RT58x BLE SDK User Guide V1.3

Introduction

This document is a reference for developing BLE applications using Rafael RT58x BLE SDK.
This version applies to the Rafael RT58x SDK 1.2 and higher.

Table of Contents

1. BLE SDK Architecture

BLE architecture is shown as below:

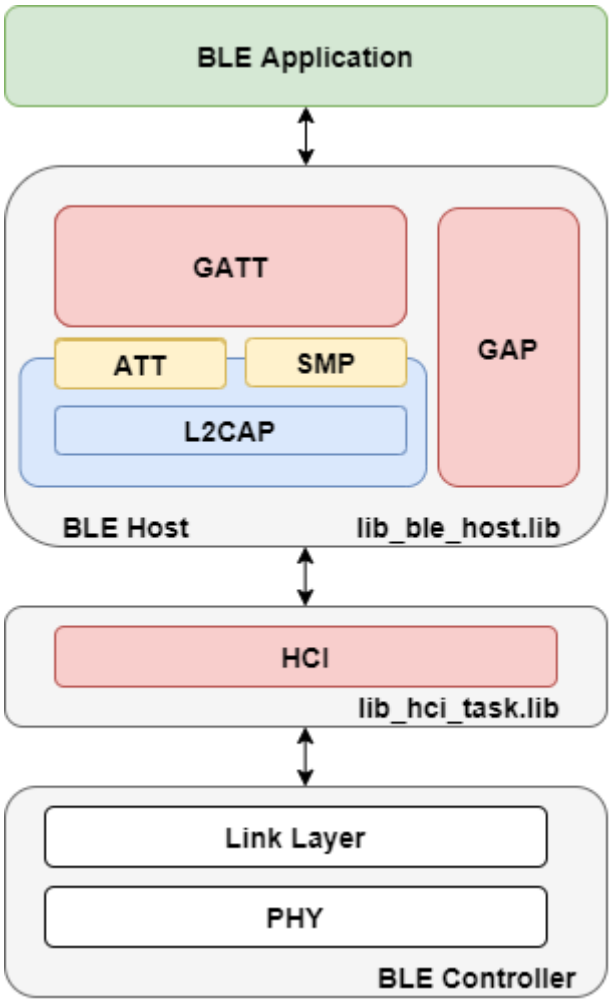


Figure 1. BLE Architecture

BLE upper layers, includes BLE application and BLE Host, communicates with BLE Controller via HCI commands in HCI layer.

GAP, L2CAP, ATT, GATT and SMP are built in library (lib_ble_host.lib) format in Rafael RT58x BLE SDK. And HCI command and event are built in library (lib_hci_task.lib). Applications can develop the required BLE functionality according to the provided API or format.

This section introduces Rafael RT58x BLE SDK folder hierarchy and task hierarchy (based on FreeRTOS) for BLE application.

1.1 BLE SDK Folder Hierarchy

1.1.1 BLE Sources

BLE SDK is in "*Middleware**BLE*", which includes BLE command, BLE services, and BLE host library is in "*Middleware**Prebuild*".

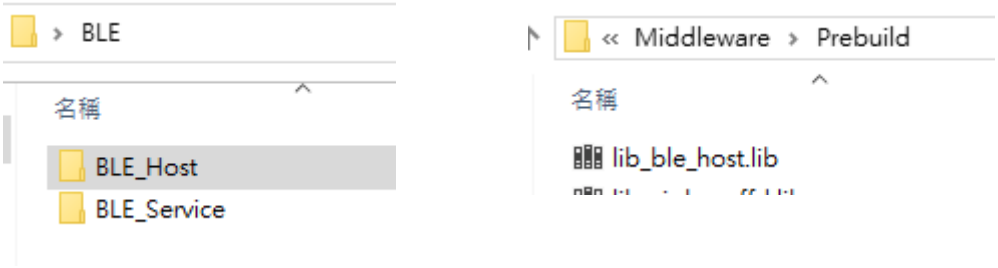


Figure 2. BLE SDK Hierarchy

- **BLE Command API & Definition**

  Different BLE APIs in the related folder.

  - **ble_cmd_ommon**: common definitions and APIs.
  - **ble_cmd_advertising**: Advertising related definitions and APIs.
  - **ble_cmd_scan**: Scan related definitions and APIs.
  - **ble_cmd_security_manager**: Security related definitions and APIs.
  - **ble_cmd_att_gatt**: ATT and GATT related definitions and APIs.
  - **ble_cmd_gap**: GAP related definitions and APIs.
  - **BLE_Host**: BLE host controller setting.
  - **BLE_Service**: contains the defined service related files with definitions and APIs.

(intentionally blank)

1.1.2 BLE Demo Projects

BLE Demo projects are in "*\Project\Application\BLE_Demo"*, which include Central, Peripheral and Multilink demos in the related folder.

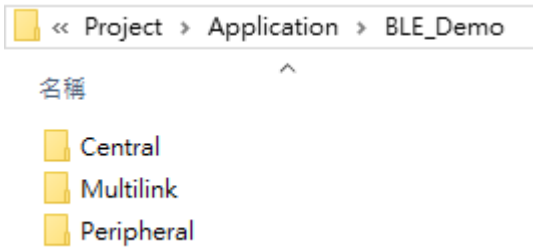


Figure 3. BLE Demo Hierarchy

More details for BLE demos please refer to "*RT58x_BLE_SDK_Application_Guide.pdf*".

(intentionally blank)

1.2 BLE System Architecture

BLE SDK created several tasks based on FreeRTOS which are BLE HCI task, BLE host task, BLE command transport task, handle app event & Notify task, and BLE application task. In addition to the BLE application task, all tasks are released in the form of a library.

1.2.1 BLE Task Definition and Priority

The priority of the tasks and mapping to the BLE SDK files are shown as below:

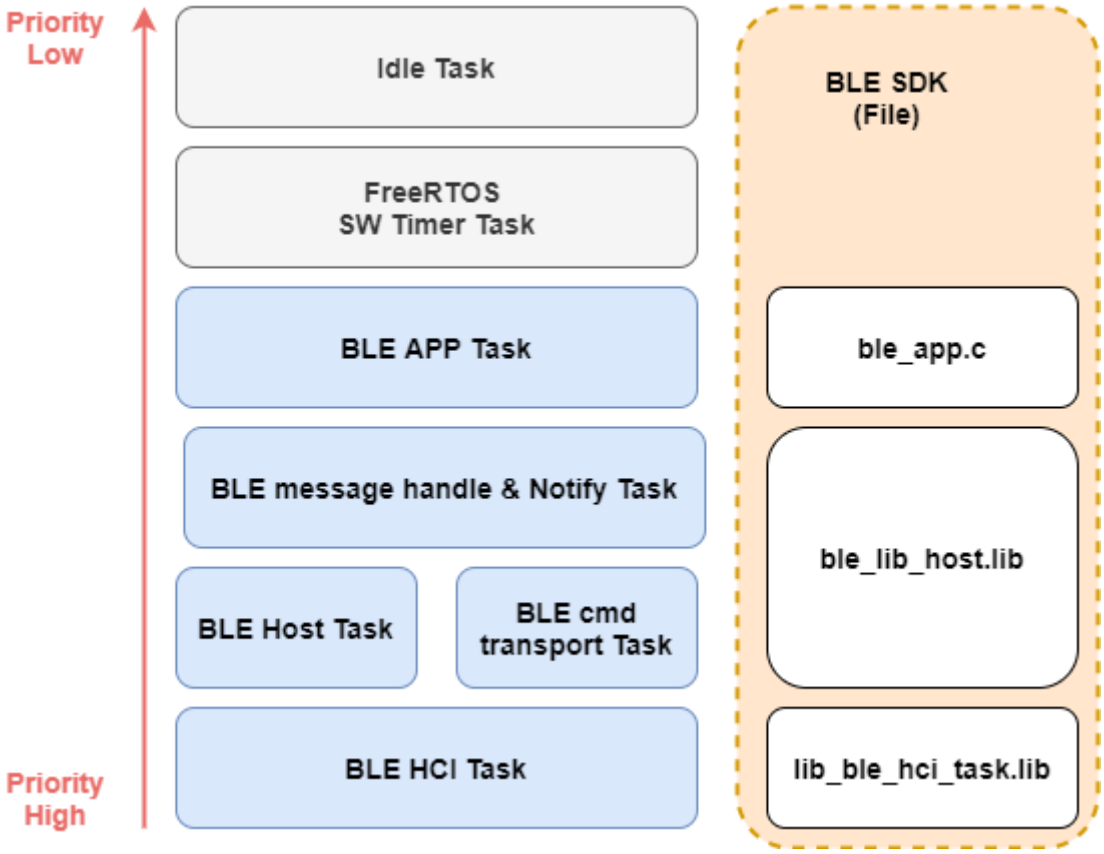


Figure 4. BLE Task Definition

1.2.1.1 BLE HCI Task

Handle HCI commands, ACL data and events.

1.2.1.2 BLE Host Task

BLE Host sub-system.

1.2.1.3 BLE cmd transport Task

Transport application command.

1.2.1.4 BLE message handle & Notify Task

Handle BLE general events and service events. And send the user message to the application task.

1.2.1.5 BLE Application Task

BLE application task.

(intentionally blank)

1.2.2 BLE Task Architecture

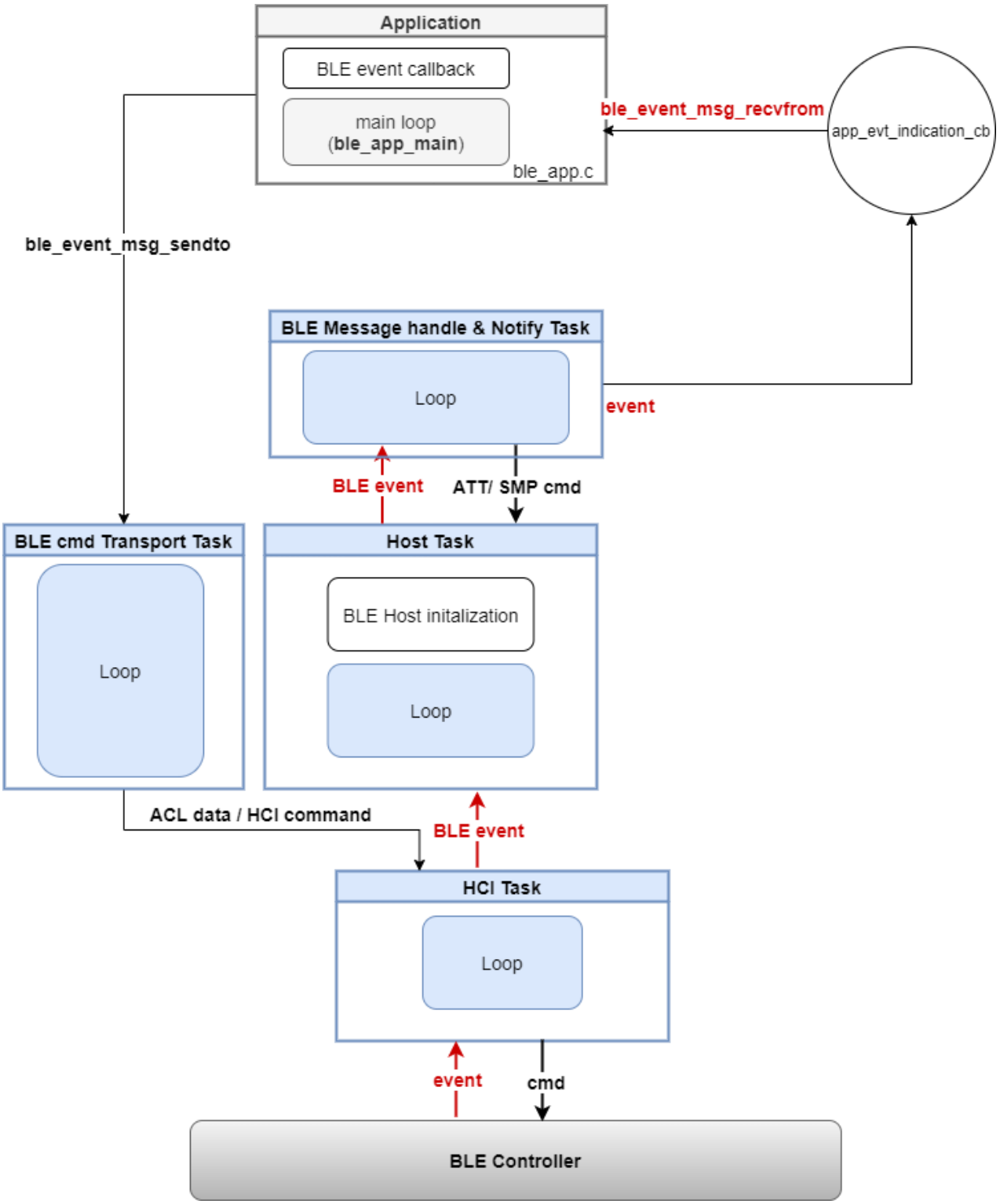


Figure 5. BLE Task Architecture

1.2.3 BLE Event Handling Flow

HCI task handles BLE events from BLE controller, and the event is triggered from ISR which is registered in HCI task file.

BLE Host task will process the event to do the related setting and also pass the event with the defined opcode to the BLE Message handle task by queue sending function. If the event needs to be passed to the application, it will call the callback function to the application. The application uses the “receive from” function to get the information.

BLE application task will handle the events. The BLE event with the defined opcode to application or pass BLE service event to application by the registered callback function. The application event opcode is defined in “ble_event.h”.

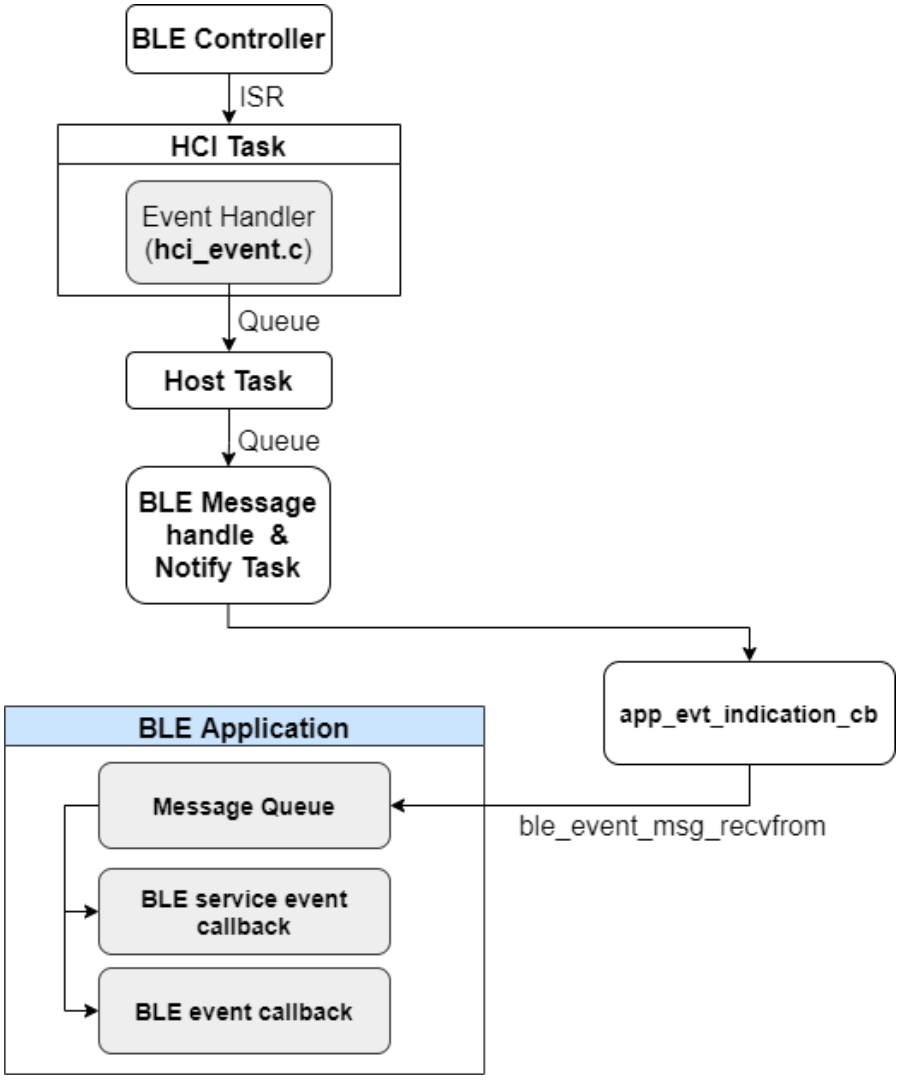


Figure 6. BLE Event Handling Flow

2. BLE Application Development

This section introduces the BLE application flow which is implemented in Rafael RT58x BLE SDK. User could follow the instruction in this section to develop a customized BLE application.

2.1 BLE Application Flow

The following figure shows the BLE application flow in Rafael RT58x BLE SDK.

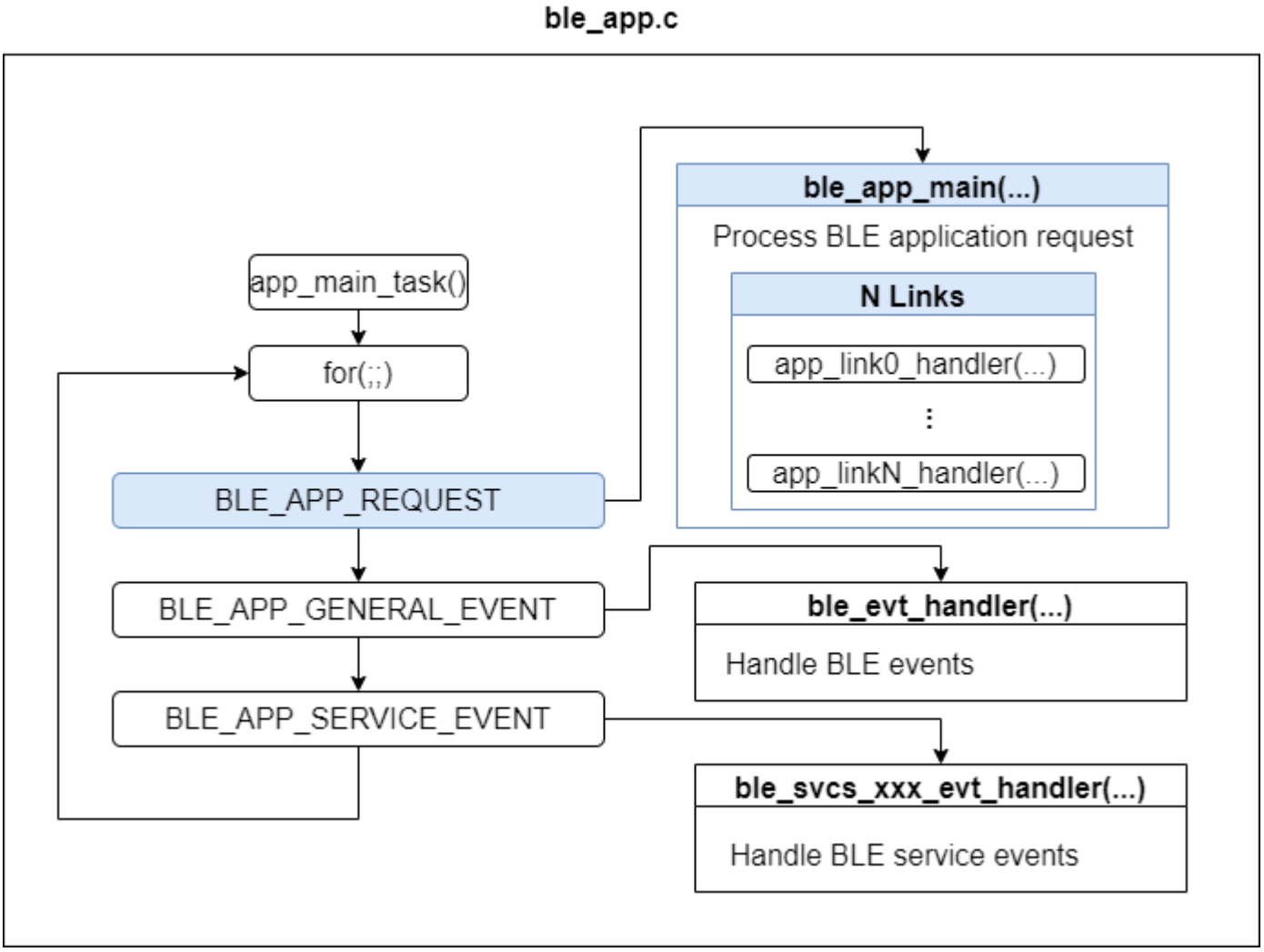


Figure 7. BLE Application Flow

BLE controller and BLE application initialization are implemented in ble_init() function and this initial function will be issued once.

Strong suggest users that the application behavior should write in ble_app_main() function, and issue “app_request_set()” to send the application request to the BLE application task.

If BLE application task receives the event “BLE_APP_REQUEST” will issue “ble_app_main()” function; receives the event “BLE_APP_GENERAL_EVENT” will issue the registered BLE event callback function; receives the event “BLE_APP_SERVICE_EVENT” will issue the registered service event callback function.

2.2 BLE Initialization

RT58x BLE general initialization process is shown as below:

1. Initial BLE controller which includes PHY and Link Layer and basic default settings.
2. Initial BLE services. (Optional)
3. Set application request. (Optional)

(intentionally blank)

```
static ble_err_t ble_init(void)
{
    ble_err_t status = BLE_ERR_OK;

    do
    {
        status = ble_cmd_phy_controller_init();
        if (status != BLE_ERR_OK)
        {
            break;
        }
        status = ble_cmd_device_addr_set((ble_gap_addr_t *)&DEVICE_ADDR);
        if (status != BLE_ERR_OK)
        {
            break;
        }
        status = ble_cmd_suggest_data_len_set(BLE_GATT_DATA_LENGTH_MAX);
        if (status != BLE_ERR_OK)
        {
            break;
        }
        // 2. BLE services init, i.e., register ble_svcs_xxx_evt_handler
        status = server_profile_init(APP_TRSP_P_HOST_ID);
        if (status != BLE_ERR_OK)
        {
            break;
        }
    } while (0);
    return status;
}

static void app_main_task(void)
{
    ble_err_t status;
    app_queue_t p_app_q;

    status = BLE_ERR_OK;
    // 1. BLE init
    status = ble_init();
    if (status != BLE_ERR_OK)
    {
        info_color(LOG_RED, "[DEBUG_ERR] ble_init() fail: %d\n", status);
        while (1);
    }

    // 3. application request, i.e., start advertising
    app_request_set(APP_TRSP_P_HOST_ID, APP_REQUEST_ADV_START, false);

    for (;;)
    {
          ……
    }
}
```

2.3 BLE Main Loop

Implement BLE main loop “app_main_task()” in ble_app.c file which is running the main application and issue “app_request_set()” to send the application request to the BLE application task to control the application flow.

Noted that the following sample codes are the excerpt from TRSP_Periph demo.

- **BLE application request definition.**
  Application request definitions could be customized.

```
typedef enum
{
   APP_REQUEST_IDLE = 0x00,
   APP_REQUEST_ADV_START,
   APP_REQUEST_TRSPS_DATA_SEND,
} app_request_t;
```

- **app_request_set() implementation.**
  User could issue this function with “request” to control the application flow. Please noted that if issue this function in ISR shall set the parameter “from_isr” to be true; otherwise set to false.

```
static bool app_request_set(uint8_t host_id, app_request_t request, bool from_isr)
{
    app_queue_t p_app_q;
    p_app_q.event = 0; // from BLE
    p_app_q.param_type = QUEUE_TYPE_APP_REQ;
    p_app_q.param.app_req.host_id = host_id;
    p_app_q.param.app_req.app_req = request;

    if (from_isr == false)
    {
        if (xQueueSendToBack(g_app_msg_q, &p_app_q, 20) != pdTRUE)
        {
            // send error
            return false;
        }
    }
    else
    {
        BaseType_t context_switch = pdFALSE;
        xQueueSendToBackFromISR(g_app_msg_q, &p_app_q, &context_switch);

        if (context_switch != pdTRUE)
        {
            // send error
            return false;
        }
    }
    return true;
}
```

- **ble_app_main() implementation.**

  This application flow can be implemented for peripheral and central role with “0” to “n” connection links, where “n” is less than or equal to BLE_SUPPORT_NUM_CONN_MAX. The connection link’s information is defined in “project_config.h”, “ble_profile_def.c”, “ble_profile_app.c” and “ble_profile.h”.

  Issued “app_linkx_handler()” function to handle the process flow of each link as below example.

```
static void app_peripheral_handler(app_req_param_t *p_param)
{
    ble_err_t status;
    uint8_t host_id;
    ble profile info t *p profile info;

    host_id = p_param->host_id;
    p_profile_info = (ble_profile_info_t *)ble_app_link_info[host_id].profile_info;

    switch (p_param->app_req)
    {
    case APP REQUEST ADV START:
          …
        break;

    case APP_REQUEST_TRSPS_DATA_SEND:
          …
        break;

    default:
        break;
    }
}

static void ble_app_main(app_req_param_t *p_param)
{
    // Link - Peripheral
    app_peripheral_handler(p_param);
}
```

(intentionally blank)

2.4 BLE Event Handler

BLE event handler to handle BLE related events from BLE stack. The event definitions are defined in “ble_event.h”.

```
static void ble_evt_handler(ble_evt_param_t *p_param)
{
    switch (p_param->event)
    {
    case BLE_ADV_EVT_SET_ENABLE:

         break;

    case BLE_GAP_EVT_CONN_COMPLETE:

         break;

    case BLE_GAP_EVT_CONN_PARAM_UPDATE:
         break;

    case BLE_GAP_EVT_PHY_READ:
    case BLE_GAP_EVT_PHY_UPDATE:
         break;

    case BLE_ATT_GATT_EVT_MTU_EXCHANGE:
         break;

    case BLE_ATT_GATT_EVT_WRITE_SUGGESTED_DEFAULT_DATA_LENGTH:
         break;

    case BLE_ATT_GATT_EVT_DATA_LENGTH_CHANGE:
         break;

    case BLE_GAP_EVT_DISCONN_COMPLETE:
         break;

    default:
        break;
    }
}
```

(intentionally blank)

2.5 BLE Service Event Handle (Optional)

BLE service handler shall be implemented if the application supports any BLE service which has to handle service events. Issue service initialization with callback parameter to register service event handler.

In service handler, user could retrieve the event's GAP role, handle number and data information by the parameter "p_param".

```
static void ble_svcs_trsps_evt_handler(ble_evt_att_param_t *p_param)
{
    if (p param->gatt role == BLE GATT ROLE SERVER)
    {
        /* ----------------- Handle event from client ----------------- */
        switch (p_param->event)
        {
        case BLESERVICE TRSPS UDATRW01 WRITE EVENT:
        case BLESERVICE_TRSPS_UDATRW01_WRITE_WITHOUT_RSP_EVENT:
            break;

        case BLESERVICE_TRSPS_UDATRW01_READ_EVENT:
             break;

        default:
            break;
        }
    }
}

static ble err t server profile init(uint8 t host id)
{
    ble err t status = BLE ERR OK;
    ble_profile_info_t *p_profile_info = (ble_profile_info_t
*)ble_app_link_info[host_id].profile_info;

    // set link's state
    ble_app_link_info[host_id].state = STATE_STANDBY;

    do
    {
       …
        // TRSPS Related
        // -------------------------------------
        status = ble_svcs_trsps_init(host_id, BLE_GATT_ROLE_SERVER, &(p_profile_info-
>svcs_info_trsps), ble_svcs_trsps_evt_handler);
        if (status != BLE_ERR_OK)
        {
            break;
        }
    } while (0);

    return status;
}
```

3. BLE Profile Implementation

There are three steps for user to implement BLE profiles in application.

Step 1: Implement BLE services. (ble_service_xxx.c/ (ble_service_xxx.h)

Step 2: Implement BLE profile definition. (ble_profile_def.c)

Step 3: Implement user data for BLE profile definitions. (ble_profile_app.c/ ble_profile.h)

For convenience and ease of use, Rafael provides the tool "***Service_Profile_CreateTool***" for user to develop a customized service and profile.

3.1 Step 1 & 2: Implemented Services and Profiles.

The instruction of create BLE service and profile please refer to the document "*RT58x_BLE_SDK_Service Profile Guide.pdf*" for more details.

3.2 Step 3: Implemented user data for BLE profiles.

Implemented profile related definitions for application in "ble_profile_app.c/ ble_profile.h" files. Defined the minimum number of connection link for each service, define data structure for each link and implement "svcs_handles_get()" function if BLE GATT role is set to "BLE_GATT_ROLE_CLIENT".

3.2.1.1 Step 3.1: Max. number of connection link for each service.

TRSP_1C1P demo (TRSP Central + TRSP Peripheral demo)

	GAP	GATT	DIS	TRSPS
TRSP Client	V	V	V	V
TRSP Server	V	V	V	V
Min. # of service link	2	2	2	2

Defined in "ble_profile.h".

```
/** Define the maximum number of BLE GAPS link. */
#define MAX_NUM_CONN_GAPS      2

/** Define the maximum number of BLE GATTS link. */
#define MAX_NUM_CONN_GATTS     2

/** Define the maximum number of BLE DIS link. */
#define MAX_NUM_CONN_DIS       2

/** Define the maximum number of BLE TRSPS link. */
#define MAX_NUM_CONN_TRSPS     2
```

3.2.1.2 Step 3.2: Define user data and data structure for each link in “ble_profile.h”.

Define the user data, related profile information and extern necessary variables definitions.

TRSP_1C1P demo (TRSP Central + TRSP Peripheral demo)

```
/** BLE Application Profile Attribute Information Structure.*/
typedef struct
{
    ble_svcs_gaps_info_t      svcs_info_gaps;      /**< GAPS information  (server/cli-
ent). */
    ble_svcs_gatts_info_t     svcs_info_gatts;     /**< GATTS information (server/cli-
ent). */
    ble_svcs_dis_info_t       svcs_info_dis;       /**< DIS information   (server/cli-
ent). */
    ble_svcs_trsps_info_t     svcs_info_trsps;     /**< TRSPS information (server/cli-
ent). */
} ble_profile_info_t;

typedef struct
{
    ble_gap_role_t   gap_role;    /**< GAP role. */
    uint8_t          state;       /**< Current state. */
    void             *profile_info;  /**< Link profile information. */
} ble_app_link_info_t;

/** Get BLE (Central) Service All Handles*/
ble_err_t svcs_handles_get(uint8_t host_id);

/** Extern maximum Number of Host Connection Link Definition. */
extern const uint8_t                              max_num_conn_host;

/** Extern BLE Connection Links Definition. */
extern const ble_att_role_by_id_t                 att_db_link[];

/** Extern BLE Connection Link Parameter Table Definition. */
extern const ble_att_db_mapping_by_id_t           att_db_mapping[];

/** Extern BLE Connection Link Mapping Size Definition. */
extern const ble_att_db_mapping_by_id_size_t      att_db_mapping_size[];

/* Extern BLE application links information which is ordered by host id. */
extern ble_app_link_info_t                        ble_app_link_info[];
```

(intentionally blank)

3.2.1.3 Step 3.3: Implement profile related function.

User shall implement customized definition or function for BLE profile in this file. Here suggest implementing “svcs_handles_get()” function to get all service attribute handles after received “BLE_ATT_GATT_EVT_DB_PARSE_COMPLETE” event if BLE GATT role is set to “BLE_GATT_ROLE_CLIENT”.

TRSP_1C1P demo (TRSP Central + TRSP Peripheral demo)

```
/** Get BLE (Central) Service All Handles */
ble err t svcs handles get(uint8 t host id)
{
    ble err t status;
    ble_profile_info_t *p_profile_info = (ble_profile_info_t
*)ble_app_link_info[host_id].profile_info;

    status = BLE_ERR_OK;
    do
    {
        // Get GAPS handles
        status = ble svcs gaps handles get(host id, BLE GATT ROLE CLIENT, (void
*)&p_profile_info->svcs_info_gaps);
        if (status != BLE_ERR_OK)
        {
            break;
        }

        // Get GATTS handles
        status = ble svcs gatts handles get(host id, BLE GATT ROLE CLIENT, (void
*)&p_profile_info->svcs_info_gatts);
        if (status != BLE ERR OK)
        {
            break;
        }

        // Get DIS handles
        status = ble_svcs_dis_handles_get(host_id, BLE_GATT_ROLE_CLIENT, (void
*)&p_profile_info->svcs_info_dis);
        if (status != BLE ERR OK)
        {
            break;
        }

        // Get TRSPS handles
        status = ble_svcs_trsps_handles_get(host_id, BLE_GATT_ROLE_CLIENT, (void
*)&p_profile_info->svcs_info_trsps);
        if (status != BLE ERR OK)
        {
            break;
        }
    } while (0);

    return status;
}
```

4. BLE SDK APIs

Rafael RT58x BLE SDK APIs are listed in below.

4.1 BLE Common APIs

Define BLE common functions.

4.1.1 ble_cmd_phy_controller_init()

```
ble_err_t
ble_cmd_phy_controller_init(void)
```

BLE set PHY controller initialization function.

Returns

BLE_ERR_OK is success or an error code.

4.1.2 ble_cmd_read_unique_code()

```
ble_err_t
ble_cmd_read_unique_code(ble_unique_code_format_t *p_param)
```

BLE read unique code function.

Parameters

in	p_param	a pointer to read BLE unique code buffer.

Returns

BLE_ERR_OK is success or an error code.

Note: This unique code must be burned into the OTP via MP_Tool and read by this API. And the BLE mac address must conform to the officially defined public address or random address format.

4.1.3 ble_cmd_read_filter_accept_list_size()

```
ble_err_t
ble_cmd_read_filter_accept_list_size(void)
```

BLE read filter accept list size function.

BLE Event

Wait for BLE_COMMON_EVT_READ_FILTER_ACCEPT_LIST_SIZE event which indicates the filter accept list size.

Returns

BLE_ERR_OK is success or an error code.

4.1.4 ble_cmd_clear_filter_accept_list()

```
ble_err_t
ble_cmd_clear_filter_accept_list (void)
```

BLE clear filter accept list function.

BLE Event

Wait for BLE_COMMON_EVT_CLEAR_FILTER_ACCEPT_LIST event which indicates the status of the clear filter accept list.

Returns

BLE_ERR_OK is success or an error code.

4.1.5 ble_cmd_add_device_to_filter_accept_list()

```
ble_err_t
ble_cmd_add_device_to_filter_accept_list(ble_filter_accept_list t *p_accept_list)
```

BLE Add device to filter accept list function.

BLE Event

Wait for BLE_COMMON_EVT_ADD_FILTER_ACCEPT_LIST event which indicates the status of adding device to the filter accept list.

Parameters

in	p_filter_list	a pointer to the filter accept list data.

Returns

BLE_ERR_OK is success or an error code.

4.1.6 ble_cmd_remove_device_from_filter_accept_list()

```
ble_err_t
ble_cmd_remove_device_from_filter_accept_list(ble_filter_accept_list t *p_accept_list)
```

BLE Add device to filter accept list function.

BLE Event

Wait for BLE_COMMON_EVT_REMOVE_FILTER_ACCEPT_LIST event which indicates the status of the device to be removed from the filter accept list.

Parameters

in	p_filter_list	a pointer to the filter accept list data.

Returns

BLE_ERR_OK is success or an error code.

4.1.7 ble_cmd_antenna_info_read()

```
ble_err_t
ble_cmd_antenna_info_read(void)
```

BLE read the antenna information function.

BLE Event

Wait for BLE_COMMON_EVT_READ_ANTENNA_INFO event which indicates the information of the antenna.

Returns

BLE_ERR_OK is success or an error code.

(intentionally blank)

4.2 BLE Advertising APIs

Define BLE advertising related functions.

4.2.1 ble_cmd_adv_data_set()

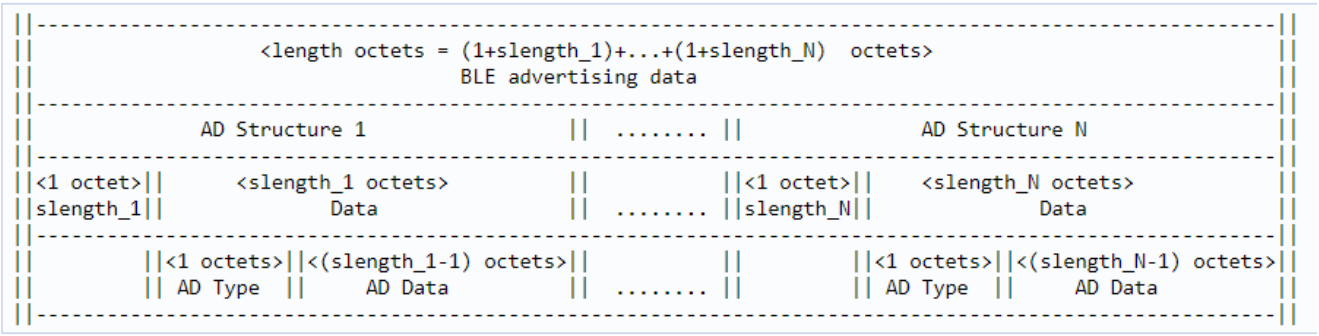
```
ble_err_t
ble_cmd_adv_data_set(ble_adv_data_param_t *p_param)
```

Set BLE advertising data.

Note

BLE advertising data length shall be less than or equal to 31 bytes.

AD Type please refer to ble_adv_type_t



Parameters

in	p_param	a pointer to advertising data parameter structure.

Returns

BLE_ERR_OK is success or an error.

(intentionally blank)

4.2.2 ble_cmd_adv_scan_rsp_set()

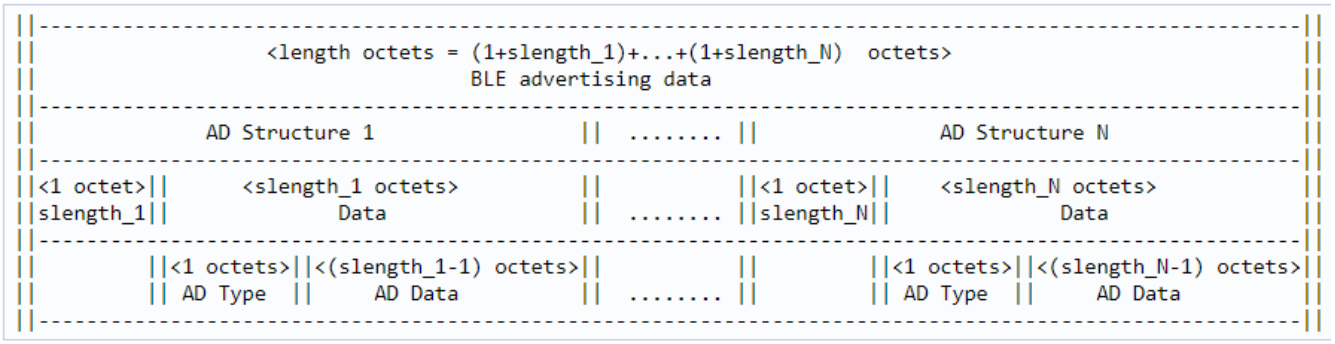
```
ble_err_t
ble_cmd_adv_scan_rsp_set(ble_adv_data_param_t *p_param)
```

Set BLE scan response data.

Note

BLE scan response data length shall be less than or equal to 31 bytes.

AD Type please refer to ble_adv_type_t



Parameters

in	p_param	a pointer to scan response data parameter structure.

Returns

BLE_ERR_OK is success or an error code.

4.2.3 ble_cmd_adv_param_set()

```
ble_err_t
ble_cmd_adv_param_set(ble_adv_param_t *p_param)
```

Set BLE advertising parameter.

Note

Advertising interval Min. and Max.: ADV_INTERVAL_MIN to ADV_INTERVAL_MAX

Advertising interval Min. shall be less than or equal to advertising interval Max.

Parameters

in	*p_param*	a pointer to advertising parameter structure.

Returns

BLE_ERR_OK is success or an error code.

4.2.4 ble_cmd_adv_enable()

```
ble_err_t
ble_cmd_adv_enable(uint8_t host_id)
```

Set BLE start advertising.

BLE Event

Wait for BLE_ADV_EVT_SET_ENABLE event which indicates the advertisement has been started.

Parameters

in	*host_id*	the link's host ID or set to BLE_HOSTID_RESERVED to enable ADV_TYPE_ADV_NONCONN_IND or ADV_TYPE_SCAN_IND advertisement.

Returns

BLE_ERR_OK is success or an error code.

4.2.5 ble_cmd_adv_disable()

```
ble_err_t
ble_cmd_adv_enable(void)
```

Stop BLE advertising.

BLE Event

Wait for BLE_ADV_EVT_SET_ENABLE event which indicates the advertisement has been stoped.

Returns

BLE_ERR_OK is success or an error code.

(intentionally blank)

4.3 BLE Scan APIs

Define BLE scan related definition functions.

4.3.1 ble_cmd_scan_param_set()

```
ble_err_t
ble_cmd_scan_param_set(ble_scan_param_t *p_param)
```

Set BLE scan parameter.

Note

Scan window can only be less than or equal to the scan interval.

Parameters

in	p_param	a pointer to scanning parameter structure.

Returns

BLE_ERR_OK is success or an error code.

4.3.2 ble_cmd_scan_enable()

```
ble_err_t
ble_cmd_scan_enable(void)
```

Set BLE start Scanning.

BLE Event

Wait for BLE_SCAN_EVT_SET_ENABLE event which indicates the scan has been started.

Wait for BLE_SCAN_EVT_ADV_REPORT event to receive scanned devices information.

Returns

BLE_ERR_OK is success or an error code.

Note: When using the SCAN function for multiple connections, ensure that the current connection interval is greater than 10ms*number of links.

(intentionally blank)

4.3.3 ble_cmd_scan_disable()

```
ble_err_t
ble_cmd_scan_disable(void)
```

Set BLE stop scanning.

BLE Event

Wait for BLE_SCAN_EVT_SET_ENABLE event which indicates the scan has been stopped.

Returns

BLE_ERR_OK is success or an error code.

4.3.4 ble_cmd_scan_report_adv_data_parsing()

```
ble_err_t
ble_cmd_scan_report_adv_data_parsing(ble_evt_scan_adv_report_t *p_rpt_data,
        ble_gap_ad_type_t          ad_type,
        uint8_t                    *p_data,
        uint8_t                    *p_data_length)
```

Get BLE advertising parsing data by advertising data type.

Parameters

in	*p_rpt_data*	a pointer to scanned advertising data to parse.
in	*ad_type*	parsing advertising type.
out	*p_data*	a pointer to the parsing data.
out	*p_data_length*	a pointer to the length of parsing data.

Returns

BLE_ERR_OK is success or an error code.

(intentionally blank)

4.4 BLE GAP APIs

Define BLE GAP related definition functions.

4.4.1 ble_cmd_device_addr_get()

```
ble_err_t
ble_cmd_device_addr_get(ble_gap_addr_t *p_addr)
```

Get BLE device address and device address type.

This function is used to get BLE Local Device Address and Device Address Type.

Note

BLE Address: Little Endian and the length is BLE_ADDR_LEN

If Device BLE Address is set to "01:02:03:04:05:06", addrParam->addr[0] = 0x06

Parameters

out	*p_addr*	a pointer to the device BLE address and BLE address type.

Returns

BLE_ERR_OK is success or an error code.

4.4.2 ble_cmd_device_addr_set()

```
ble_err_t ble_cmd_device_addr_set(ble_gap_addr_t *p_addr)
```

Set BLE device address and device address type.

This function is used to set BLE Local Device Address and Device Address Type.

Note

BLE Address: Little Endian and the length is BLE_ADDR_LEN

If Device BLE Address is set to "01:02:03:04:05:06", addrParam->addr[0] = 0x06

Parameters

in	*p_addr*	a pointer to the device BLE address and BLE address type.

Returns

BLE_ERR_OK is success or an error code.

4.4.3 ble_cmd_conn_create()

```
ble_err_t
ble_cmd_conn_create(ble_gap_create_conn_param_t *p_param)
```

BLE connection create command.

BLE Event

Wait for BLE_GAP_EVT_CONN_COMPLETE event which indicates the link connected.

Parameters

in	*p_param*	a pointer to the create connection parameter.

Returns

BLE_ERR_OK is success or an error code.

4.4.4 ble_cmd_conn_create_cancel()

```
ble_err_t
ble_cmd_conn_create_cancel(void)
```

BLE cancel create connection process command.

BLE Event

Wait for BLE_GAP_EVT_CONN_CANCEL event which indicates the create connection canceled.

Returns

BLE_ERR_OK is success or an error code.

4.4.5 ble_cmd_conn_param_update()

```
ble_err_t
ble_cmd_conn_param_update(ble_gap_conn_param_update_param_t *p_param)
```

Set BLE connection parameter update.

BLE Event

Wait for BLE_GAP_EVT_CONN_PARAM_UPDATE event which indicates the link connection parameter updated.

Parameters

in	*p_param*	a pointer to the connection parameter update.

Returns

BLE_ERR_OK is success or an error code.

***Note: When using the SCAN function for multiple connections, ensure that the current connection interval is greater than 10ms*number of links.**

4.4.6 ble_cmd_conn_terminate()

```
ble_err_t
ble_cmd_conn_terminate(uint8_t host_id)
```

BLE terminate the connection link.

BLE Event

Wait for BLE_GAP_EVT_DISCONN_COMPLETE event which indicates the link disconnected.

Parameters

in	*host_id*	the link's host ID.

Returns

BLE_ERR_OK is success or an error code.

4.4.7 ble_cmd_phy_update()

```
ble_err_t
ble_cmd_phy_update(ble_gap_phy_update_param_t *p_param)
```

BLE set PHY update command.

BLE Event

Wait for BLE_GAP_EVT_PHY_UPDATE event which indicates the link PHY updated.

Parameters

in	p_param	A pointer to the phy update parameter.

Returns

BLE_ERR_OK is success or an error code.

4.4.8 ble_cmd_phy_read()

```
ble_err_t
ble_cmd_phy_read(uint8_t host_id)
```

BLE read RF PHY command.

BLE Event

Wait for BLE_GAP_EVT_PHY_READ event which indicates the BLE link phy information.

Parameters

in	host_id	the link's host ID.

Returns

BLE_ERR_OK is success or an error code.

4.4.9 ble_cmd_rssi_read()

```
ble_err_t
ble_cmd_rssi_read(uint8_t host_id)
```

BLE read RSSI command.

BLE Event

Wait for BLE_GAP_EVT_RSSI_READ event which indicates the link RSSI value.

Parameters

in	host_id	the link's host ID.

Returns

BLE_ERR_OK is success or an error code.

4.4.10 ble_cmd_host_ch_classif_set()

```
ble_err_t
ble_cmd_host_ch_classif_set(ble_gap_host_ch_classif_t *p_param)
```

BLE set host channel classification command.

Parameters

in	p_param	a pointer to the host channel classification parameter.

Returns

BLE_ERR_OK is success or an error code.

4.4.11 ble_cmd_channel_map_read()

```
ble_err_t
ble_cmd_channel_map_read(uint8_t host_id)
```

BLE read channel map command.

BLE Event

Wait for BLE_GAP_EVT_READ_CHANNEL_MAP event which indicates the channel map value.

Parameters

in	host_id	the link's host ID.

Returns

BLE_ERR_OK is success or an error code.

4.4.12 ble_cmd_resolvable_address_init()

```
ble_err_t
ble_cmd_resolvable_address_init(void)
```

BLE Init resolvable address.

Returns

BLE_ERR_OK is success or an error code.

4.4.13 ble_cmd_regenerate_resolvable_address()

```
ble_err_t
ble_cmd_regenerate_resolvable_address(uint8_t host_id, uint8_t en_new_irk)
```

BLE regenerate resolvable address.

Parameters

in	host_id	the link's host ID.
in	En_new_irk	whether to generate a new IRK.

Returns

BLE_ERR_OK is success or an error code.

(intentionally blank)

4.5 BLE ATT and GATT APIs

Define BLE ATT and GATT related definition functions.

4.5.1 ble_cmd_suggest_data_len_set()

```
ble_err_t
ble_cmd_suggest_data_len_set(uint16_t tx_octets)
```

Set preferred data length.

Note

The range of “tx_octets” is BLE_GATT_DATA_LENGTH_MIN to BLE_GATT_DATA_LENGTH_MAX bytes.

Parameters

in	*tx_octets*	preferred maximum number of payload octets.

Returns

BLE_ERR_OK is success or an error code.

4.5.2 ble_cmd_default_mtu_size_set()

```
ble_err_t
ble_cmd_default_mtu_size_set(uint8_t host_id, uint16_t mtu)
```

Set preferred MTU size.

Note

The range of mtu is BLE_GATT_ATT_MTU_MIN to BLE_GATT_ATT_MTU_MAX bytes.

Parameters

in	*host_id*	the link's host ID.
in	*mtu*	preferred MTU size.

Returns

BLE_ERR_OK is success or an error code.

4.5.3 ble_cmd_mtu_size_update()

```
ble_err_t
ble_cmd_mtu_size_update(uint8_t host_id, uint16_t mtu)
```

ATT_MTU exchange request.

BLE Event

Wait for BLE_ATT_GATT_EVT_MTU_EXCHANGE event to get exchanged MTU size.

Attention

Only supported if GATT role is BLE_GATT_ROLE_CLIENT.

Note

The range of mtu is BLE_GATT_ATT_MTU_MIN to BLE_GATT_ATT_MTU_MAX bytes.

Parameters

in	*host_id*	the link's host ID.
in	*mtu*	preferred MTU size.

Returns

BLE_ERR_OK is success or an error code.

4.5.4 ble_cmd_data_len_update()

```
ble_err_t
ble_cmd_data_len_update(uint8_t host_id, uint16_t tx_octets)
```

Set data length update.

BLE Event

Wait for BLE_ATT_GATT_EVT_DATA_LENGTH_CHANGE event to get updated BLE data length.

Note

The range of “tx_octets” is BLE_GATT_DATA_LENGTH_MIN to BLE_GATT_DATA_LENGTH_MAX bytes.

Parameters

in	*host_id*	the link's host ID.
in	*tx_octets*	preferred maximum number of payload octets.

Returns

BLE_ERR_OK is success or an error code.

4.5.5 ble_cmd_mtu_size_get()

```
ble_err_t
ble_cmd_mtu_size_get(uint8_t host_id, uint16_t *mtu)
```

Get BLE GATT MTU Size.

Parameters

in	*host_id*	the link's host ID.
out	*mtu*	the link's MTU size.

Returns

BLE_ERR_OK is success or an error code.

4.5.6 ble_cmd_gatt_read_rsp()

```
ble_err_t
ble_cmd_gatt_read_rsp(ble_gatt_data_param_t *p_param)
```

BLE GATT Read Response.

Attention

Only supported if GATT role is BLE_GATT_ROLE_SERVER.

Note

The read response is sent in reply to a received Read Request and contains the value of the attribute that has been read.

Parameters

in	p_param	A pointer to read response parameter.

Returns

BLE_ERR_OK is success or an error code.

(intentionally blank)

4.5.7 ble_cmd_gatt_read_by_type_rsp()

```
ble_err_t
ble_cmd_gatt_read_by_type_rsp(ble_gatt_data_param_t *p_param)
```

BLE GATT Read By Type Response.

Attention

Only supported if GATT role is BLE_GATT_ROLE_SERVER.

Note

The read by type response is sent in reply to a received Read By Type Request and contains the handle number and value of the attribute that has been read.

Parameters

in	*p_param*	A pointer to read by type response parameter.

Returns

BLE_ERR_OK is success or an error.

4.5.8 ble_cmd_gatt_read_blob_rsp()

```
ble_err_t
ble_cmd_gatt_read_blob_rsp(ble_gatt_data_param_t *p_param)
```

BLE GATT Read Blob Response.

Attention

Only supported if GATT role is BLE_GATT_ROLE_SERVER.

Note

The read blob response is sent in reply to a received Read Blob Request and contains the value of the attribute that has been read.

Parameters

in	*p_param*	A pointer to the read blob response parameter.

Returns

BLE_ERR_OK is success or an error code.

4.5.9 ble_cmd_gatt_error_rsp()

```
ble_err_t
ble_cmd_gatt_error_rsp(ble_gatt_err_rsp_param_t *p_param)
```

BLE GATT Error Response.

Attention

Only supported if GATT role is BLE_GATT_ROLE_SERVER.

Note

The Error Response is used to state that a given request cannot be performed, and to provide the reason.

Parameters

in	*p_param*	A pointer to the error response parameter.

Returns

BLE_ERR_OK is success or an error code.

4.5.10 ble_cmd_gatt_notification()

```
ble_err_t
ble_cmd_gatt_notification(ble_gatt_data_param_t *p_param)
```

BLE GATT Notification.

Attention

Only supported if GATT role is BLE_GATT_ROLE_SERVER.

Note

When a server is configured to notify a Characteristic Value to a client without the acknowledgment that the notification was successfully received.

Parameters

in	*p_param*	A pointer to the notification parameter.

Returns

BLE_ERR_OK is success or an error code.

4.5.11 ble_cmd_gatt_indication()

```
ble_err_t
ble_cmd_gatt_indication(ble_gatt_data_param_t *p_param)
```

BLE GATT Indication.

Attention

Only supported if GATT role is BLE_GATT_ROLE_SERVER.

Note

When a server is configured to indicate a Characteristic Value to a client and expects the indication was successfully received.

Parameters

in	*p_param*	A pointer to the indication parameter.

Returns

BLE_ERR_OK is success or an error code.

4.5.12 ble_cmd_gatt_write_req()

```
ble_err_t
ble_cmd_gatt_write_req(ble_gatt_data_param_t *p_param)
```

BLE GATT Write Request.

Attention

Only supported if GATT role is BLE_GATT_ROLE_CLIENT.

Note

The Write Request is used to request the server to write the value of an attribute and acknowledge that this has been achieved in a Write Response.

A Write Response shall be sent by the server if the write of the Characteristic Value succeeded.

Parameters

in	*p_param*	A pointer to the write request parameter.

Returns

BLE_ERR_OK is success or an error code.

4.5.13 ble_cmd_gatt_write_cmd()

```
ble_err_t
ble_cmd_gatt_write_cmd(ble_gatt_data_param_t *p_param)
```

BLE GATT Write Command.

Attention

Only supported if GATT role is BLE_GATT_ROLE_CLIENT.

Note

Write a Characteristic Value to a server when the client knows the Characteristic Value Handle and the client does not need an acknowledgment that the write was successfully performed.

Parameters

in	*p_param*	A pointer to the write command parameter.

Returns

BLE_ERR_OK is success or an error code.

4.5.14 ble_cmd_gatt_read_req()

```
ble_err_t
ble_cmd_gatt_read_req(ble_gatt_read_req_param_t *p_param)
```

BLE GATT Read Request.

Attention

Only supported if GATT role is BLE_GATT_ROLE_CLIENT.

Note

Read a Characteristic Value from a server when the client knows the Characteristic Value Handle.

Parameters

in	*p_param*	A pointer to the read request parameter.

Returns

BLE_ERR_OK is success or an error code.

4.5.15 ble_cmd_gatt_read_blob_req()

```
ble_err_t
ble_cmd_gatt_read_blob_req(ble_gatt_read_blob_req_param_t *p_param)
```

BLE GATT Read Blob Request.

Attention

Only supported if GATT role is BLE_GATT_ROLE_CLIENT.

Note

Read part of the value of an attribute at a given offset from a server when the client knows the Characteristic Value Handle.

Parameters

in	*p_param*	A pointer to the read blob request parameter.

Returns

BLE_ERR_OK is success or an error code.

(intentionally blank)

4.6 BLE Security APIs

Define BLE security related definition functions.

4.6.1 ble_cmd_security_request_set()

```
ble_err_t
ble_cmd_security_request_set(uint8_t host_id)
```

BLE send security request.

Parameters

id	*host_id*	the link's host ID.

Returns

BLE_ERR_OK is success or an error code.

4.6.2 ble_cmd_passkey_set()

```
ble_err_t
ble_cmd_passkey_set(ble_sm_passkey_param_t *p_param)
```

Set BLE pairing passkey value.

Parameters

in	*p_param*	A pointer to the passkey parameter.

Returns

BLE_ERR_OK is success or an error code.

4.6.3 ble_cmd_io_capability_set()

```
ble_err_t
ble_cmd_io_capability_set(ble_sm_io_cap_param_t *p_param)
```

Set BLE IO capabilities.

Parameters

in	*p_param*	a pointer to the IO capability parameter.

Returns

BLE_ERR_OK is success or an error.

4.6.4 ble_cmd_bonding_flag_set()

```
ble_err_t
ble_cmd_bonding_flag_set(ble_sm_bonding_flag_param_t *p_param)
```

Set BLE bonding flags.

Parameters

in	*p_param*	a pointer to the bonding parameter.

Returns

BLE_ERR_OK is success or an error code.

4.6.5 ble_cmd_cccd_restore()

```
ble_err_t
ble_cmd_cccd_restore(uint8_t host_id)
```

BLE get the restore CCCD.

Parameters

in	*host_id*	The link's id.

Returns

BLE_ERR_OK is success or an error code.

4.6.6 ble_cmd_bonding_space_init()

```
ble_err_t
ble_cmd_bonding_space_init(void)
```

BLE initialize the bonding space.

Returns

BLE_ERR_OK is success or an error code.

4.6.7 ble_cmd_write_identity_resolving_key()

```
ble_err_t
ble_cmd_write_identity_resolving_key(ble_sm_irk_param_t *p_param)
```

BLE write identity resolving key.

Parameters

in	*p_param*	a pointer to the identity resolving key parameter.

Returns

BLE_ERR_OK is success or an error code.

4.7 BLE Privacy APIs

Define BLE privacy related definition functions.

4.7.1 ble_cmd_privacy_enable()

```
ble_err_t
ble_cmd_privacy_enable(ble_set_privacy_cfg_t *p_param)
```

BLE privacy enable function.

Parameters

in	p_param	a pointer to the privacy configuration value

Returns

BLE_ERR_OK is success or an error code.

4.7.2 ble_cmd_privacy_disable()

```
ble_err_t
ble_cmd_privacy_disable(void)
```

BLE privacy disable function.

Returns

BLE_ERR_OK is success or an error code.

(intentionally blank)

4.8 BLE Connect CTE APIs

Define BLE connect CTE related definition functions.

4.8.1 ble_cmd_connection_cte_receive_parameters_set()

```
ble_err_t
ble_cmd_connection_cte_receive_parameters_set(ble_connection_cte_rx_param_t
*p_param)
```

BLE Set connection CTE receiver parameters.

BLE Event

Wait for BLE_CTE_EVT_SET_CONN_CTE_RX_PARAM event which indicates the command status.

Parameters

in	p_param	a pointer of the CTE receiver parameters structure.

Returns

BLE_ERR_OK is success or an error code.

4.8.2 ble_cmd_connection_cte_transmit_parameters_set()

```
ble_err_t
ble_cmd_connection_cte_transmit_parameters_set(ble_connection_cte_tx_param_t
*p_param)
```

BLE Set connection CTE transmit parameters.

BLE Event

Wait for BLE_CTE_EVT_SET_CONN_CTE_TX_PARAM event which indicates the command status.

Parameters

in	p_param	a pointer of the CTE transmit parameters structure.

Returns

BLE_ERR_OK is success or an error code.

4.8.3 ble_cmd_connection_cte_request_enable()

```
ble_err_t
ble_cmd_connection_cte_request_enable(ble_connection_cte_req_enable_t *p_param)
```

BLE Set connection CTE request enable.

BLE Event

Wait for BLE_CTE_EVT_SET_CONN_CTE_REQ event which indicates the command status.

Parameters

in	p_param	a pointer of the CTE request enable structure.

Returns

BLE_ERR_OK is success or an error code.

4.8.4 ble_cmd_connection_cte_response_enable()

```
ble_err_t
ble_cmd_connection_cte_response_enable(ble_connection_cte_rsp_enable_t *p_param)
```

BLE Set connection CTE response enable.

BLE Event

Wait for BLE_CTE_EVT_SET_CONN_CTE_RSP event which indicates the command status.

Parameters

in	p_param	a pointer of the CTE transmit parameters structure.

Returns

BLE_ERR_OK is success or an error code.

(intentionally blank)

5. Revision History

Revision	Description	Owner	Date
1.0	Initial version.	Yuwei	2022/02/15
1.1	Add more description in section 4.4.5	Yuwei	2022/03/28
1.2	Add filter accept list api	Yuwei	2022/06/06
1.3	Add privacy & connect CTE api	Yuwei	2022/07/19