsp_t，包含设置的 status。

- BLE_CONN_EVT_LOC_TX_PWR_RPT_INFO

该消息会在 APP 调用了 ble_conn_tx_pwr_report_ctrl enable 了 local report 且 local transmit power 发生变化后通知 APP，消息数据类型为 ble_gap_tx_pwr_report_info_t，包含本地 report 的 PHY，对应 PHY 上的 transmit power，power flags 及发生变化的 transmit power delta。

- BLE_CONN_EVT_PEER_TX_PWR_RPT_INFO

该消息会在 APP 调用了 ble_conn_tx_pwr_report_ctrl enable 了 peer report 且对端 transmit power 发生变化后通知 APP，消息数据类型为 ble_gap_tx_pwr_report_info_t，包含对端 report 的 PHY，对应 PHY 上的 transmit power，power flags 及发生变化的 transmit power delta。

- BLE_CONN_EVT_PATH_LOSS_CTRL_RSP

该消息返回 APP 调用 ble_conn_path_loss_ctrl 设置 path loss 的结果，消息数据类型为 ble_gap_path_loss_ctrl_rsp_t，包含设置的 status。

- BLE_CONN_EVT_PATH_LOSS_THRESHOLD_INFO

该消息会在 APP 调用 ble_conn_path_loss_ctrl 设置了 path loss 后 path loss zone 发生变化的时候通知 APP，消息数据类型为 ble_gap_path_loss_threshold_info_t，包含当前的 path loss 值及所处的 zone 信息。

- BLE_CONN_EVT_PER_SYNC_TRANS_RSP

该消息返回 APP 调用 ble_conn_per_adv_sync_trans 将 periodic advertising sync transfer 到对端设备的结果，消息类型为 ble_gap_per_adv_sync_trans_rsp_t，包含 transfer 是否成功的 status。

2.5.2. ble_conn_init

原型：ble_status_t ble_conn_init(void)

功能：初始化 BLE connection 模块

输入参数：无

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.5.3. ble_conn_callback_register

原型：ble_status_t ble_conn_callback_register(ble_conn_evt_handler_t callback)

功能：注册处理 BLE connection 消息的 callback 函数

输入参数：callback，处理 BLE connection 消息的函数，connection 消息的说明

见 ***connection 消息类型***

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.5.4. ble_conn_callback_unregister

原型：ble_status_t ble_conn_callback_unregister(ble_conn_evt_handler_t callback)

功能：向 BLE connection 模块取消注册的 callback 函数

输入参数：callback，处理 BLE connection 消息的函数

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.5.5. ble_conn_connect

原型：ble_status_t ble_conn_connect(ble_gap_init_param_t *p_param,

ble_gap_local_addr_type_t own_addr_type,

ble_gap_addr_t *p_peer_addr_info, bool use_wl)

功能：发起 BLE 连线

输入参数：p_param，发起连线时的参数结构体指针，包括connection interval、window 等

own_addr_type，建立连线时使用的 local address type

p_peer_addr_info，对端设备地址信息指针

use_wl，是否使用 FAL，如果使用，则会和FAL 中的设备进行连线，而非

p_peer_addr_info 指定的 address

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的error code

连接成功后会有 BLE_CONN_EVT_STATE_CHG 消息通知 callback 函数，

state 为 BLE_CONN_STATE_CONNECTED，connection info 中包含的

connection index 可用于后续操作

2.5.6. ble_conn_disconnect

原型：ble_status_t ble_conn_disconnect(uint8_t conidx, uint16_t reason)

功能：断开 BLE 连线

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

reason，断开连线的原因，可使用BLE_ERROR_HL_TO_HCI

(BLE_LL_ERR_xxx)，BLE_LL_ERR_xxx 为 ble_err_t 中 LL group 的 error code

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的error code

连线断开后会有 BLE_CONN_EVT_STATE_CHG 消息通知 callback 函数，state 为 BLE_CONN_STATE_DISCONNECTED

2.5.7. ble_conn_connect_cancel

原型：ble_status_t ble_conn_connect_cancel(void)

功能：取消正在发起的BLE 连线

输入参数：无

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的error code

2.5.8. ble_conn_sec_info_set

原型：ble_status_t ble_conn_sec_info_set(uint8_t conidx, uint8_t *p_local_csrk, uint8_t *p_peer_csrk, uint8_t pairing_lvl, uint8_t enc_key_present)

功能：如果由 APP 管理 key 信息，在收到 BLE_CONN_EVT_STATE_CHG 消息且 state 为 BLE_CONN_STATE_CONNECTED 后需要调用该接口将 key 信息传递给 BLE stack

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

p_local_csrk，local CSRK

p_peer_csrk，对端的 CSRK

pairing_lvl，pairing level

enc_key_present，encryption key 是否存在

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

2.5.9. ble_conn_peer_name_get

原型：ble_status_t ble_conn_peer_name_get(uint8_t conidx)

功能：获取已建立连线的对端设备 GATT database 中的名字

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_CONN_EVT_PEER_NAME_GET_RSP 消息通知 callback 函数

2.5.10. ble_conn_peer_feats_get

原型：ble_status_t ble_conn_peer_feats_get(uint8_t conidx)

功能：获取已建立连线的对端设备支持的 feature

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_CONN_EVT_PEER_FEATS_GET_RSP 消息通知 callback 函数

2.5.11. ble_conn_peer_appearance_get

原型：ble_status_t ble_conn_peer_appearance_get(uint8_t conidx)

功能：获取已建立连线的对端设备 GATT database 中的 appearance

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_CONN_EVT_PEER_APPEARANCE_GET_RSP 消息通知 callback 函数

2.5.12. ble_conn_peer_version_get

原型：ble_status_t ble_conn_peer_version_get(uint8_t conidx)

功能：获取已建立连线的对端设备的版本信息

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_CONN_EVT_PEER_VERSION_GET_RSP 消息通知 callback 函数

2.5.13. ble_conn_peer_slave_prefer_param_get

原型：ble_status_t ble_conn_peer_slave_prefer_param_get(uint8_t conidx)

功能：获取已建立连线的对端设备 GATT database 中的 slave prefer parameters 属性

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_CONN_EVT_PEER_SLV_PRF_PARAM_GET_RSP 消息通知 callback 函数

2.5.14. ble_conn_peer_addr_resolution_support_get

原型：ble_status_t ble_conn_peer_addr_resolution_support_get(uint8_t conidx)

功能：获取已建立连线的对端设备 GATT database 中的 address resolution support 属性

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_CONN_EVT_PEER_ADDR_RESLV_GET_RSP 消息通知 callback 函数

2.5.15. ble_conn_peer_rpa_only_get

原型：ble_status_t ble_conn_peer_rpa_only_get(uint8_t conidx)

功能：获取已建立连线的对端设备 GATT database 中的 RPA only 属性

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后有 BLE_CONN_EVT_PEER_RPA_ONLY_GET_RSP 消息通知 callback 函数

2.5.16. ble_conn_peer_db_hash_get

原型：ble_status_t ble_conn_peer_db_hash_get(uint8_t conidx)

功能：获取已建立连线的对端设备 GATT database 中的 database hash 属性

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后有 BLE_CONN_EVT_PEER_DB_HASH_GET_RSP 消息通知 callback 函数

2.5.17. ble_conn_phy_get

原型：ble_status_t ble_conn_phy_get(uint8_t conidx)

功能：获取已建立连线正在使用的 PHY

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

执行后会有 BLE_CONN_EVT_PHY_GET_RSP 消息通知 callback 函数，如果成功获取到还会有 BLE_CONN_EVT_PHY_INFO 消息通知 callback 函数

2.5.18. ble_conn_phy_set

原型：ble_status_t ble_conn_phy_set(uint8_t conidx, uint8_t tx_phy, uint8_t rx_phy,

uint8_t phy_opt)

功能：设置已建立连线使用的 PHY

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

tx_phy，tx 使用的 PHY bitfield，由 ble_gap_le_phy_bf_t 组合而成

rx_phy，rx 使用的 PHY bitfield，由 ble_gap_le_phy_bf_t 组合而成

phy_opt，如果使用 coded PHY，可以设置是否更倾向使用 S=2 或 S=8

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

执行后会有 BLE_CONN_EVT_PHY_SET_RSP 消息通知 callback 函数，

在 PHY 设置完成后还会有 BLE_CONN_EVT_PHY_INFO 消息通知 callback 函数

2.5.19. ble_conn_pkt_size_set

原型：ble_status_t ble_conn_pkt_size_set(uint8_t conidx, uint16_t tx_octets,

uint16_t tx_time)

功能：设置已建立连线在 transmit 时可使用的最大的 packet size

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

tx_octets，tx packet 最大的 octet 数

tx_time，tx packet 最大发送时间

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

执行后会有 BLE_CONN_EVT_PKT_SIZE_SET_RSP 消息通知 callback 函数，

在 packet size 设置完成后会有 BLE_CONN_EVT_PKT_SIZE_INFO 消息

通知 callback 函数

2.5.20. ble_conn_chann_map_get

原型：ble_status_t ble_conn_chann_map_get(uint8_t conidx)

功能：获取已建立的连线使用的 channel map

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

执行后会有 BLE_CONN_EVT_CHANN_MAP_GET_RSP 消息通知 callback 函数

2.5.21. ble_conn_ping_to_get

原型：ble_status_t ble_conn_ping_to_get(uint8_t conidx)

功能：获取已建立的连线的 ping timeout 值

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

执行后会有 BLE_CONN_EVT_PING_TO_VAL_GET_RSP 消息通知 callback 函数

2.5.22. ble_conn_ping_to_set

原型：ble_status_t ble_conn_ping_to_set(uint8_t conidx, uint16_t tout)

功能：设置已建立的连线的 ping timeout 值

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

tout，ping timeout 值，以 10 ms 为单位

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

执行后会有 BLE_CONN_EVT_PING_TO_SET_RSP 消息通知 callback 函数

2.5.23. ble_conn_rssi_get

原型：ble_status_t ble_conn_rssi_get(uint8_t conidx)

功能：获取已建立的连线上最近收到的 packet 的 rssi

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

执行后会有 BLE_CONN_EVT_RSSI_GET_RSP 消息通知 callback 函数

2.5.24. ble_conn_param_update_req

原型：ble_status_t ble_conn_param_update_req (uint8_t conidx, uint16_t interval,

uint16_t latency, uint16_t supv_to, uint16_t ce_len)

功能：设置已建立的连线的连接参数

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

interval，需要设置的 connection event 周期，以 1.25 ms 为单位

latency，slave 可以不用听 master 包的最大 connection event 数

supv_to，断线超时，以 10 ms 为单位

ce_len，connection event 的长度，以 0.625 ms 为单位

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

执行后会有 BLE_CONN_EVT_PARAM_UPDATE_RSP 消息通知 callback 函数，在连线参数更新完成后还会有 BLE_CONN_EVT_PARAM_UPDATE_INFO 消息通知 callback 函数

2.5.25. ble_conn_per_adv_sync_trans

原型：ble_status_t ble_conn_per_adv_sync_trans(uint8_t conidx, uint8_t trans_idx,

uint16_t srv_data)

功能：将 periodic advertising 信息转发给已建立连线的对端设备，使其可以直接发起 sync

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

trans_idx，需要转发的 index，可以是 local 创建的 periodic advertising 的 index，也可以是 local sync 成功后的 sync index

srv_data，app 可以设置的 service data

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

执行后会有 BLE_CONN_EVT_PER_SYNC_TRANS_RSP 消息通知 callback 函数

2.5.26. ble_conn_name_get_cfm

原型：ble_status_t ble_conn_name_get_cfm(uint8_t conidx, uint16_t status, uint16_t token, uint16_t cmpl_len, uint8_t *p_name, uint16_t name_len)

功能：在 callback 中收到 BLE_CONN_EVT_NAME_GET_IND 消息后调用该函数回复对端发起的获取本地 name 的申请

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

status，confirm 状态，如果有错误或者异常就填入 error code，否则填 0

token，message token，在 BLE_CONN_EVT_NAME_GET_IND 消息中获取

cmpl_len，本地 name 的总长度

p_name，回复的 name 的全部或者部分内容指针

name_len，本次回复的 name 长度，若回复全部 name 则与 cmpl_len 相等

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.5.27. ble_conn_appearance_get_cfm

原型：ble_status_t ble_conn_appearance_get_cfm(uint8_t conidx, uint16_t status,

uint16_t token, uint16_t appearance)

功能：在 callback 中收到 BLE_CONN_EVT_APPEARANCE_GET_IND 消息后调用该函数回复对端发起的获取本地 appearance 的申请

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

status，confirm 状态，如果有错误或者异常就填入 error code，否则填 0

token，在 BLE_CONN_EVT_APPEARANCE_GET_IND 消息中获取

appearance，回复的本地 appearance

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.5.28. ble_conn_slave_prefer_param_get_cfm

原型：ble_status_t ble_conn_slave_prefer_param_get_cfm(uint8_t conidx,

uint16_t status, uint16_t token, ble_gap_prefer_periph_param_t *p_param)

功能：在 callback 中收到 BLE_CONN_EVT_SLAVE_PREFER_PARAM_GET_IND 消息后调用该函数回复对端 slave prefer parameter

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

status，confirm 状态，如果有错误或者异常就填入 error code，否则填 0

token，在 BLE_CONN_EVT_SLAVE_PREFER_PARAM_GET_IND 消息中获取

p_param，slave prefer parameter 结构体指针，包括 interval，latency 等

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.5.29. ble_conn_name_set_cfm

原型：ble_status_t ble_conn_name_set_cfm(uint8_t conidx, uint16_t status, uint16_t token)

功能：在 callback 中收到 BLE_CONN_EVT_NAME_SET_IND 消息后调用该函数回复对端发起的设置本地 name 的请求

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

status，confirm 状态，如果有错误或者异常就填入 error code，否则填 0

token，在 BLE_CONN_EVT_NAME_SET_IND 消息中获取

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.5.30. ble_conn_appearance_set_cfm

原型：ble_status_t ble_conn_appearance_set_cfm(uint8_t conidx, uint16_t status,
uint16_t token)

功能：在 callback 中收到 BLE_CONN_EVT_APPEARANCE_SET_IND 消息后调用该函数回复对端发起的设置本地 appearance 的请求

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

status，confirm 状态，如果有错误或者异常就填入 error code，否则填 0

token，在 BLE_CONN_EVT_APPEARANCE_SET_IND 消息中获取

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.5.31. ble_conn_param_update_cfm

原型：ble_status_t ble_conn_param_update_cfm(uint8_t conidx, bool accept,
uint16_t ce_len_min, uint16_t ce_len_max)

功能：在 callback 中收到 BLE_CONN_EVT_PARAM_UPDATE_IND 消息后调用该函数回复对端发起的 connection parameter update 的请求

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

accept，true 表示接受 connection 参数更新请求，否则回复 false

ce_len_min，connection event 的最小时间，以 0.625 ms 为单位

ce_len_max，connection event 的最大时间，以 0.625 ms 为单位

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.5.32. ble_conn_local_tx_pwr_get

原型：ble_status_t ble_conn_local_tx_pwr_get(uint8_t conidx,
ble_gap_phy_pwr_value_t phy)

功能：获取已建立连线对应 PHY 上本地 transmit 时使用的 power

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

phy，获取 power 对应的 PHY

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后有 BLE_CONN_EVT_LOC_TX_PWR_GET_RSP 消息通知 callback 函数

2.5.33. ble_conn_peer_tx_pwr_get

原型：ble_status_t ble_conn_peer_tx_pwr_get(uint8_t conidx,
ble_gap_phy_pwr_value_t phy)

功能：获取已建立连线对应 PHY 上对端 transmit 时使用的 power

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

phy，获取 power 对应的 PHY

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后有 BLE_CONN_EVT_PEER_TX_PWR_GET_RSP 消息通知 callback 函数

2.5.34. ble_conn_tx_pwr_report_ctrl

原型：ble_status_t ble_conn_tx_pwr_report_ctrl(uint8_t conidx, uint8_t local_en,

uint8_t remote_en)

功能：设置已建立的连线上本地或者对端transmit power 发生变化时是否发送通知给 APP

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

local_en，本地 transmit power 发生变化时是否通知 remote_en，

对端 transmit power 发生变化时是否通知

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_CONN_EVT_TX_PWR_RPT_CTRL_RSP 消息通知 callback 函数

如果成功设置了 local enable，在本地 transmit power 发生变化时会

有 BLE_CONN_EVT_LOC_TX_PWR_RPT_INFO 消息通知 callback 函数

如果成功设置了 remote enable，在对端 tx power 发生变化时会

有 BLE_CONN_EVT_PEER_TX_PWR_RPT_INFO 消息通知 callback 函数

2.5.35. ble_conn_path_loss_ctrl

原型：ble_status_t ble_conn_path_loss_ctrl(uint8_t conidx, uint8_t enable,

uint8_t high_threshold, uint8_thigh_hysteresis, int8_t low_threshold,

uint8_t low_hysteresis, uint16_t min_time)

功能：设置已建立的连线上的 path loss 通知

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

enable，是否通知 path loss

high_threshold，high zone 的 path loss 阈值

high_hysteresis，high threshold 的 hysteresis 值

low_threshold，low zone 的 path loss 阈值

low_hysteresis，low threshold 的 hysteresis 值

min_time，path 发生变化后需要停留的最小 connection event 数

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_CONN_EVT_PATH_LOSS_CTRL_RSP 消息通知 callback 函数

如果成功设置为 enable，在 path zone 发生变化时会

有 BLE_CONN_EVT_PATH_LOSS_THRESHOLD_INFO 通知 callback 函数

2.5.36. ble_conn_enable_central_feat

原型：ble_status_t ble_conn_enable_central_feat(uint8_t conidx)

功能：设备作为 peripheral 时，获取及配置 central 的 gap service 信息，当 ble_adp_cfg 中 cfg.att_cfg 设置 BLE_GAP_ATT_CLI_DIS_AUTO_FEAT_EN bit 位时，连接成功后需调用该接口，否则可不使用该接口

输入参数：conidx，BLE 连线 index，可在连接成功的消息中获得

输出参数：无

返回值：成功开始执行返回 0，失败返回 ble_status_t 中定义的 error code

2.6. BLE security API

头文件 ble_sec.h。

BLE security 模块主要提供 pairing，authentication，encryption 等过程中交互的接口。

2.6.1. security 消息类型

APP 可以向 BLE security 模块注册 callback 函数，BLE 协议栈会通过 callback 函数发送以下的 event message 给 APP。

- BLE_SEC_EVT_PAIRING_REQ_IND

该消息是在收到对端发起的 pairing request 后通知 APP，消息数据类型为 ble_gap_pairing_req_ind_t，包含对端 authentication request level 等信息。APP 可以调用 ble_sec_pairing_req_cfm 进行回复。

- BLE_SEC_EVT_LTK_REQ_IND

该消息是在 authentication 过程中向 APP 获取已配对设备的 long term key，消息数据类型为 ble_gap_ltk_req_ind_t，包含 LTK size 信息。APP 可以调用 ble_sec_ltk_req_cfm 进行回复。

- BLE_SEC_EVT_KEY_DISPLAY_REQ_IND

该消息是在配对过程中需要 PIN CODE 时向 APP 获取，消息数据类型为 ble_gap_tk_req_ind_t，包含 connection index 信息。APP 可以调用 ble_sec_key_display_enter_cfm 进行回复。

- BLE_SEC_EVT_KEY_ENTER_REQ_IND

该消息是在配对过程中需要用户输入 passkey 时通知 APP，消息数据类型为 ble_gap_tk_req_ind_t，包含 connection index 信息。APP 可以调用 ble_sec_key_display_enter_cfm 进行回复。

- BLE_SEC_EVT_KEY_OOB_REQ_IND

该消息是在配对过程中需要 APP 使用 OOB data 作为 temp key 时通知 APP，消息数据类型为 ble_gap_tk_req_ind_t，包含 connection index 信息。APP 可以调用 ble_sec_oob_req_cfm 进行回复。

- BLE_SEC_EVT_NUMERIC_COMPARISON_IND

该消息是在配对过程中需要用户对产生的 number 进行比对时通知 APP，消息数据类型为 ble_gap_nc_ind_t，包含需要比对的数字。APP 可以调用 ble_sec_nc_cfm 进行回复。

- BLE_SEC_EVT_IRK_REQ_IND

该消息是在配对过程中需要获取本地 IRK 进行分发时通知 APP，消息数据类型为 ble_gap_irk_req_ind_t，包含 connection index 信息。APP 可以调用 ble_sec_irk_req_cfm 函数进行回复。

- BLE_SEC_EVT_CSRK_REQ_IND

该消息是在配对过程中需要获取本地 CSRK 进行分发时通知 APP，消息数据类型为 ble_gap_csrk_req_ind_t，包含 connection index 信息。APP 可以调用 ble_sec_csrk_req_cfm 函数进行回复。

- BLE_SEC_EVT_OOB_DATA_REQ_IND

该消息是在配对过程中使用 OOB 方式时向 APP 获取 OOB data，消息数据类型为 ble_gap_oob_data_req_ind_t，包含 connection index 信息。APP 可以调用 ble_sec_oob_data_req_cfm 函数进行回复。

- BLE_SEC_EVT_PAIRING_SUCCESS_INFO

该消息是在配对成功后通知 APP，消息数据类型为 ble_sec_pairing_success_t，包含是否是 secure connection 以及 pairing level 等信息。

- BLE_SEC_EVT_PAIRING_FAIL_INFO

该消息是在配对失败时通知 APP，消息数据类型为 ble_sec_pairing_fail_t，包含 pairing 失败的原因等。

- BLE_SEC_EVT_SECURITY_REQ_INFO

该消息是在作为 master 时收到对端 slave 发起 security request 时通知 APP，消息数据类型为 ble_sec_security_req_info_t，包含对端的 authentication request level 等。APP 在收到该消息后可以根据是否有对端的 LTK 来决定发起 encryption 或者 pairing。

- BLE_SEC_EVT_ENCRYPT_REQ_IND

该消息是在收到对端发起的 encryption request 后通知 APP，消息数据类型为

ble_gap_encrypt_req_ind_t，包含 ediv 和 random number 等信息。APP 可以调用 ble_sec_encrypt_req_cfm 进行回复。

- BLE_SEC_EVT_ENCRYPT_INFO

该消息会在 encryption 完成后通知 APP。消息数据类型为 ble_sec_encrypt_info_t，包含是否 encryption 成功的 status，如果成功还会有 pairing level 等信息。

- BLE_SEC_EVT_OOB_DATA_GEN_INFO

该消息会在 APP 调用 ble_sec_oob_data_gen 后成功生成一组 OOB data 时通知 APP。消息数据类型为 ble_sec_oob_data_info_t，包含生成的 OOB data。

- BLE_SEC_EVT_KEY_PRESS_NOTIFY_RSP

该消息返回 APP 调用 ble_sec_key_press_notify 的结果，消息数据类型为 ble_gap_key_press_ntf_rsp_t，包含发送 key press notification 的 status。

- BLE_SEC_EVT_KEY_PRESS_INFO

该消息会在收到对端的 key press notification 后通知 APP，消息数据类型为 ble_gap_key_pressed_info_t，包含对端的 key press type 等信息。

2.6.2. ble_sec_init

原型：ble_status_t ble_sec_init(void)

功能：初始化 BLE security 模块

输入参数：无

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.6.3. ble_sec_callback_register

原型：ble_status_t ble_sec_callback_register(ble_sec_evt_handler_t callback)

功能：该接口用于注册 BLE security 模块的 event 消息处理函数

输入参数：callback，callback 处理函数，security 消息的说明见 ***security 消息类型***

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.6.4. ble_sec_callback_unregister

原型：ble_status_t ble_sec_callback_unregister(ble_sec_evt_handler_t callback)

功能：向 BLE security 模块取消注册的消息处理函数

输入参数：callback，需要取消的 callback 函数

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.6.5. ble_sec_security_req

原型：ble_status_t ble_sec_security_req(uint8_t conidx, uint8_t auth)

功能：作为 slave 时主动配对发送的 security request 消息

输入参数：conidx，connection index

auth，指示配对安全类型，参考枚举 ble_gap_auth_mask_t

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.6.6. ble_sec_bond_req

原型：ble_status_t ble_sec_bond_req(uint8_t conidx,
ble_gap_pairing_param_t *p_param, uint8_t sec_req_level)

功能：作为 master 时主动发起配对发送的 pairing request 消息，或者在收到 BLE_SEC_EVT_SECURITY_REQ_INFO 消息后响应对端 slave 的 security request 发起配对

输入参数：conidx，connection index

p_param，pairing request 消息的参数，参考结构体 ble_gap_pairing_param_t

sec_req_level，安全请求 level，参考枚举 ble_gap_sec_req_t

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.6.7. ble_sec_encrypt_req

原型：ble_status_t ble_sec_encrypt_req(uint8_t conidx, ble_gap_ltk_t *p_peer_ltk)

功能：存在对端的 LTK 时，发送加密请求

输入参数：conidx，connection index

p_peer_ltk，对端的 LTK

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.6.8. ble_sec_key_press_notify

原型：ble_status_t ble_sec_key_press_notify(uint8_t conidx, uint8_t type)

功能：发送 keypress notify 消息

输入参数：conidx，connection index

type，0：Passkey entry started

1：Passkey digit entered

2：Passkey digit erased

3：Passkey cleared

4：Passkey entry completed

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_SEC_EVT_KEY_PRESS_NOTIFY_RSP 消息通知 callback 函数

2.6.9. ble_sec_key_display_enter_cfm

原型：ble_status_t ble_sec_key_display_enter_cfm(uint8_t conidx, bool accept, uint32_t passkey)

功能：pairing 过程中在 callback 函数中收到 BLE_SEC_EVT_KEY_DISPLAY_REQ_IND 或者 BLE_SEC_EVT_KEY_ENTER_REQ_IND 后调用该函数回复 PIN CODE 或者 passkey

输入参数：conidx，connection index

accept，是否接收请求

passkey，值的范围在 000000-999999

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.6.10. ble_sec_oob_req_cfm

原型：ble_status_t ble_sec_oob_req_cfm(uint8_t conidx, bool accept, uint8_t *p_key)

功能：pairing 过程中在 callback 函数中收到 BLE_SEC_EVT_KEY_OOB_REQ_IND 消息后调用该函数回复 OOB TK

输入参数：conidx，connection index

accept，是否接收请求

p_key，128bit 的 key 值

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.6.11. ble_sec_nc_cfm

原型：ble_status_t ble_sec_nc_cfm(uint8_t conidx, bool accept)

功能：pairing 过程中在 callback 函数中收到 BLE_SEC_EVT_NUMERIC_COMPARISON_IND 消息后调用该函数回复 numeric comparison 的结果

输入参数：conidx，connection index

accept，numeric comparison 结果是否一致

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.6.12. ble_sec_ltk_req_cfm

原型：ble_status_t ble_sec_ltk_req_cfm(uint8_t conidx, uint8_t accept, ble_gap_ltk_t *p_ltk)

功能：在 callback 函数中收到 BLE_SEC_EVT_LTK_REQ_IND 消息后调用该函数回复本地的 LTK 信息或者拒绝该请求

输入参数：conidx，connection index

accept，是否接收请求

p_ltk，LTK 值的指针

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.6.13. ble_sec_irk_req_cfm

原型：ble_status_t ble_sec_irk_req_cfm(uint8_t conidx, uint8_t accept, ble_gap_irk_t *p_irk)

功能：在 callback 函数中收到 BLE_SEC_EVT_IRK_REQ_IND 消息后调用该函数回复本地的 IRK 信息或者拒绝该请求

输入参数：conidx，connection index

accept，是否接收请求

p_irk，IRK 值的指针

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.6.14. ble_sec_csrk_req_cfm

原型：ble_status_t ble_sec_csrk_req_cfm(uint8_t conidx, uint8_t accept,

ble_gap_csrk_t *p_csrk)

功能：在 callback 函数中收到 BLE_SEC_EVT_CSRK_REQ_IND 消息后调用该函数回复本地的 CSRK 信息或者拒绝该请求

输入参数：conidx，connection index

accept，是否接收请求

p_csrk，CSRK 值的指针

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.6.15. ble_sec_encrypt_req_cfm

原型：ble_status_t ble_sec_encrypt_req_cfm(uint8_t conidx, bool found, uint8_t *p_ltk,

uint8_t key_size)

功能：encryption 过程中在 callback 中收到 BLE_SEC_EVT_ENCRYPT_REQ_IND 消息后调用该函数回复本地的 LTK 信息或者拒绝该请求

输入参数：conidx，connection index

found，是否存在 key

p_ltk，local LTK 值的指针

key_size，key 的 size

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.6.16. ble_sec_pairing_req_cfm

原型：ble_status_t ble_sec_pairing_req_cfm(uint8_t conidx, uint8_t accept,

ble_gap_pairing_param_t *p_param,uint8_t sec_req_lvl)

功能：在 callback 函数中收到 BLE_SEC_EVT_PAIRING_REQ_IND 消息后调用该函数回复

pairing response 给对端设置或者拒绝该请求

输入参数：conidx，connection index

accept，是否接收请求

p_param，pairing response 消息的参数，参考结构体 ble_gap_pairing_param_t

sec_req_level，安全请求 level，参考枚举 ble_gap_sec_req_t

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.6.17. ble_sec_oob_data_req_cfm

原型：ble_status_t ble_sec_oob_data_req_cfm(uint8_t conidx, uint8_t accept,

uint8_t *p_conf, uint8_t *p_rand)

功能：pairing 过程中在 callback 函数中收到 BLE_SEC_EVT_OOB_DATA_REQ_IND 消息后调用该函数回复本地的 OOB data 信息或者拒绝该请求

输入参数：conidx，connection index

accept，是否接收请求

p_conf，peer confirm 值

p_rand，peer random 值

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.6.18. ble_sec_oob_data_gen

原型：ble_status_t ble_sec_oob_data_gen(void)

功能：调用该接口生成一组 OOB data

输入参数：无

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

成功生成 OOB data 后会有 BLE_SEC_EVT_OOB_DATA_GEN_INFO 消息

通知 callback 函数

2.7. BLE list API

头文件 ble_list.h。

BLE list 模块主要提供对 FAL，RAL，PAL 进行操作的接口，包括添加 device 到 list，删除 list 中的 device，清除 list 等。

2.7.1. list 消息类型

- BLE_LIST_EVT_OP_RSP

该消息返回 APP 调用 ble_fal_op, ble_fal_list_set, ble_fal_list_clear, ble_ral_op, ble_ral_list_set, ble_ral_list_clear, ble_pal_op, ble_pal_list_set, ble_pal_list_clear 函数操作 list 的结果，消息数据类型为 ble_list_data_t，包含 list type，op type 等内容，可以通过对数据中的 type 进行判断知道是针对哪个 list 操作的回复。

- BLE_LIST_EVT_LOC_RPA_GET_RSP

该消息返回 APP 调用 ble_loc_rpa_get 获取 local resolvable address 的结果，消息数据类型为 ble_list_data_t，其中 list type 为 BLE_RAL_TYPE，op type 为 GET_LOC_RPA。

- BLE_LIST_EVT_PEER_RPA_GET_RSP

该消息返回 APP 调用 ble_peer_rpa_get 获取 peer resolvable address 的结果，消息数据类型为 ble_list_data_t，其中 list type 为 BLE_RAL_TYPE，op type 为 GET_PEER_RPA。

2.7.2. ble_list_init

原型：ble_status_t ble_list_init(void)

功能：初始化 BLE list 模块

输入参数：无

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.7.3. ble_list_callback_register

原型：ble_status_t ble_list_callback_register(ble_list_evt_handler_t callback)

功能：注册处理 BLE list 消息的 callback 函数

输入参数：callback，处理 BLE list 消息的函数，list 消息的说明见 ***list 消息类型***

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.7.4. ble_list_callback_unregister

原型：ble_status_t ble_list_callback_unregister(ble_list_evt_handler_t callback)

功能：向 BLE list 模块取消注册的 callback 函数

输入参数：callback，需要取消的 callback 函数

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.7.5. ble_fal_op

原型：ble_status_t ble_fal_op(ble_gap_addr_t *p_addr_info, bool add)

功能：将指定 device 加入或移出 filter accept list

输入参数：p_addr_info，device address 指针

add，true 表示加入 FAL，false 表示移出 FAL

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_LIST_EVT_OP_RSP 消息通知 callback 函数，list type

为 BLE_FAL_TYPE，op type 为 RMV_DEVICE_FROM_LIST

或者 ADD_DEVICE_TO_LIST

2.7.6. ble_fal_list_set

原型：ble_status_t ble_fal_list_set(uint8_t num, ble_gap_addr_t *p_addr_info)

功能：设置 filter accept list，该操作会将 FAL 全部更新为指定内容

输入参数：num，需要设置到 FAL 中的 device 个数

p_addr_info，device 数组，数组中包含 num 个 address 信息

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_LIST_EVT_OP_RSP 消息通知 callback 函数，list type

为 BLE_FAL_TYPE，op type 为 SET_DEVICES_TO_LIST

2.7.7. ble_fal_clear

原型：ble_status_t ble_fal_clear(void)

功能：清空 filter accept list

输入参数：无

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_LIST_EVT_OP_RSP 消息通知 callback 函数，

list type 为 BLE_FAL_TYPE，op type 为 CLEAR_DEVICE_LIST

2.7.8. ble_fal_size_get

原型：uint8_t ble_fal_size_get(void)

功能：获取 filter accept list 最大元素个数

输入参数：无

输出参数：无

返回值：filter accept list 最大元素个数

2.7.9. ble_ral_op

原型：ble_status_t ble_ral_op(ble_gap_ral_info_t *p_ral_info, bool add)

功能：将指定设备加入或移出 resolving list

输入参数：p_ral_info，RAL 结构体指针，包括 identity address，IRK 等

add，true 表示加入 RAL，false 表示移出 RAL

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_LIST_EVT_OP_RSP 消息通知 callback 函数，list type 为 BLE_RAL_TYPE，op type 为 RMV_DEVICE_FROM_LIST 或者 ADD_DEVICE_TO_LIST

2.7.10. ble_ral_list_set

原型：ble_status_t ble_ral_list_set(uint8_t num, ble_gap_ral_info_t *p_ral_info)

功能：设置 resolving list，该操作会将 RAL 全部更新为指定内容

输入参数：num，需要设置到 RAL 中的 device 个数

p_ral_info，RAL 结构体数组，数组中包含 num 个 RAL 结构体

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_LIST_EVT_OP_RSP 消息通知 callback 函数，list type 为 BLE_RAL_TYPE，op type 为 SET_DEVICES_TO_LIST

2.7.11. ble_ral_clear

原型：ble_status_t ble_ral_clear(void)

功能：清空 resolving list

输入参数：无

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_LIST_EVT_OP_RSP 消息通知 callback 函数，list type 为 BLE_RAL_TYPE，op type 为 CLEAR_DEVICE_LIST

2.7.12. ble_ral_size_get

原型：uint8_t ble_ral_size_get(void)

功能：获取 resolving list 最大元素个数

输入参数：无

输出参数：无

返回值：resolving list 最大元素个数

2.7.13. ble_loc_rpa_get

原型：ble_status_t ble_loc_rpa_get(uint8_t *p_peer_id, uint8_t peer_id_type)

功能：获取当前对指定 device 使用的本地 resolvable private address

输入参数：p_peer_id，指定 device 的 identity address

peer_id_type，指定 device 的 identity address type

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_LIST_EVT_LOC_RPA_GET_RSP 消息通知 callback 函数

2.7.14. ble_peer_rpa_get

原型：ble_status_t ble_peer_rpa_get(uint8_t *p_peer_id, uint8_t peer_id_type)

功能：获取指定 device 当前使用的 resolvable private address

输入参数：p_peer_id，指定 device 的 identity address

peer_id_type，指定 device 的 identity address type

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_LIST_EVT_PEER_RPA_GET_RSP 消息通知 callback 函数

2.7.15. ble_pal_op

原型：ble_status_t ble_pal_op(ble_gap_pal_info_t *p_pal_info, bool add)

功能：将指定 device 加入或移出 periodic advertising list

输入参数：p_pal_info，PAL 结构体指针，包括 address，SID 等

add，true 表示加入 PAL，false 表示移出 PAL

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_LIST_EVT_OP_RSP 消息通知 callback 函数，list type 为 BLE_PAL_TYPE，op type 为 RMV_DEVICE_FROM_LIST 或者 ADD_DEVICE_TO_LIST

2.7.16. ble_pal_list_set

原型：ble_status_t ble_pal_list_set(uint8_t num, ble_gap_pal_info_t *p_pal_info)

功能：设置 periodic advertising list，该操作会将 PAL 全部更新为指定内容

输入参数：num，需要设置到 PAL 中的 device 个数

p_ral_info，PAL 结构体数组，数组中包含 num 个 PAL 结构体

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_LIST_EVT_OP_RSP 消息通知 callback 函数，list type 为 BLE_PAL_TYPE，op type 为 SET_DEVICES_TO_LIST

2.7.17. ble_pal_clear

原型：ble_status_t ble_pal_clear(void)

功能：清空 periodic advertising list

输入参数：无

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_LIST_EVT_OP_RSP 消息通知 callback 函数，list type 为 BLE_PAL_TYPE，op type 为 CLEAR_DEVICE_LIST

2.7.18. ble_pal_size_get

原型：uint8_t ble_pal_size_get(void)

功能：获取 periodic advertising list 最大元素个数

输入参数：无

输出参数：无

返回值：periodic advertising list 最大元素个数

2.8. BLE periodic sync API

头文件 ble_per_sync.h。

BLE periodic sync 模块主要提供 sync periodic advertising，上报接收到的 periodic advertising data 等接口。

2.8.1. periodic sync 消息类型

APP 可以向 BLE periodic sync 模块注册 callback 函数，BLE 协议栈会通过 callback 函数发送以下的 event message 给 APP。

- BLE_PER_SYNC_EVT_STATE_CHG

该消息会在 periodic sync 状态发生变化时通知 callback 函数，消息数据类型为 ble_per_sync_state_chg_t，包含新的状态及发生变化的原因。

- BLE_PER_SYNC_EVT_REPORT

该消息会在收到 periodic advertising report 后通知 callback 函数，消息数据类型为 ble_gap_adv_report_info_t，包含发送 periodic advertising 的设备地址，发送的 PHY，advertising data 等内容。

- BLE_PER_SYNC_EVT_ESTABLISHED

该消息会在sync上periodic advertising后通知callback函数，消息数据类型为ble_per_sync_established_t，包含sync上的periodic advertising的PHY，interval，SID等内容。

- BLE_PER_SYNC_EVT_RPT_CTRL_RSP

该消息是对APP调用ble_per_sync_report_ctrl设置report内容的回复，消息数据类型为ble_per_sync_rpt_ctrl_rsp_t，包含设置的status。

2.8.2. ble_per_sync_init

原型：ble_status_t ble_per_sync_init(void)

功能：初始化BLE periodic sync模块

输入参数：无

输出参数：无

返回值：成功返回0，失败返回ble_status_t中定义的error code

2.8.3. ble_per_sync_callback_register

原型：ble_status_t ble_per_sync_callback_register(ble_per_sync_evt_handler_t callback)

功能：注册处理periodic sync消息的callback函数，per sync消息的说明

见 ***periodic sync 消息类型***

输入参数：callback，处理periodic sync消息的callback函数

输出参数：无

返回值：成功返回0，失败返回ble_status_t中定义的error code

2.8.4. ble_per_sync_callback_unregister

原型：ble_status_t ble_per_sync_callback_unregister(ble_per_sync_evt_handler_t callback)

功能：向BLE periodic sync模块取消注册的callback函数

输入参数：callback，需要取消的callback函数

输出参数：无

返回值：成功返回0，失败返回ble_status_t中定义的error code

2.8.5. ble_per_sync_start

原型：ble_status_t ble_per_sync_start (ble_gap_local_addr_type_t own_addr_type,
ble_gap_per_sync_param_t *p_param)

功能：开始 periodic sync

输入参数：own_addr_type，sync 过程中使用的 local address type
p_param，periodic sync 参数结构体指针

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

执行后会有 BLE_PER_SYNC_EVT_STATE_CHG 消息通知 callback 函数，如果成功 sync 上 periodic advertising，还会有 BLE_PER_SYNC_EVT_ESTABLISHED 通知 callback 函数以及 BLE_PER_SYNC_EVT_REPORT 通知接收到的数据

2.8.6. ble_per_sync_cancel

原型：ble_status_t ble_per_sync_cancel (void)

功能：取消正在进行的 periodic sync 流程

输入参数：无

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

执行后会有 BLE_PER_SYNC_EVT_STATE_CHG 消息通知 callback 函数

2.8.7. ble_per_sync_terminate

原型：ble_status_t ble_per_sync_terminate (uint8_t sync_idx)

功能：中止已经同步成功的 periodic sync train

输入参数：sync_idx，sync index

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

执行后会有 BLE_PER_SYNC_EVT_STATE_CHG 消息通知 callback 函数

2.8.8. ble_per_sync_ctrl

原型：ble_status_t ble_per_sync_ctrl(uint8_t sync_idx, uint8_t ctrl)

功能：修改同步成功后上报通知的内容

输入参数：sync_idx，sync index

ctrl，periodic sync report 控制位，由 ble_per_sync_rpt_ctrl_bit_t 中的 bit 组合

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

设置后会有 BLE_PER_SYNC_EVT_RPT_CTRL_RSP 通知 callback 函数

2.9. BLE storage API

头文件 ble_storage.h，该模块使用 flash 来存储并管理 peer 的 bond 信息，包括 peer_irk, peer_ltk, peer_csrk, local_irk, local_ltk 和 local_csrk 等。

头文件中宏定义 BLE_PEER_NUM_MAX 用来定义最大 peer 的个数，当存储的 peer 个数已达到上限，还需要存储新的 peer 信息，会使用 LRU 算法来删除最久未被使用的 peer 信息。

2.9.1. ble_storage_init

原型：ble_status_t ble_storage_init(void)

功能：初始化 storage 模块，从 flash 中获取所有 peer 的信息，只需在初始化流程中调用一次

输入参数：无

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.9.2. ble_peer_data_bond_store

原型：ble_status_t ble_peer_data_bond_store(ble_gap_addr_t *addr,

ble_gap_sec_bond_data_t *bond_data)

功能：该函数用于存储 peer 的 bond 信息，该信息也会被保存至 flash 中，若之前存在相同索引的 bond 信息，将会进行更新保存。如果 BLE adapter config 时 keys_user_mgr 为 false，则 BLE security 会自动进行 bond 信息的存储，APP 无需进行相关操作。

输入参数：addr，连接设备的地址，若 bond_data 中不包含 identity addr，该地址做为索引进行存储，若 bond_data 中包含 identity addr，identity addr 将作为索引进行存储，该地址则无作用，但不能为空

bond_data，需要存储的 bond 信息

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.9.3. ble_peer_data_bond_load

原型：ble_status_t ble_peer_data_bond_load(ble_gap_addr_t *addr,
ble_gap_sec_bond_data_t *bond_data)

功能：该函数用于获取 bond 信息

输入参数：addr，将以该地址作为索引进行获取，可为 identity addr 或 RPA

输出参数：bond_data，获取到的 bond 信息

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.9.4. ble_peer_data_delete

原型：ble_status_t ble_peer_data_delete(ble_gap_addr_t *addr)

功能：该函数用于删除指定 addr 对应的 peer 信息，flash 中的内容也会被删除

输入参数：addr，将以该地址作为索引进行删除 peer 信息，可为 identity addr 或者 RPA

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.9.5. ble_peer_all_addr_get

原型：ble_status_t ble_peer_all_addr_get(uint8_t *num, ble_gap_addr_t *id_addrs)

功能：该函数用于获取 storage 模块下所有的 peer 设备的 identity addr

输入参数：num，num 指针的值表示需要获取 peer 设备的最大个数，不能超过
BLE_PEER_NUM_MAX，并且决定 id_addrs 指针指向的内存大小为
num*sizeof(ble_gap_addr_t)

输出参数：num， num 中的值为获取到的实际个数
id_addrs， id_addrs 指针中保存实际获取到的 peer identity addr

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.9.6. ble_svc_data_save

原型：ble_status_t ble_svc_data_save(uint8_t conn_idx, uint16_t data_id ，uint32_t len,
uint8_t *p_data)

功能：该函数提供给上层 BLE service，将对应连接设备该 service 的相关数据存储至 flash

输入参数：conn_idx，BLE 连线 index，可在连接成功的消息中获得

data_id，用于标识 service 信息，由上层 APP 定义

p_len，存储 service 信息 p_data 的长度

p_data，存储 service 信息指针

输出参数：无

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.9.7. ble_svc_data_load

原型：ble_status_t ble_svc_data_load(uint8_t conn_idx, uint16_t data_id, void **pp_data, uint32_t *p_len)

功能：该函数提供给上层 BLE service 用于获取对应连接设备在 flash 中存储的该 service 的相关数据

输入参数：conn_idx，BLE 连线 index，可在连接成功的消息中获得

data_id，用于标识 service 信息，由上层 APP 定义

输出参数：pp_data，获取到的 service 信息

p_len，获取到 service 信息 pp_data 的长度

返回值：成功执行返回 0，失败返回 ble_status_t 中定义的 error code

2.10. BLE gatts API

头文件 ble_gatts.h

BLE GATT server 模块主要提供注册/删除 GATT service，向 client 发送 notification/indication 等接口。

2.10.1. gatts 消息类型

BLE services 可以向 BLE GATT server 模块注册 callback 函数，BLE GATT server 模块会通过 callback 函数发送以下的 event message 给 BLE services。

- BLE_SRV_EVT_SVC_ADD_RSP

该消息返回调用 ble_gatts_svc_add 函数向 GATT server 模块添加 service 的结果，消息数据类型为 ble_gatts_svc_add_rsp_t，包含 add service 的 status，如果 status 为 0，还包含分配的 service ID 及该 service 在 database 中的 start handle 值。

- BLE_SRV_EVT_SVC_RMV_RSP

该消息返回调用 ble_gatts_svc_rmv 函数向 GATT server 模块删除 service 的结果，消息数据类型为 ble_gatts_svc_rmv_rsp_t，包含 remove service 的 status 及 service ID。

- BLE_SRV_EVT_CONN_STATE_CHANGE_IND

该消息会在设备连接状态发生变化时通知 callback 函数，消息数据类型为 ble_gatts_conn_state_change_ind_t，包含连接状态。如果是 connected 状态，还会包含连线的 connection index 及对端设备的 address 信息；如果是 disconnected 状态，还会包括断线的原因等。

- BLE_SRV_EVT_GATT_OPERATION

该消息会在与对端 GATT client 发生交互时通知 callback 函数，消息数据类型为 ble_gatts_op_info_t，包含 subevent，发生交互的连线的 connection index 以及不同的 subevent 对应的消息数据。该消息的 subevent 包含以下几种：

- BLE_SRV_EVT_READ_REQ

  该 subevent 会在对端 client 发起 attribute read 请求时通知到 callback 函数，对应的 subevent 数据类型为 ble_gatts_read_req_t，包含需要读取的 attribute index，attribute value 的 offset 及最大长度等。同时该消息中还包括 pending_cfm flag，可以由上层决定是否在 callback 函数处理完成后直接由 GATT server 模块回复 read 的结果给对端 client。如果需要，可以将数据拷贝到 server 模块预先分配好的地址（分配的大小为最大长度）；否则就将 pending_cfm 置为 true，然后根据需求调用 ble_gatts_svc_attr_read_cfm 进行回复。

- BLE_SRV_EVT_WRITE_REQ

  该 subevent 会在对端 client 发起 attribute write 请求时通知到 callback 函数，对应的 subevent 数据类型为 ble_gatts_write_req_t，包含需要写的 attribute index，写入的数据对应的 offset，长度及内容等。同时该消息中还包括 pending_cfm flag，可以由上层决定是否在 callback 函数处理完成后直接由 GATT server 模块回复 write 的结果。如果不需要，可以将 pending_cfm 置为 true，然后根据需求调用 ble_gatts_svc_attr_write_cfm 进行回复。

- BLE_SRV_EVT_NTF_IND_SEND_RSP

  该 subevent 返回调用 ble_gatts_ntf_ind_send 或 ble_gatts_ntf_ind_send_by_handle 发送 GATT notification 或 indication 的结果，subevent 数据类型为 ble_gatts_ntf_ind_send_rsp_t，包含发送数据的 status，对应的 service id 和 attribute index。

- BLE_SRV_EVT_NTF_IND_MTP_SEND_RSP

  该 subevent 返回调用 ble_gatts_ntf_ind_mtp_send 向多个远端设备发送 notification 或者 indication 的结果，消息数据类型为 ble_gatts_ntf_ind_mtp_send_rsp_t，包含发送数据的 status，对应的 service id 和 attribute index。

2.10.2. ble_gatts_init

原型：ble_status_t ble_gatts_init(void)

功能：BLE GATT server 模块初始化

输入参数：无

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.10.3. ble_gatts_svc_add

原型：ble_status_t ble_gatts_svc_add(uint8_t *p_svc_id, const uint8_t *uuid, uint16_t start_hdl, uint8_t info, const void *p_table, uint16_t table_length, p_fun_srv_cb srv_cb)

功能：向 GATT server 模块添加 service

输入参数：uuid，service UUID 地址

start_hdl，service 起始 attribute handle 值，0 表示不指定，由模块自动分配

info，service information, 详见 ble_gatt_svc_info_bf

p_table，service 所有 attribute 数组，每个 attribute 结构都是 ble_gatt_attr_desc_t

table_length，service attribute 数组长度

srv_cb，GATT server 消息处理函数，消息类型见 ***gatts 消息类型***

输出参数：p_svc_id，BLE GATT server 模块为该 service 分配的 ID

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_SRV_EVT_SVC_ADD_RSP 通知 callback 函数

2.10.4. ble_gatts_svc_rmv

原型：ble_status_t ble_gatts_svc_rmv(uint8_t svc_id)

功能：删除服务

输入参数：svc_id，调用 ble_gatts_svc_add 时为 service 分配的 ID 值

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_SRV_EVT_SVC_RMV_RSP 通知 callback 函数

2.10.5. ble_gatts_ntf_ind_send

原型：ble_status_t ble_gatts_ntf_ind_send(uint8_t conn_idx, uint8_t svc_id,

uint16_t att_idx, uint8_t *p_val, uint16_t len, ble_gatt_evt_type_t evt_type)

功能：发送 notification/indication

输入参数：conn_idx，连线的 connection index

svc_id，调用 ble_gatts_svc_add 时为 service 分配的 ID 值

att_idx，attribute 在 ble_gatts_svc_add 时的数组中的 index 值

p_val，需要发送的数据地址

len，需要发送的数据长度

evt_type，此次发送的类型是 notification 还是 indication

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_SRV_EVT_GATT_OPERATION 消息，subevent 为

BLE_SRV_EVT_NTF_IND_SEND_RSP 通知 callback 函数

2.10.6. ble_gatts_ntf_ind_send_by_handle

原型：ble_status_t ble_gatts_ntf_ind_send_by_handle(uint8_t conn_idx,

uint16_t handle, uint8_t *p_val, uint16_t len, ble_gatt_evt_type_t evt_type)

功能：通过 attribute handle 发送 notification/indication

输入参数：conn_idx，连线的 connection index

handle，attribute 的 handle 值，可由 ble_gatts_svc_add 时 attribute 在数组的 index 和该 service 的 start handle 得到

p_val，需要发送的数据地址

len，需要发送的数据长度

evt_type，此次发送的类型是 notification 还是 indication

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_SRV_EVT_GATT_OPERATION 消息，subevent 为

BLE_SRV_EVT_NTF_IND_SEND_RSP 通知 callback 函数

2.10.7. ble_gatts_ntf_ind_mtp_send

原型：ble_status_t ble_gatts_ntf_ind_mtp_send(uint32_t conidx_bf, uint8_t svc_id,

uint16_t att_idx, uint8_t *p_val, uint16_t len, ble_gatt_evt_type_t evt_type)

功能：向多个连接发送 notification/indication

输入参数：conidx_bf，connection index bit 组合，bit 0 表示 connection index 0x00,

bit 1 表示 connection index 0x01，依次类推

svc_id，调用 ble_gatts_svc_add 时为 service 分配的 ID 值

att_idx，attribute 在 ble_gatts_svc_add 时的数组中的 index 值

p_val，需要发送的数据地址

len，需要发送的数据长度

evt_type，此次发送的类型是 notification 还是 indication

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

完成后会有 BLE_SRV_EVT_GATT_OPERATION 消息，subevent 为

BLE_SRV_EVT_NTF_IND_MTP_SEND_RSP 通知 callback 函数

2.10.8. ble_gatts_mtu_get

原型：ble_status_t ble_gatts_mtu_get(uint8_t conidx, uint16_t *p_mtu)

功能：获取对应连线的 GATT MTU

输入参数：conn_idx，连线的 connection index

输出参数：p_mtu，获取到的连线 GATT MTU

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.10.9. ble_gatts_svc_attr_write_cfm

原型：ble_status_t ble_gatts_svc_attr_write_cfm(uint8_t conn_idx, uint16_t token,

uint16_t status)

功能：在 callback 函数中收到 BLE_SRV_EVT_GATT_OPERATION 消息且 subevent 为

BLE_SRV_EVT_WRITE_REQ 时，如果不需要 GATT server 模块自动回复可以将消息

数据中的 pending_cfm 置为 true，然后再根据需求调用该接口对 write 请求进行回复

输入参数：conn_idx，连线的 connection index

token，GATT token，在 BLE_SRV_EVT_WRITE_REQ 消息中获取

status，对 write 请求回复的状态

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.10.10. ble_gatts_svc_attr_read_cfm

原型：ble_status_t ble_gatts_svc_attr_read_cfm(uint8_t conn_idx, uint16_t token,
uint16_t status, uint16_t total_len, uint16_t value_len, uint8_t *p_value)

功能：在 callback 函数中收到 BLE_SRV_EVT_GATT_OPERATION 消息且 subevent 为 BLE_SRV_EVT_READ_REQ 时，如果不需要 GATT server 模块自动回复可以将消息数据中的 pending_cfm 置为 true，然后再根据需求调用该接口对 read 请求进行回复

输入参数：conn_idx，连线的 connection index

token，GATT token，在 BLE_SRV_EVT_READ_REQ 消息中获取

status，对 read 请求回复的状态

total_len，需要读取的 attribute 的总长度

value_len，对本次读取请求回复的 attribute 数据长度

p_value，对本次读取请求回复的 attribute 数据内容

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.10.11. ble_gatts_get_start_hdl

原型：ble_status_t ble_gatts_get_start_hdl(uint8_t svc_id, uint16_t *p_handle)

功能：获取 GATT server 模块为 service 分配的 start handle 值

输入参数：svc_id，service id，在 ble_gatts_svc_add 中获取

输出参数：p_handle，获取到的 start handle 值

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.11. BLE gattc API

头文件 ble_gattc.h。

BLE GATT client 模块主要提供发起 GATT discovery，读写对端 GATT server 中的 attribute 等接口。

2.11.1. gattc 消息类型

BLE services 可以向 BLE GATT client 模块注册 callback，BLE GATT client 模块会通过 callback 函数将以下的 event message 发送给 BLE services。

■ BLE_CLI_EVT_CONN_STATE_CHANGE_IND

该消息会在设备连接状态发生变化时通知 callback 函数，消息数据类型为 ble_gattc_conn_state_change_ind_t，包括连接状态 conn_state。如果是 connected 状态，还会包含连线的 connection index 及对端设备的 address 信息；如果是 disconnected 状态，还会包括断线的原因等。

■ BLE_CLI_EVT_GATT_OPERATION

该消息会在与对端 GATT server 发生交互时通知到 callback 函数，消息数据类型为 ble_gattc_op_info_t，包括 GATT client 操作的子类型 gattc_op_sub_evt，连接索引 conn_idx 以及不同的 subevent 对应的消息数据。该消息的 subevent 包含以下几种：

- BLE_CLI_EVT_SVC_DISC_DONE_RSP

  该 subevent 会在调用 ble_gattc_start_discovery 查询对端 GATT service 后返回是否找到注册的 UUID 对应的 service，对应的 subevent 数据类型为 ble_gattc_svc_dis_done_t，包含是否找到该 service 以及对应的 instance 数量等。

- BLE_CLI_EVT_READ_RSP

  该消息返回调用 ble_gattc_read 读取对端 GATT server attribute 数据的结果，对应的 subevent 数据类型为 ble_gattc_read_rsp_t，包含 service uuid，characteristic uuid 等。

- BLE_CLI_EVT_WRITE_RSP

  该消息返回调用 ble_gattc_write_req，ble_gattc_write_cmd 或 ble_gattc_write_signed 向对端 GATT server 写数据的结果，对应的 subevent 数据类型为 ble_gattc_write_rsp_t，包含 service uuid，characteristic uuid 等。

- BLE_CLI_EVT_NTF_IND_RCV

  该消息会在对端 GATT server 发送 notification 或 indication 时通知 callback 函数，对应的 subevent 消息数据类型为 ble_gattc_ntf_ind_t，包含 service uuid，characteristic uuid，attribute handle 等。

2.11.2. ble_gattc_init

原型：ble_status_t ble_gattc_init(void)

功能：初始化 GATT client 模块

输入参数：无

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.11.3. ble_gattc_start_discovery

原型：ble_status_t ble_gattc_start_discovery(uint8_t conn_idx,
p_discovery_done_cb callback)

功能：开始查询对端 GATT server 中的服务

输入参数：conn_idx，连线的 connection index

callback，查询完成回调函数

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.11.4. ble_gattc_svc_reg

原型：ble_status_t ble_gattc_svc_reg(ble_uuid_t *p_svc_uuid, p_fun_cli_cb p_cb)

功能：向 BLE GATT client 模块注册 callback 函数及对应的 service UUID

输入参数：p_svc_uuid，需要关注的 service uuid

p_cb，GATT client 消息处理函数，消息类型见 *gattc 消息类型*

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.11.5. ble_gattc_svc_unreg

原型：ble_status_t ble_gattc_svc_unreg(ble_uuid_t *p_svc_uuid)

功能：向 BLE GATT client 模块取消注册的 callback 函数及对应的 service UUID

输入参数：p_svc_uuid，需要取消关注的 service uuid

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.11.6. ble_gattc_read

原型：ble_status_t ble_gattc_read(uint8_t conidx, uint16_t hdl, uint16_t offset, uint16_t length)

功能：读取对端 GATT attribute 数据

输入参数：conidx，连线的 connection index

hdl，attribute handle

offset，读取数据偏移

length，读取数据长度

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code
完成后会有 BLE_CLI_EVT_GATT_OPERATION 消息，subevent 为
BLE_CLI_EVT_READ_RSP 通知 callback 函数

2.11.7. ble_gattc_write_req

原型：ble_status_t ble_gattc_write_req(uint8_t conidx, uint16_t hdl, uint16_t length,

uint8_t *p_value)

功能：写对端数据（write request）

输入参数：conidx，连线的 connection index

hdl，attribute handle

length，写数据长度

p_value，指向要写的数据

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code
完成后会有 BLE_CLI_EVT_GATT_OPERATION 消息，subevent 为
BLE_CLI_EVT_ WRITE_RSP 通知 callback 函数

2.11.8. ble_gattc_write_cmd

原型：ble_status_t ble_gattc_write_cmd(uint8_t conidx, uint16_t hdl, uint16_t length,

uint8_t *p_value)

功能：写对端数据（write command）

输入参数：conidx，连线的 connection index

hdl，attribute handle

length，写数据长度

p_value，指向要写的数据

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code
完成后会有 BLE_CLI_EVT_GATT_OPERATION 消息，subevent 为
BLE_CLI_EVT_ WRITE_RSP 通知 callback 函数

2.11.9. ble_gattc_write_signed

原型：ble_status_t ble_gattc_write_cmd(uint8_t conidx, uint16_t hdl, uint16_t length,

uint8_t *p_value)

功能：写对端数据（write signed）

输入参数：conidx，连线的 connection index

hdl，attribute handle

length，写数据长度

p_value，指向要写的数据

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code
完成后会有 BLE_CLI_EVT_GATT_OPERATION 消息，subevent 为
BLE_CLI_EVT_ WRITE_RSP 通知 callback 函数

2.11.10. ble_gattc_mtu_update

原型：ble_status_t ble_gattc_mtu_update(uint8_t conidx)

功能：更新 GATT mtu

输入参数：conidx，连线的 connection index

输出参数：无

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.11.11. ble_gattc_mtu_get

原型：ble_status_t ble_status_t ble_gattc_mtu_get(uint8_t conidx, uint16_t *p_mtu)

功能：获取对应连线上的 GATT mtu 值

输入参数：conidx，连线的 connection index

输出参数：p_mtu，mtu size

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.11.12. ble_gattc_find_char_handle

原型：ble_status_t ble_gattc_find_char_handle(uint8_t conn_idx, ble_gattc_uuid_info_t
*svc_uuid, ble_gattc_uuid_info_t *char_uuid, uint16_t *handle)

功能：找到对应 characteristic 的 value handle 值

输入参数：conidx，连线的 connection index

svc_uuid，指向 service uuid

char_uuid，指向 characteristic uuid

输出参数：handle，对应的 attribute handle 值

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.11.13. ble_gattc_find_desc_handle

原型：ble_status_t ble_gattc_find_desc_handle(uint8_t conn_idx, ble_gattc_uuid_info_t
*svc_uuid, ble_gattc_uuid_info_t *char_uuid,
ble_gattc_uuid_info_t *desc_uuid, uint16_t *handle)

功能：找到对应 description 的 handle 值

输入参数：conidx，连线的 connection index

svc_uuid，指向 service uuid

char_uuid，指向 characteristic uuid

desc_uuid，指向 description uuid

handle，对应的 attribute handle 值

返回值：成功返回 0，失败返回 ble_status_t 中定义的 error code

2.12. BLE export API

头文件 ble_export.h。

该文件包含 BLE 协议栈初始化，BLE 协议栈 task 和 BLE APP task 的初始化。

2.12.1. ble_stack_init

原型：void ble_stack_init(ble_cfg_func_t en_cfg, ble_os_api_t *p_os_api)

功能：BLE 协议栈初始化

输入参数：en_cfg，需要使能的配置，基于 bit 位实现，参考 ble_cfg_func_t

p_os_api，OS 相关 api 函数

输出参数：无

返回值：无

2.12.2. ble_stack_task_resume

原型：void ble_stack_task_resume(bool isr)

功能：BLE 任务恢复。BLE 任务无事情处理时会进入休眠，可调用该函数唤醒 BLE 任务

输入参数：isr，是否为中断调用

输出参数：无

返回值：无

2.12.3. ble_stack_task_init

原型：uint32_t ble_stack_task_init(uint32_t stack_size, uint32_t priority)

功能：BLE 任务的初始化

输入参数：stack_size，任务栈大小，单位为 4 个字节

priority，任务的优先级

输出参数：无

返回值：成功返回 0，失败返回其他值

2.12.4. ble_stack_task_deinit

原型：void ble_stack_task_deinit(void)

功能：删除 BLE 任务

输入参数：无

输出参数：无

返回值：无

2.12.5. ble_app_task_init

原型：uint32_t ble_app_task_init(uint32_t stack_size, uint32_t priority)

功能：BLE APP 任务的初始化

输入参数：stack_size，任务栈大小，单位为 4 个字节

priority，任务的优先级

输出参数：无

返回值：成功返回 0，失败返回其他值

2.12.6. ble_app_task_deinit

原型：void ble_app_task_deinit(void)

功能：删除 BLE APP 任务

输入参数：无

输出参数：无

返回值：无

2.12.7. ble_local_app_msg_send

原型：bool ble_local_app_msg_send (void *p_msg, uint16_t msg_len)

功能：上层若需异步处理一些消息，可通过该函数向 BLE APP task 发送消息，在调用 ble_app_msg_hdl_reg 注册的回调函数中处理该消息

输入参数：p_msg，消息内容

msg_len，消息内容长度

输出参数：无

返回值：成功返回 true，失败返回 false

2.12.8. ble_app_msg_hdl_reg

原型：void ble_app_msg_hdl_reg(ble_app_msg_hdl_t p_hdl)

功能：注册 APP 消息回调函数，与 ble_local_app_msg_send 配套使用

输入参数：p_hdl，回调函数

输出参数：无

返回值：无

2.12.9. ble_sleep_mode_set

原型：void ble_sleep_mode_set(uint8_t mode)

功能：设置 BLE 的 sleep 模式

输入参数：mode，0 表示一直为 active 状态；1 表示无任务处理时会进入睡眠模式

输出参数：无

返回值：无

2.12.10. ble_sleep_mode_get

原型：uint8_t ble_sleep_mode_get(void)

功能：获取 BLE 的 sleep 模式

输入参数：无

输出参数：无

返回值：mode，0 表示一直为 active 状态；1 表示无任务处理时会进入睡眠模式

2.12.11. ble_core_is_deep_sleep

原型：bool ble_core_is_deep_sleep(void)

功能：查询 BLE core 当前是否在 deep sleep 模式下

输入参数：无

输出参数：无

返回值：true 为在 deep sleep 模式下，否则为 flase

2.12.12. ble_modem_config

原型：void ble_modem_config(void)

功能：配置 BLE core 下的 modem 参数，每次 BLE core 睡眠唤醒后需配置

输入参数：无

输出参数：无

返回值：无

2.12.13. ble_work_status_set

原型：void ble_work_status_set(enum ble_work_status_t mode)

功能：设置 BLE 的工作状态，通过它可动态开关 BLE

参考基本命令中 ble_enable 和 ble_disable

输入参数：mode，0 为 enable；1 为 disable

输出参数：无

返回值：无

2.12.14. ble_ work_status_get

原型：ble_work_status_t ble_ work_status_get(void)

功能：获取 BLE 的工作状态

输入参数：无

输出参数：无

返回值：mode， 0 为 enable；1 为 disable

2.12.15. ble_internal_encode

原型：void ble_internal_encode(uint8_t *data, uint16_t len, uint8_t rand)

功能：对数据使用内部算法进行编码

输入参数：data，输入的数据

len，输入数据的长度

rand，随机数，相同的输入通过随机数可输出不同的值

输出参数：data，编码后的数据

返回值：无

2.12.16. ble_internal_decode

原型：void ble_internal_decode(uint8_t *data, uint16_t len, uint8_t rand)

功能：对数据使用内部算法进行解码

输入参数：data，输入的数据

len，输入数据的长度

rand，随机数，相同的输入通过随机数可输出不同的值

输出参数：data，解码后的数据

返回值：无

2.12.17. ble_register_hci_uart

原型：bool ble_register_hci_uart(ble_uart_func_t *p_func)

功能：注册 HCI uart 相关回调函数，用于 controller only 的特殊 lib 场景

输入参数：p_func，uart 功能回调函数，参考 ble_uart_func_t

输出参数：无

返回值：true 为注册成功，否则 false

3. 应用举例

3.1. 扫描

BLE 扫描功能用于查找周围环境中的低功耗蓝牙设备。使能扫描功能后，将会把扫描到的设备上报至应用层。

该功能快速使用主要分以下几步：

1. 注册 event 处理函数，可处理扫描状态的变化和广播数据的上报。

表 3-1. 扫描 event 处理函数示例代码

```
static void ble_app_scan_mgr_evt_handler(ble_scan_evt_t  event, ble_scan_data_u  *p_data)
{
    sw itch (event) {
    case BLE_SCAN_EVT_STATE_CHG:
        if (p_data->scan_state.scan_state == BLE_SCAN_STATE_ENABLED)  {
            dbg_print(NOTICE,  "Ble Scan enabled status 0x%x\r\n", p_data->scan_state.reason);
        } else if (p_data->scan_state.scan_state  == BLE_SCAN_STATE_ENABLING)  {
            scan_mgr_clear_dev_list();
        } else if (p_data->scan_state.scan_state  == BLE_SCAN_STATE_DISABLED)  {
            dbg_print(NOTICE, "Ble Scan disabled status 0x%x\r\n",
p_data->scan_state.reason);
        }
        break;

    case BLE_SCAN_EVT_ADV_RPT:
        scan_mgr_report_hdlr(p_data->p_adv_rpt);
        break;
    }
}
```

2. 通过 ble_scan_param_set 配置扫描参数，结构体参数如下：

type---扫描类型，可设置为 general discovery(通用扫描)、limit discovery(限制扫描)等。

prop---扫描属性，可设置 1M 和 CODED PHY 的主动扫描或被动扫描以及 filter 策略等。

dup_filt_pol---重复过滤，使能后，不会把收到的广播信号重复上报至 application。

scan_intv---扫描间隔，控制器间隔多长时间扫描一次。

scan_win---扫描窗口，每一次扫描持续的时间。

duration---扫描时长，配置 0 表示持续扫描。

period---是否周期扫描，以 duration 为周期。

表 3-2. 配置扫描参数示例代码

```
/**@brief Function for set scan parameters.
 *
 * @param[in] param                    scan parameters (see enum #ble_gap_scan_param_t)
 * @retval BLE_ERR_NO_ERROR                If ble scan module disable successfully.
 */
ble_status_t ble_scan_param_set(ble_gap_scan_param_t *p_param);

/** 默认的扫描参数如下*/
p_ble_scan_env->param.type = BLE_GAP_SCAN_TYPE_GEN_DISC;
p_ble_scan_env->param.prop = BLE_GAP_SCAN_PROP_PHY_1M_BIT |
                             BLE_GAP_SCAN_PROP_ACTIVE_1M_BIT |
                             BLE_GAP_SCAN_PROP_PHY_CODED_BIT |
                             BLE_GAP_SCAN_PROP_ACTIVE_CODED_BIT;
p_ble_scan_env->param.dup_filt_pol = BLE_GAP_DUP_FILT_EN;
p_ble_scan_env->param.scan_intv_1m = 160; // 100ms
p_ble_scan_env->param.scan_intv_coded = 160; // 100ms
p_ble_scan_env->param.scan_win_1m = 48;  // 30ms
p_ble_scan_env->param.scan_win_coded = 48;  // 30ms
p_ble_scan_env->param.duration = 0;
p_ble_scan_env->param.period = 0;
```

3. 使能扫描，调用 ble_scan_enable API 即可开启扫描。

表 3-3. 使能扫描示例代码

```
void app_scan_enable(bool update_rssi)
{
    if (ble_scan_enable() != BLE_ERR_NO_ERROR) {
        dbg_print(NOTICE, "app_scan_enable fail!\r\n");
        return;
    }
}
```

3.2. 广播

BLE 广播功能用于发送广播报文，可以让周围的低功耗蓝牙设备发现并连接或者发送周期性数据等。可配置为 legacy advertising（传统广播）、extened advertising（扩展广播）、periodic advertising（周期广播）。

该功能快速使用主要分以下几步：

1. 注册 event 处理函数，用来处理广播状态的变化、收到的扫描请求的上报。

表 3-4. 广播 event 处理函数示例代码

```
static void app_adv_mgr_evt_hdlr(ble_adv_evt_t adv_evt, void *p_data, void *p_context)
```

```
{
    app_adv_actv_t *p_adv = (app_adv_actv_t *)p_context;

    switch (adv_evt) {
    case BLE_ADV_EVT_STATE_CHG: {
        ble_adv_state_chg_t *p_chg = (ble_adv_state_chg_t *)p_data;
        ble_adv_state_t old_state = p_adv->state;

        dbg_print(NOTICE, "adv state change 0x%x ==> 0x%x, reason 0x%x\r\n", old_state,
p_chg->state, p_chg->reason);

        p_adv->state = p_chg->state;

        if ((p_chg->state == BLE_ADV_STATE_CREATE) && (old_state ==
BLE_ADV_STATE_CREATING)) {
            p_adv->idx = p_chg->adv_idx;
            app_print("adv index %d\r\n", p_adv->idx);

            app_adv_start(p_adv);
        } else if ((p_chg->state == BLE_ADV_STATE_CREATE) && (old_state ==
BLE_ADV_STATE_START)) {
            dbg_print(NOTICE, "adv stopped, remove %d\r\n", p_adv->remove_after_stop);

            if (p_adv->remove_after_stop) {
                ble_adv_remove(p_adv->idx);
                p_adv->remove_after_stop = false;
            }
        } else if (p_chg->state == BLE_ADV_STATE_IDLE) {
            free_adv_actv(p_adv);
        }
    } break;

    case BLE_ADV_EVT_DATA_UPDATE_RSP: {
        ble_adv_data_update_rsp_t *p_rsp = (ble_adv_data_update_rsp_t *)p_data;
        dbg_print(NOTICE, "adv data update rsp, type %d, status 0x%x\r\n", p_rsp->type,
p_rsp->status);
    } break;

    case BLE_ADV_EVT_SCAN_REQ_RCV: {
        ble_adv_scan_req_rcv_t *p_req = (ble_adv_scan_req_rcv_t *)p_data;
        dbg_print(NOTICE, "scan req rcv, device addr %02X:%02X:%02X:%02X:%02X:%02X\r\n",
              p_req->peer_addr.addr[5], p_req->peer_addr.addr[4], p_req->peer_addr.addr[3],
              p_req->peer_addr.addr[2], p_req->peer_addr.addr[1], p_req->peer_addr.addr[0]);
    } break;

    default:
        break;
    }
}
```

2. 设备打出广播报文主要分两步执行：创建广播和使能广播。在广播状态为创建成功后才能使能广播。例如以下应用层创建广播代码，基于不同的广播类型配置不同的广播参数。

表 3-5. 创建广播示例代码

```
ble_status_t app_adv_create(app_adv_param_t  *p_param)
{
    app_adv_actv_t *p_adv;
    ble_adv_param_t  adv_param = {0};

    p_adv = get_free_adv_actv();
    if (p_adv == NULL) {
        return BLE_ERR_NO_RESOURCES;
    }

    p_adv->max_data_len  = p_param->max_data_len;

    adv_param.param.own_addr_type  = p_param->own_addr_type;

    if (p_param->type  == BLE_ADV_TYPE_LEGACY)  {
        adv_param.param.type  = BLE_GAP_ADV_TYPE_LEGACY;
        adv_param.param.prop  = p_param->prop;

        if (p_param->wl_enable)  {
            adv_param.param.filter_pol  = BLE_GAP_ADV_ALLOW_SCAN_FAL_CON_FAL;
            adv_param.param.disc_mode  = BLE_GAP_ADV_MODE_NON_DISC;
        } else {
            adv_param.param.filter_pol  = BLE_GAP_ADV_ALLOW_SCAN_ANY_CON_ANY;
            adv_param.param.disc_mode  = p_param->disc_mode;
        }

        adv_param.param.ch_map  = APP_ADV_CHMAP;
        adv_param.param.primary_phy  = p_param->pri_phy;
    } else if (p_param->type  == BLE_ADV_TYPE_EXTENDED)  {
        adv_param.param.type  = BLE_GAP_ADV_TYPE_EXTENDED;
        adv_param.param.prop  = p_param->prop;

        if (p_param->wl_enable)  {
            adv_param.param.filter_pol  = BLE_GAP_ADV_ALLOW_SCAN_FAL_CON_FAL;
            adv_param.param.disc_mode  = BLE_GAP_ADV_MODE_NON_DISC;
        } else {
            adv_param.param.filter_pol  = BLE_GAP_ADV_ALLOW_SCAN_ANY_CON_ANY;
            adv_param.param.disc_mode  = p_param->disc_mode;
        }

        adv_param.param.ch_map  = APP_ADV_CHMAP;
        adv_param.param.primary_phy  = p_param->pri_phy;
        adv_param.param.adv_sid  = get_adv_sid();
        adv_param.param.max_skip  = 0x00;
        adv_param.param.secondary_phy  = p_param->sec_phy;
    } else {
        return BLE_GAP_ERR_INVALID_PARAM;
    }

    if (adv_param.param.prop  & BLE_GAP_ADV_PROP_DIRECTED_BIT)  {
        adv_param.param.peer_addr  = p_param->peer_addr;
        adv_param.param.disc_mode  = BLE_GAP_ADV_MODE_NON_DISC;
        p_adv->peer_addr = p_param->peer_addr;
    }

    if (adv_param.param.prop  & BLE_GAP_ADV_PROP_ANONYMOUS_BIT)  {
        adv_param.param.disc_mode  = BLE_GAP_ADV_MODE_NON_DISC;
    }

    p_adv->disc_mode  = adv_param.param.disc_mode;

    adv_param.param.adv_intv_min  = APP_ADV_INT_MIN;
    adv_param.param.adv_intv_max  = APP_ADV_INT_MAX;

    if (p_adv->disc_mode  == BLE_GAP_ADV_MODE_LIM_DISC)  {
        adv_param.param.duration  = 1000;           // 10s
    }

    if (p_param->type  != BLE_ADV_TYPE_LEGACY)  {
        adv_param.include_tx_pwr  = true;
        adv_param.scan_req_ntf  = true;
    }

    return ble_adv_create(&adv_param,  app_adv_mgr_evt_hdlr,  p_adv);
}
```

3. 使能广播。在注册的 event 处理函数中收到创建广播成功的消息后就可调用 ble_adv_start 接口来使能广播。之后，在 event 处理函数中收到上报广播状态为

BLE_ADV_STATE_START 表示使能成功。

ble_adv_start API 中后三个参数分别用来设置广播数据，扫描响应数据和周期性广播数据，可由应用层来直接设置内容，也可通过配置参数由 BLE ADV 模块来封装。例如以下 code，所有数据内容都是直接由应用层来设置的。

表 3-6. 使能广播示例代码

```
static uint8_t adv_data_1[7]  = {0x06, 0x16, 0x52, 0x18, 0x18, 0x36, 0x9A};
static uint8_t per_data_1[52]  = {0x33, 0x16, 0x51, 0x18, 0x40, 0x9c, 0x00, 0x01, 0x02, 0x06,
                                   0x00, 0x00, 0x00, 0x00, 0x0d, 0x02, 0x01, 0x08, 0x02, 0x02,
                                   0x01, 0x03, 0x04, 0x78, 0x00, 0x02, 0x05, 0x01, 0x07, 0x03,
                                   0x02, 0x04, 0x00, 0x02, 0x04, 0x80, 0x01, 0x06, 0x05, 0x03,
                                   0x00, 0x04, 0x00, 0x00, 0x02, 0x06, 0x05, 0x03, 0x00, 0x08,
                                   0x00, 0x00
                                  };

static void app_adv_start(app_adv_actv_t  *p_adv)
{
    ble_adv_data_set_t  adv;
    ble_adv_data_set_t  scan_rsp;
    ble_adv_data_set_t  per_adv;
    ble_data_t adv_data;
    ble_data_t per_adv_data;

    adv.data_force = true;
    scan_rsp.data_force = true;
    per_adv.data_force  = true;

    adv_data.len  = 7;
    adv_data.p_data  = adv_data_1;

    per_adv_data.len  = 52;
    per_adv_data.p_data  = per_data_1;

    adv.data.p_data_force  = &adv_data;
    scan_rsp.data.p_data_force  = &adv_data;
    per_adv.data.p_data_force  = &per_adv_data;

    ble_adv_start(p_adv->idx, &adv, &scan_rsp, &per_adv);
}
```

3.3. GATT server 应用

GD32VW553 SDK 针对 BLE GATT server role 提供了添加/删除 service、发送

notification/indication 等功能，用户可根据需要实现特定的 service，具体 API 可以参考 ***BLE gatts API***。下面以 DIS 为例说明如何使用这些 API 实现一个 service server，文件为 MSDK\ble\profile\dis\ble_diss.c。

3.3.1. service 添加

向 BLE GATT server 模块中添加 service 主要通过 ble_gatts_svc_add 函数完成，该函数的输入参数包括 service UUID，service attribute database，GATT server 消息的 callback 处理函数等内容。Service UUID 可以为 16 bit，32 bit 或者 128bit 中的任意一种，需要在 info 参数中予以说明，如 ble diss 的 code，service UUID 使用的是 UUID 16，调用 ble_gatts_svc_add 函数时就用 SVC_UUID(16)进行说明。

表 3-7. 添加 service 示例代码

```
ret = ble_gatts_svc_add(&ble_diss_svc_id, ble_dis_uuid, 0, SVC_UUID(16), ble_diss_attr_db,
                              BLE_DIS_HDL_NB, ble_diss_srv_cb);
```

3.3.2. service attribute database

Service attribute database 是由一系列 ble_gatt_attr_desc_t 元素组成的数组，数组中的每一个元素就是一个 attribute，可以是 primary service，characteristic declaration，characteristic value declaration 等内容，用户可以根据不同 service 的需求进行自由组合。

每一个 attribute 由 UUID 和其属性说明组成，DIS 所有的 attribute 都是只读的，所以只需要指定 RD property 就可以，对于 characteristic value declaration，还可以指定 value 的最大 size。

表 3-8. service database 示例代码

```
const ble_gatt_attr_desc_t ble_diss_attr_db[BLE_DIS_HDL_NB] =
{
    [BLE_DIS_HDL_SVC] = {UUID_16BIT_TO_ARRAY(BLE_GATT_DECL_PRIMARY_SERVICE),
PROP(RD), 0},

    [BLE_DIS_HDL_MANUFACT_NAME_CHAR] =
{UUID_16BIT_TO_ARRAY(BLE_GATT_DECL_CHARACTERISTIC), PROP(RD), 0},
    [BLE_DIS_HDL_MANUFACT_NAME_VAL] =
{UUID_16BIT_TO_ARRAY(BLE_DIS_CHAR_MANUF_NAME), PROP(RD),
BLE_DIS_VAL_MAX_LEN},

    [BLE_DIS_HDL_MODEL_NB_CHAR] =
{UUID_16BIT_TO_ARRAY(BLE_GATT_DECL_CHARACTERISTIC), PROP(RD), 0},
    [BLE_DIS_HDL_MODEL_NB_VAL] =
{UUID_16BIT_TO_ARRAY(BLE_DIS_CHAR_MODEL_NB), PROP(RD),
BLE_DIS_VAL_MAX_LEN},

    [BLE_DIS_HDL_SERIAL_NB_CHAR] =
{UUID_16BIT_TO_ARRAY(BLE_GATT_DECL_CHARACTERISTIC), PROP(RD), 0},
    [BLE_DIS_HDL_SERIAL_NB_VAL] =
{UUID_16BIT_TO_ARRAY(BLE_DIS_CHAR_SERIAL_NB), PROP(RD),
BLE_DIS_VAL_MAX_LEN},

    [BLE_DIS_HDL_HARD_REV_CHAR] =
{UUID_16BIT_TO_ARRAY(BLE_GATT_DECL_CHARACTERISTIC), PROP(RD), 0},
    [BLE_DIS_HDL_HARD_REV_VAL] = {UUID_16BIT_TO_ARRAY(BLE_DIS_CHAR_HW_REV),
PROP(RD), BLE_DIS_VAL_MAX_LEN},

    [BLE_DIS_HDL_FIRM_REV_CHAR] =
{UUID_16BIT_TO_ARRAY(BLE_GATT_DECL_CHARACTERISTIC), PROP(RD), 0},
    [BLE_DIS_HDL_FIRM_REV_VAL] = {UUID_16BIT_TO_ARRAY(BLE_DIS_CHAR_FW_REV),
PROP(RD), BLE_DIS_VAL_MAX_LEN},

    [BLE_DIS_HDL_SW_REV_CHAR] =
{UUID_16BIT_TO_ARRAY(BLE_GATT_DECL_CHARACTERISTIC), PROP(RD), 0},
    [BLE_DIS_HDL_SW_REV_VAL] = {UUID_16BIT_TO_ARRAY(BLE_DIS_CHAR_SW_REV),
PROP(RD), BLE_DIS_VAL_MAX_LEN},

    [BLE_DIS_HDL_SYSTEM_ID_CHAR] =
{UUID_16BIT_TO_ARRAY(BLE_GATT_DECL_CHARACTERISTIC), PROP(RD), 0},
    [BLE_DIS_HDL_SYSTEM_ID_VAL] = {UUID_16BIT_TO_ARRAY(BLE_DIS_CHAR_SYS_ID),
PROP(RD), BLE_DIS_SYS_ID_LEN},

    [BLE_DIS_HDL_IEEE_CHAR] =
{UUID_16BIT_TO_ARRAY(BLE_GATT_DECL_CHARACTERISTIC), PROP(RD), 0},
    [BLE_DIS_HDL_IEEE_VAL] = {UUID_16BIT_TO_ARRAY(BLE_DIS_CHAR_IEEE_CERTIF),
PROP(RD), BLE_DIS_VAL_MAX_LEN},

    [BLE_DIS_HDL_PNP_ID_CHAR] =
{UUID_16BIT_TO_ARRAY(BLE_GATT_DECL_CHARACTERISTIC), PROP(RD), 0},
   [BLE_DIS_HDL_PNP_ID_VAL] = {UUID_16BIT_TO_ARRAY(BLE_DIS_CHAR_PNP_ID),
PROP(RD), BLE_DIS_PNP_ID_LEN},
};
```

3.3.3. service attribute 读写处理

ble_gatts_svc_add 最后一个参数是注册一个 GATT server event callback 函数，当对端 client 对该 service 进行 read、write 操作时就会执行该 callback 函数，消息类型为 BLE_SRV_EVT_GATT_OPERATION，subevent 为 BLE_SRV_EVT_READ_REQ 或 BLE_SRV_EVT_WRITE_REQ。subevent 数据类型为 ble_gatts_read_req_t 或者 ble_gatts_write_req_t，其中有 att_idx 参数表明当前操作对应的 attribute 在注册的 service attribute database 数组中对应的 index。

表 3-9. attribute 读写函数示例代码

```
ble_status_t ble_diss_srv_cb(ble_gatts_msg_info_t  *p_srv_msg_info)
{
    uint8_t attr_idx  = 0;
    uint16_t len  = 0;
    uint8_t attr_len  = 0;
    uint8_t *p_attr  = NULL;

    if (p_srv_msg_info->srv_msg_type  == BLE_SRV_EVT_GATT_OPERATION)  {
        if (p_srv_msg_info->msg_data.gatts_op_info.gatts_op_sub_evt  ==
BLE_SRV_EVT_READ_REQ)  {
            ble_gatts_read_req_t  * p_read_req  =
&p_srv_msg_info->msg_data.gatts_op_info.gatts_op_data.read_req;

            attr_idx  = p_read_req->att_idx;
            switch (attr_idx) {
            case BLE_DIS_HDL_MANUFACT_NAME_VAL:  {
                p_attr      = ble_diss_val.manufact_name;
                attr_len  = ble_diss_val.manufact_name_len;
            } break;

            case BLE_DIS_HDL_MODEL_NB_VAL:  {
                p_attr      = ble_diss_val.model_num;
                attr_len  = ble_diss_val.model_num_len;
            } break;

            case BLE_DIS_HDL_SERIAL_NB_VAL:  {
                p_attr      = ble_diss_val.serial_num;
                attr_len  = ble_diss_val.serial_num_len;
            } break;

            case BLE_DIS_HDL_HARD_REV_VAL:  {
                p_attr      = ble_diss_val.hw_rev;
                attr_len  = ble_diss_val.hw_rev_len;
            } break;

            case BLE_DIS_HDL_FIRM_REV_VAL:  {
                p_attr      = ble_diss_val.fw_rev;
                attr_len  = ble_diss_val.fw_rev_len;
            } break;

            case BLE_DIS_HDL_SW_REV_VAL:  {
                p_attr      = ble_diss_val.sw_rev;
                attr_len = ble_diss_val.sw_rev_len;
            } break;

            case BLE_DIS_HDL_SYSTEM_ID_VAL: {
                p_attr     = ble_diss_val.sys_id;
                attr_len = BLE_DIS_SYS_ID_LEN;
            } break;

            case BLE_DIS_HDL_IEEE_VAL: {
                p_attr     = ble_diss_val.ieee_data;
                attr_len = ble_diss_val.ieee_data_len;
            } break;

            case BLE_DIS_HDL_PNP_ID_VAL: {
                p_attr     = ble_diss_val.pnp_id;
                attr_len = BLE_DIS_PNP_ID_LEN;
            } break;

            default:
                return BLE_ATT_ERR_INVALID_HANDLE;
            }

            if (p_read_req->offset > attr_len) {
                return BLE_ATT_ERR_INVALID_OFFSET;
            }

            len = ble_min(p_read_req->max_len, attr_len - p_read_req->offset);
            p_read_req->val_len = len;
            memcpy(p_read_req->p_val, p_attr, len);
        }
    }

    return BLE_ERR_NO_ERROR;
}
```

如果 service 中有 attribute 支持 Client Characteristic Configuration declaration(CCCD)，在使能后，可调用 ble_gatts_ntf_ind_send 接口发送 notification/indication。

表 3-10. 发送 notification 示例代码

```
static void bcwl_ntf_event_send(uint8_t *p_val, uint16_t len)
{
    if (bcwl_env.ntf_cfg == 0) {
        dbg_print(ERR, "%s fail\r\n", __func__);
        return;
    }

    ble_gatts_ntf_ind_send(bcwl_env.conn_id, bcwl_env.prf_id, BCW_IDX_NTF, p_val, len,
BLE_GATT_NOTIFY);
}
```

3.4. BLE 配网

Blue courier 是一项基于 BLE 的 WIFI 网络配置功能。通过协议将 WIFI 的 SSID、密码、信道、加密类型等传输到 GD 设备，基于这些信息，GD 设备可连接到 AP 或者建立 SoftAP。链路上支持数据的分片及 CRC16 完整性校验，安全上依赖 BLE 链路上的加密，其次对于涉及 SSID 和密码的消息传输，使用了编码方式避免空气中传输明文。使用参考《AN153 GD32VW553 基本指令用户指南》的 ble_courier_wifi 命令。

3.4.1. Blue courier 流程

下面以配置 WIFI 作为 station 连接 AP 举例，介绍了广播、连接、服务发现、使能通知、握手、传输数据、回传连接状态等关键步骤。

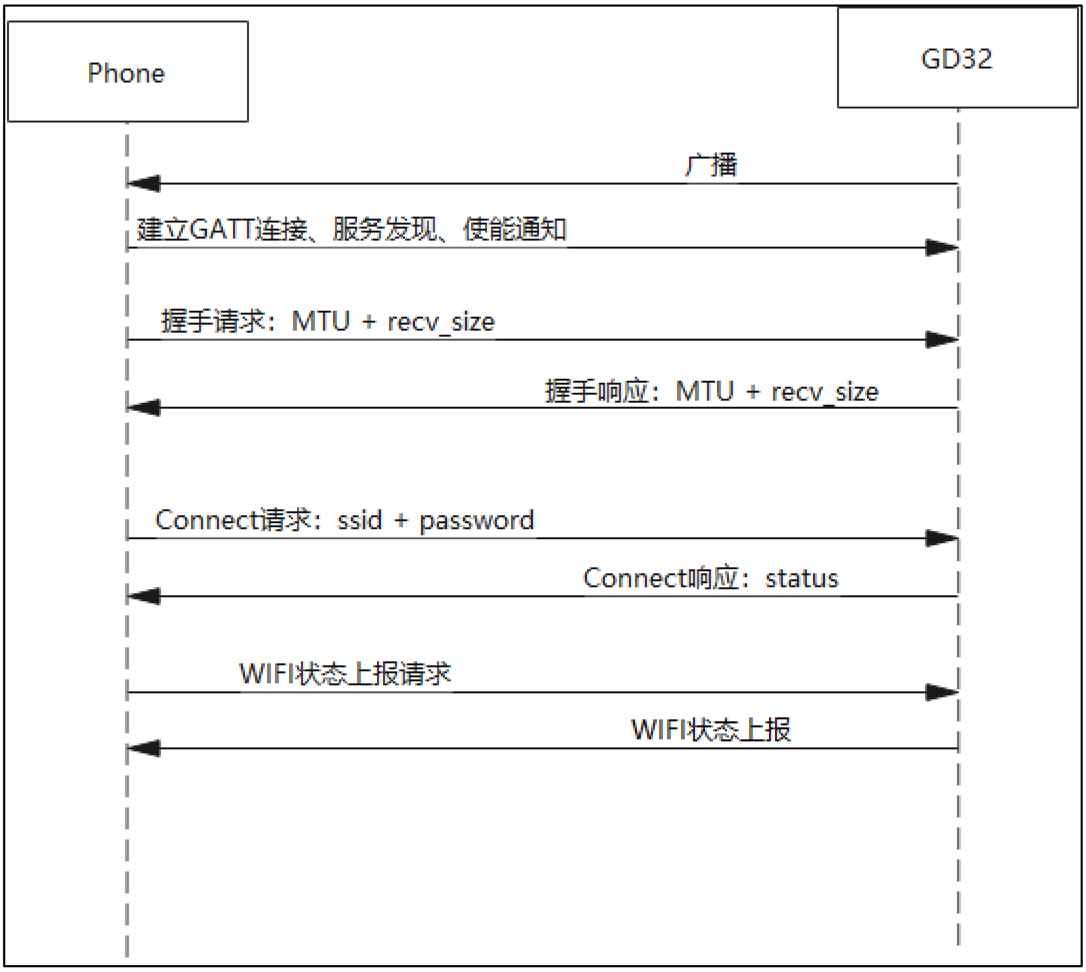
1. GD 设备开启 Blue courier wifi 后，会向 GATT server 模块注册特定的 service，发送带有特定 advertising data 的广播。广播内容可以按需对其进行自定义。

2. 使用微信小程序搜索到该广播后，手机将作为 GATT Client 连接 GD 设备。

3. 成功建立 GATT 连接后，手机会向 GD 设备发送握手请求消息，GD 设备收到握手消息后将回复握手响应消息。

4. 手机可向 GD 设备发送连接 WIFI、创建 SoftAp、获取 WIFI 状态等消息。

图 3-1. Blue courier 流程图



3.4.2. GATT 相关说明

配网 service 的添加可以参考 *service 添加*中的说明。

配网 service 使用的 UUID 说明见*表 3-11. 配网 service UUID 错误!未找到引用源。*。

表 3-11. 配网 service UUID

Attribute	Description
Blue courier WIFI Service	UUID = 0000FFF0-0000-1000-8000-00805F9B34FB
C1 Characteristic (Client TX Buffer)	UUID = 0000FFF1-0000-1000-8000-00805F9B34FB Characteristic Properties = Write max length = 256 bytes Security level=unauth(要求链路必须加密，第一次连接需配对)
C2 Characteristic (Client RX Buffer)	UUID = 0000FFF2-0000-1000-8000-00805F9B34FB Characteristic Properties = Notify max length = 256 bytes

3.4.3. 广播数据

为了便于其他设备发现本地设备支持BLE配网功能，在广播数据中必须携带Bluecourier WIFI Service UUID，对端可以在搜索 BLE 设备时基于Service UUID 进行过滤，具体内容见。

表 3-12. 广播数据中service UUID 内容

Byte	Value	Description
0	0x03	AD[0] Length == 3 bytes
1	0x03	AD[0] Type == 1 (Flags) Complete list of 16 bit service UUIDs.
2-3	0xFFF0	16-bit Blue courier WIFI Service UUID

3.4.4. 帧格式

Blue courier 手机应用程序与 GD 设备之间的通信帧格式如下：

表 3-13. blue courier 帧格式

字段	大小(字节)
flag	1
sequence	1
opcode	1
data_len	1
data	${data_len}
crc	2

flag

帧控制字段，占 1 字节，每个位表示不同含义，具体内容见下表：

表 3-14. 帧控制字段

位	含义
0x01	Begin：表示是否为首个分片报文。 ● 0 表示此帧为剩下分片报文。 ● 1 表示此帧为首个分片报文。 ● 分片报文用来传输较长的数据。只有分片包的首包数据字段前 2 字节为数据内容总长度，并用于指示对端接收分配的内存大小。即 data = total_len + data
0x02	End：表示是否为最后一个分片报文。 ● 0 表示此帧不是最后一个分片报文。 ● 1 表示此帧为最后一个分片报文。 若 Begin 和 End bit 同时置 1，表示该报文为非分片报文。
0x04	ACK：表示是否要求对方回复 ACK。

	● 0 表示不要求回复 ACK。 ● 1 表示要求回复 ACK。
0x08~0x80	保留

sequence

序列控制字段。帧发送时，无论帧的类型是什么，序列都会自动加 1，用来防止重放攻击 (Replay Attack)。每次重新连接后，序列清零。

opcode

opcode 字段占 1 字节，分为类型和子类型两部分。其中，类型占高 2 位，表明该帧为管理帧或数据帧；子类型占低 6 位，表示此管理帧或数据帧的具体含义。

1. 管理帧（二进制：0x0 b’00）。

表 3-15. 管理帧内容

管理帧	含义	解释	内容
0x0 (b'000000)	握手	握手用于交换两端的 mtu 及最大接收长度，决定分片包大小及最大报文总长度。mtu 两者取小作为两端分片大小，recv_size 为对端最大接收长度，接收方应将其作为发送最大长度。	共占 4 个字节，mtu 和 recv_size 均占 2 字节。 手机->GD 设备: mtu + recv_size GD 设备->手机： mtu + recv_size
0x1 (b'000001)	ACK	ACK 帧的数据字段使用回复对象帧的序列值。	数据字段占用 1 字节，其序列值与回复对象帧的序列值相同。
0x2 (b'000100)	错误上报	用于向对端上报错误。错误码可自定义。	status: 1byte

2. 数据帧 (二进制：0x1 b’01)。

表 3-16. 数据帧内容

数据帧	含义	解释	备注
0x0 (b'000000)	发送自定义数据。	用于向对端传输自定义数据，可用于测试。	
0x1 (b'000001)	获取 WIFI 扫描列表信息。	手机向 GD 设备发送该消息，内容长度应该为 0。GD 设备收到该消息后，会触发 wifi 扫描，并将扫描信息通过该消息发送给手机。	GD 设备->手机： 每组 ssid 结构为： len+rssi+mode+ssid len = 2byte(rssi+mode) + ssid 长度
0x2 (b'000010)	发送 STA 模式的连	发送 STA 设备要连接的 AP 的信息。GD 设备收到该消息后，触	手机->GD 设备: ssid_len + ssid + password_len +

	接请求	发 WIFI 连接并将连接结果传送给手机端。发送的数据需做编码处理，随机数可避免每次编码的数据相同。	password + random GD 设备->手机： status ssid_len, password_len, random, status: 1byte
0x3 (b'000011)	发送 STA 模式的关闭连接请求。	手机向 GD 设备发送该消息，内容长度应该为 0。GD 设备收到该消息后，会触发 WIFI 连接关闭，并将状态传送给手机端。	GD 设备->手机： status status: 1byte
0x4 (b'000100)	发送 SoftAP 模式的创建请求。	发送 STA 设备要创建的 AP 的信息。GD 设备收到该消息后，触发创建 softAp 并将创建结果传送给手机端。发送的数据需做编码处理，随机数可避免每次编码的数据相同。	手机->GD 设备: ssid_len + ssid + password_len + password + channel + akm + hide + random GD 设备->手机： status ssid_len, password_len, channel, akm, hide, random, status: 1byte
0x5 (b'000101)	发送 SoftAP 模式的停止请求。	手机向 GD 设备发送该消息，内容长度应该为 0。GD 设备收到该消息后，会触发停止 softAp，并将状态传送给手机端。	GD 设备->手机： status status: 1byte
0x6 (b'000110)	获取 wifi 状态。	手机向 GD 设备发送该消息，内容长度应该为 0。GD 设备收到该消息后，会将 wifi 状态上报至手机端。	会向手机回发一个报告 Wi-Fi 连接状态告知手机端当前所处的 设备模式、连接状态、SSID、信道等信息。消息内容结构参考 application 实现端。

crc

crc16 用于 Blue courier 通信中完整性校验，其计算基于 sequence、opcode、data_len、data 四部分。

4. 版本历史

表 4-1. 版本历史

版本号.	说明	日期
1.0	首次发布	2023 年 10 月 17 日
1.1	新增 ble_conn_enable_central_feat， ble_svc_data_save， ble_svc_data_load， ble_register_hci_uart 等接口 修改 ble_stack_init， ble_stack_task_init 等接口	2024 年 02 月 29 日
1.2	新增 ble_adp_callback_unregister， ble_adp_disable， ble_scan_callback_unregister， ble_sec_callback_unregister， ble_list_callback_unregister， ble_per_sync_callback_unregister， ble_gattc_svc_unreg, ble_stack_task_deinit， ble_app_task_deinit 等接口 修改 ble_stack_init， ble_stack_task_init 等接口 删除 ble_stack_task_suspend 接口	2024 年 07 月 10 日

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the fu nctionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary busi ness, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traff ic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Produ ct could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and sel ling the Products in accordance with the applicable laws and regulations. The Company is not liable , in whole or in part, and customers shall and hereby do release the Company as well as it’s suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as it’s suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for persona l injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, withoutn otice.

© 2023 GigaDevice – All rights reserved