OPL1000_WIFI_BLE_API_GUIDE

# 1.0.1.20

Generated by Doxygen 1.8.14

# Contents

# Chapter 1

# SDK PREVIEW

- BLE APIs :
  - GAP APIs : BLE GAP APIs
  - GATT APIs : BLE GATT APIs
  - CM APIs : BLE CM APIs
  - MSG APIs : BLE MSG APIs
  - SMP APIs : BLE SMP APIs
- WiFi APIs :
  - Station APIs : STATION APIs
  - Common APIs : COMMON APIs
  - Enumerations : ENUMERATIONS

# Chapter 2

# Module Index

## 2.1 Modules

Here is a list of all modules:

# Chapter 3

# Data Structure Index

## 3.1 Data Structures

Here are the data structures with brief descriptions:

# Chapter 4

# Module Documentation

## 4.1 BLE ALL APIs

BLE ALL APIs.

### Modules

- BLE CM APIs
- BLE GAP APIs
- BLE GATT APIs
- BLE MSG APIs
- BLE SMP APIs

### 4.1.1 Detailed Description

BLE ALL APIs.

## 4.2 BLE CM APIs

### Data Structures

- struct LE_CM_CONNECTION_COMPLETE_IND_T
- struct LE_CM_MSG_ADVERTISE_REPORT_IND_T
- struct LE_CM_MSG_CONN_PARA_REQ_T
- struct LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T
- struct LE_CM_MSG_DATA_LEN_CHANGE_IND_T
- struct LE_CM_MSG_DIRECT_ADV_REPORT_IND_T
- struct LE_CM_MSG_DISCONNECT_COMPLETE_IND_T
- struct LE_CM_MSG_ENCRYPTION_CHANGE_IND_T
- struct LE_CM_MSG_ENCRYPTION_REFRESH_IND_T
- struct LE_CM_MSG_INIT_COMPLETE_CFM_T
- struct LE_CM_MSG_LTK_REQ_IND_T
- struct LE_CM_MSG_READ_ADV_TX_POWER_CFM_T
- struct LE_CM_MSG_READ_BD_ADDR_CFM_T
- struct LE_CM_MSG_READ_CHANNEL_MAP_CFM_T
- struct LE_CM_MSG_READ_RESOLVING_LIST_SIZE_CFM_T
- struct LE_CM_MSG_READ_RSSI_CFM_T
- struct LE_CM_MSG_READ_TX_POWER_CFM_T
- struct LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM_T
- struct LE_CM_MSG_SET_DATA_LENGTH_CFM_T
- struct LE_CM_MSG_SET_DISCONNECT_CFM_T
- struct LE_CM_MSG_SIGNAL_UPDATE_REQ_T
- struct LE_CM_REQ_STATUS_T

### Typedefs

- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_ADD_TO_RESOLVING_LIST_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_ADD_TO_WHITE_LIST_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_CANCEL_CONNECTION_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_CLEAR_RESOLVING_LIST_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_CLEAR_WHITE_LIST_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_CREATE_CONNECTION_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_ENTER_ADVERTISING_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_ENTER_SCANNING_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_EXIT_ADVERTISING_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_EXIT_SCANNING_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_REMOVE_FROM_RESOLVING_LIST_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_REMOVE_FROM_WHITE_LIST_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_ADVERTISING_DATA_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_ADVERTISING_PARAMS_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_CHANNEL_MAP_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_RANDOM_ADDRESS_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_RPA_TIMEOUT_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_SCAN_PARAMS_CFM_T
- typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_SCAN_RSP_DATA_CFM_T

### Enumerations

- enum {
  LE_CM_MSG_INIT_COMPLETE_CFM = LE_CM_MSG_BASE, LE_CM_MSG_SET_DISCONNECT_CFM,
  LE_CM_MSG_DISCONNECT_COMPLETE_IND, LE_CM_MSG_SET_ADVERTISING_DATA_CFM,
  LE_CM_MSG_SET_SCAN_RSP_DATA_CFM, LE_CM_MSG_SET_ADVERTISING_PARAMS_CFM,
  LE_CM_MSG_ENTER_ADVERTISING_CFM, LE_CM_MSG_EXIT_ADVERTISING_CFM,
  LE_CM_MSG_SET_SCAN_PARAMS_CFM, LE_CM_MSG_ENTER_SCANNING_CFM,
  LE_CM_MSG_EXIT_SCANNING_CFM, LE_CM_MSG_CREATE_CONNECTION_CFM,
  LE_CM_MSG_CANCEL_CONNECTION_CFM, LE_CM_MSG_READ_TX_POWER_CFM,
  LE_CM_MSG_READ_BD_ADDR_CFM, LE_CM_MSG_READ_RSSI_CFM,
  LE_CM_MSG_SET_RANDOM_ADDRESS_CFM, LE_CM_MSG_READ_ADV_TX_POWER_CFM,
  LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM LE_CM_MSG_CLEAR_WHITE_LIST_CFM,
  LE_CM_MSG_ADD_TO_WHITE_LIST_CFM, LE_CM_MSG_REMOVE_FROM_WHITE_LIST_CFM,
  LE_CM_MSG_SET_CHANNEL_MAP_CFM, LE_CM_MSG_READ_CHANNEL_MAP_CFM,
  LE_CM_MSG_SET_DATA_LENGTH_CFM, LE_CM_MSG_DATA_LEN_CHANGE_IND,
  LE_CM_MSG_ADD_TO_RESOLVING_LIST_CFM LE_CM_MSG_REMOVE_FROM_RESOLVING_LIST_CFM,
  LE_CM_MSG_CLEAR_RESOLVING_LIST_CFM, LE_CM_MSG_READ_RESOLVING_LIST_SIZE_CFM,
  LE_CM_MSG_SET_RPA_TIMEOUT_CFM, LE_CM_MSG_SIGNAL_UPDATE_REQ,
  LE_CM_MSG_CONN_UPDATE_COMPLETE_IND, LE_CM_MSG_CONN_PARA_REQ,
  LE_CM_MSG_ENCRYPTION_CHANGE_IND LE_CM_MSG_ENCRYPTION_REFRESH_IND,
  LE_CM_MSG_LTK_REQ_IND, LE_CM_MSG_ADVERTISE_REPORT_IND, LE_CM_MSG_DIRECT_ADV_REPORT_IND,
  LE_CM_CONNECTION_COMPLETE_IND,
  LE_CM_MSG_READ_LOCAL_RPA_CFM, LE_CM_MSG_TOP }

  *BLE connection management message id.*

### Functions

- void LeCmInit (TASK appTask)

  *BLE Connection Management Module Init.*

## 4.2.1 Detailed Description

## 4.2.2 Typedef Documentation

#### 4.2.2.1 LE_CM_MSG_ADD_TO_RESOLVING_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_ADD_TO_RESOLVING_LIST_CFM_T
```

#### 4.2.2.2 LE_CM_MSG_ADD_TO_WHITE_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_ADD_TO_WHITE_LIST_CFM_T
```

#### 4.2.2.3 LE_CM_MSG_CANCEL_CONNECTION_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_CANCEL_CONNECTION_CFM_T
```

#### 4.2.2.4 LE_CM_MSG_CLEAR_RESOLVING_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_CLEAR_RESOLVING_LIST_CFM_T
```

#### 4.2.2.5 LE_CM_MSG_CLEAR_WHITE_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_CLEAR_WHITE_LIST_CFM_T
```

#### 4.2.2.6 LE_CM_MSG_CREATE_CONNECTION_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_CREATE_CONNECTION_CFM_T
```

#### 4.2.2.7 LE_CM_MSG_ENTER_ADVERTISING_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_ENTER_ADVERTISING_CFM_T
```

#### 4.2.2.8 LE_CM_MSG_ENTER_SCANNING_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_ENTER_SCANNING_CFM_T
```

#### 4.2.2.9 LE_CM_MSG_EXIT_ADVERTISING_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_EXIT_ADVERTISING_CFM_T
```

#### 4.2.2.10 LE_CM_MSG_EXIT_SCANNING_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_EXIT_SCANNING_CFM_T
```

#### 4.2.2.11 LE_CM_MSG_REMOVE_FROM_RESOLVING_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_REMOVE_FROM_RESOLVING_LIST_CFM_T
```

#### 4.2.2.12 LE_CM_MSG_REMOVE_FROM_WHITE_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_REMOVE_FROM_WHITE_LIST_CFM_T
```

#### 4.2.2.13 LE_CM_MSG_SET_ADVERTISING_DATA_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_ADVERTISING_DATA_CFM_T
```

#### 4.2.2.14 LE_CM_MSG_SET_ADVERTISING_PARAMS_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_ADVERTISING_PARAMS_CFM_T
```

#### 4.2.2.15 LE_CM_MSG_SET_CHANNEL_MAP_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_CHANNEL_MAP_CFM_T
```

#### 4.2.2.16 LE_CM_MSG_SET_RANDOM_ADDRESS_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_RANDOM_ADDRESS_CFM_T
```

#### 4.2.2.17 LE_CM_MSG_SET_RPA_TIMEOUT_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_RPA_TIMEOUT_CFM_T
```

#### 4.2.2.18 LE_CM_MSG_SET_SCAN_PARAMS_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_SCAN_PARAMS_CFM_T
```

#### 4.2.2.19 LE_CM_MSG_SET_SCAN_RSP_DATA_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_SCAN_RSP_DATA_CFM_T
```

### 4.2.3 Enumeration Type Documentation

#### 4.2.3.1 anonymous enum

```
anonymous enum
```

BLE connection management message id.

**Enumerator**

| | |
|---|---|
| LE_CM_MSG_INIT_COMPLETE_CFM | initialize complete |
| LE_CM_MSG_SET_DISCONNECT_CFM | set disconnect confirm |
| LE_CM_MSG_DISCONNECT_COMPLETE_IND | disconnect complete indication |
| LE_CM_MSG_SET_ADVERTISING_DATA_CFM | set advertising data confirm |
| LE_CM_MSG_SET_SCAN_RSP_DATA_CFM | set scan response data confirm |
| LE_CM_MSG_SET_ADVERTISING_PARAMS_CFM | set advertising parameters confirm |
| LE_CM_MSG_ENTER_ADVERTISING_CFM | enter advertising confirm |
| LE_CM_MSG_EXIT_ADVERTISING_CFM | exit advertising confirm |
| LE_CM_MSG_SET_SCAN_PARAMS_CFM | set scan parameters confirm |
| LE_CM_MSG_ENTER_SCANNING_CFM | enter scanning confirm |
| LE_CM_MSG_EXIT_SCANNING_CFM | exit scanning confirm |
| LE_CM_MSG_CREATE_CONNECTION_CFM | create connection confirm |
| LE_CM_MSG_CANCEL_CONNECTION_CFM | cancel connection confirm |
| LE_CM_MSG_READ_TX_POWER_CFM | read tx power confirm |
| LE_CM_MSG_READ_BD_ADDR_CFM | read device address confirm |
| LE_CM_MSG_READ_RSSI_CFM | read RSSI confirm |
| LE_CM_MSG_SET_RANDOM_ADDRESS_CFM | set random address confirm |
| LE_CM_MSG_READ_ADV_TX_POWER_CFM | read advertising tx power confirm |
| LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM | read whitelist size confirm |
| LE_CM_MSG_CLEAR_WHITE_LIST_CFM | clear whitelist confirm |
| LE_CM_MSG_ADD_TO_WHITE_LIST_CFM | add to whitelist confirm |
| LE_CM_MSG_REMOVE_FROM_WHITE_LIST_CFM | remove from whitelist confirm |
| LE_CM_MSG_SET_CHANNEL_MAP_CFM | set channel map confirm |
| LE_CM_MSG_READ_CHANNEL_MAP_CFM | read channel map confirm |
| LE_CM_MSG_SET_DATA_LENGTH_CFM | set data length confirm |
| LE_CM_MSG_DATA_LEN_CHANGE_IND | data length change indication |
| LE_CM_MSG_ADD_TO_RESOLVING_LIST_CFM | add to resolving list confirm |
| LE_CM_MSG_REMOVE_FROM_RESOLVING_LIST_CFM | remove from resolving list confirm |
| LE_CM_MSG_CLEAR_RESOLVING_LIST_CFM | clear resolving list confirm |
| LE_CM_MSG_READ_RESOLVING_LIST_SIZE_CFM | read resolving list size confirm |
| LE_CM_MSG_SET_RPA_TIMEOUT_CFM | set resolving private address timeout confirm |
| LE_CM_MSG_SIGNAL_UPDATE_REQ | signal update request |

**Enumerator**

| | |
|---|---|
| LE_CM_MSG_CONN_UPDATE_COMPLETE_IND | connection update complete indication |
| LE_CM_MSG_CONN_PARA_REQ | connection parameters request |
| LE_CM_MSG_ENCRYPTION_CHANGE_IND | encryption change indication |
| LE_CM_MSG_ENCRYPTION_REFRESH_IND | encryption refresh indication |
| LE_CM_MSG_LTK_REQ_IND | long term key indication |
| LE_CM_MSG_ADVERTISE_REPORT_IND | advertising report indication |
| LE_CM_MSG_DIRECT_ADV_REPORT_IND | direct advertising report indication |
| LE_CM_CONNECTION_COMPLETE_IND | connection complete indication |
| LE_CM_MSG_READ_LOCAL_RPA_CFM | read local resolving private address confirm |
| LE_CM_MSG_TOP | top of CM message id |

### 4.2.4 Function Documentation

#### 4.2.4.1 LeCmInit()

```
void LeCmInit (
            TASK appTask )
```

BLE Connection Management Module Init.

**Parameters**

| | |
|---|---|
| *the* | reference of BLE task. |

**Returns**

None.

## 4.3 BLE GAP APIs

### Data Structures

- struct LE_GAP_ADVERTISING_PARAM_T
- struct LE_GAP_CONN_PARAM_T
- struct LE_GAP_SCAN_PARAM_T

### Macros

- #define GAP_ADTYPE_128BIT_COMPLETE 0x07
- #define GAP_ADTYPE_128BIT_MORE 0x06
- #define GAP_ADTYPE_16BIT_COMPLETE 0x03
- #define GAP_ADTYPE_16BIT_MORE 0x02
- #define GAP_ADTYPE_32BIT_COMPLETE 0x05
- #define GAP_ADTYPE_32BIT_MORE 0x04
- #define GAP_ADTYPE_3D_INFO_DATA 0x3D
- #define GAP_ADTYPE_ADV_INTERVAL 0x1A
- #define GAP_ADTYPE_APPEARANCE 0x19
- #define GAP_ADTYPE_FLAGS 0x01
- #define GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED 0x04
- #define GAP_ADTYPE_FLAGS_GENERAL 0x02
- #define GAP_ADTYPE_FLAGS_LIMITED 0x01
- #define GAP_ADTYPE_LE_BD_ADDR 0x1B
- #define GAP_ADTYPE_LE_ROLE 0x1C
- #define GAP_ADTYPE_LOCAL_NAME_COMPLETE 0x09
- #define GAP_ADTYPE_LOCAL_NAME_SHORT 0x08
- #define GAP_ADTYPE_MANUFACTURER_SPECIFIC 0xFF
- #define GAP_ADTYPE_OOB_CLASS_OF_DEVICE 0x0D
- #define GAP_ADTYPE_OOB_SIMPLE_PAIRING_HASHC 0x0E
- #define GAP_ADTYPE_OOB_SIMPLE_PAIRING_RANDR 0x0F
- #define GAP_ADTYPE_POWER_LEVEL 0x0A
- #define GAP_ADTYPE_PUBLIC_TARGET_ADDR 0x17
- #define GAP_ADTYPE_RANDOM_TARGET_ADDR 0x18
- #define GAP_ADTYPE_SERVICE_DATA 0x16
- #define GAP_ADTYPE_SERVICE_DATA_128BIT 0x21
- #define GAP_ADTYPE_SERVICE_DATA_32BIT 0x20
- #define GAP_ADTYPE_SERVICES_LIST_128BIT 0x15
- #define GAP_ADTYPE_SERVICES_LIST_16BIT 0x14
- #define GAP_ADTYPE_SIGNED_DATA 0x13
- #define GAP_ADTYPE_SIMPLE_PAIRING_HASHC_256 0x1D
- #define GAP_ADTYPE_SIMPLE_PAIRING_RANDR_256 0x1E
- #define GAP_ADTYPE_SLAVE_CONN_INTERVAL_RANGE 0x12
- #define GAP_ADTYPE_SM_OOB_FLAG 0x11
- #define GAP_ADTYPE_SM_TK 0x10
- #define GAP_PUBLIC_ADDR 0
- #define GAP_RAND_ADDR_NRPA 2
- #define GAP_RAND_ADDR_RPA 3
- #define GAP_RAND_ADDR_STATIC 1
- #define GAP_SCAN_TYPE_ACTIVE 1
- #define GAP_SCAN_TYPE_PASSIVE 0
- #define GAP_TX_PWR_CURR_VAL 0
- #define GAP_TX_PWR_MAX_VAL 1

- #define GAPBOND_IO_CAP_DISPLAY_ONLY 0x00
- #define GAPBOND_IO_CAP_DISPLAY_YES_NO 0x01
- #define GAPBOND_IO_CAP_KEYBOARD_DISPLAY 0x04
- #define GAPBOND_IO_CAP_KEYBOARD_ONLY 0x02
- #define GAPBOND_IO_CAP_NO_INPUT_NO_OUTPUT 0x03
- #define GAPBOND_PAIRING_MODE_INITIATE 0x02
- #define GAPBOND_PAIRING_MODE_NO_PAIRING 0x00
- #define GAPBOND_PAIRING_MODE_WAIT_FOR_REQ 0x01
- #define LE_GAP_ADV_MAX_SIZE 31

## Functions

- LE_ERR_STATE LeGapAddToResolvingList (LE_BT_ADDR_T *bt_addr, UINT8 *irk)

  *Add device to resolving-list.*
- LE_ERR_STATE LeGapAddToWhiteList (LE_BT_ADDR_T *bt_addr)

  *Add device to whitelist.*
- LE_ERR_STATE LeGapAdvertisingEnable (BOOL start)

  *Enable or disable advertising function.*
- LE_ERR_STATE LeGapCentralConnectReq (LE_BT_ADDR_T *taddr, UINT8 own_addr_type)

  *Central connect request.*
- LE_ERR_STATE LeGapCentralSetDataChannel (UINT8 *ch)

  *Central set data channel.*
- LE_ERR_STATE LeGapClearResolvingList (void)

  *Clear the resolving-list in the controller.*
- LE_ERR_STATE LeGapClearWhiteList (void)

  *Clear whitelist in the controller.*
- LE_ERR_STATE LeGapConnectCancelReq (void)

  *Cancel connect request.*
- void LeGapConnParaRequestRsp (UINT16 conn_hdl, BOOL accept)

  *Connection parameters request response.*
- void LeGapConnUpdateRequest (UINT16 conn_hdl, LE_CONN_PARA_T *para)

  *Connection parameters update request.*
- void LeGapConnUpdateResponse (UINT16 conn_hdl, UINT8 identifier, BOOL accept)

  *Connection parameters update response.*
- LE_ERR_STATE LeGapDisconnectReq (UINT16 conn_hdl)

  *Disconnect the physical connection.*
- LE_ERR_STATE LeGapGenRandAddr (UINT8 type, BD_ADDR addr)

  *Called to generation random address.*
- void LeGapGetBtAddr (void)

  *Get owner device address.*
- void LeGapReadAdvChannelTxPower (void)

  *Read ADV channel txpower.*
- LE_ERR_STATE LeGapReadChannelMap (UINT16 conn_hdl)

  *Read channel map.*
- void LeGapReadResolvingListSize (void)

  *Read the resolving-list size in the controller.*
- LE_ERR_STATE LeGapReadRssi (UINT16 conn_hdl)

  *Read RSSI value from controller.*
- LE_ERR_STATE LeGapReadTxPower (UINT16 conn_hdl, UINT8 type)

  *Read tx power value for the specified connection.*
- void LeGapReadWhiteListSize (void)

*Read whitelist size in the controller.*

- LE_ERR_STATE LeGapRemoveFromWhiteList (LE_BT_ADDR_T *bt_addr)

  *Remove device from whitelist.*
- LE_ERR_STATE LeGapScanningReq (BOOL start, BOOL filter)

  *Request scanning start.*
- LE_ERR_STATE LeGapSetAdvData (UINT8 len, UINT8 *data)

  *Called to set ADV data.*
- LE_ERR_STATE LeGapSetAdvParameter (LE_GAP_ADVERTISING_PARAM_T *params)

  *Called to set ADV parameters.*
- LE_ERR_STATE LeGapSetConnParameter (UINT16 interval_min, UINT16 interval_max, UINT16 slave_↩ latency, UINT16 supervision_timeout)

  *Called to set connection parameters.*
- LE_ERR_STATE LeGapSetDataChannelPduLen (UINT16 conn_hdl, UINT16 tx_octets, UINT16 tx_time)

  *Set data channel PDU length.*
- LE_ERR_STATE LeGapSetRandAddr (BD_ADDR addr)

  *Called to set random address.*
- LE_ERR_STATE LeGapSetRpaTimeout (UINT16 timeout)

  *Set resolvable private address timeout.*
- LE_ERR_STATE LeGapSetStaticAddr (BD_ADDR addr)

  *Called to set static address.*
- LE_ERR_STATE LeSetScanParameter (LE_GAP_SCAN_PARAM_T *params)

  *Called to set scan parameters.*
- LE_ERR_STATE LeSetScanRspData (UINT8 len, UINT8 *data)

  *Called to set scan response data.*

### 4.3.1 Detailed Description

### 4.3.2 Macro Definition Documentation

#### 4.3.2.1 GAP_ADTYPE_128BIT_COMPLETE

```
#define GAP_ADTYPE_128BIT_COMPLETE 0x07
```

#### 4.3.2.2 GAP_ADTYPE_128BIT_MORE

```
#define GAP_ADTYPE_128BIT_MORE 0x06
```

#### 4.3.2.3 GAP_ADTYPE_16BIT_COMPLETE

```
#define GAP_ADTYPE_16BIT_COMPLETE 0x03
```

#### 4.3.2.4 GAP_ADTYPE_16BIT_MORE

```
#define GAP_ADTYPE_16BIT_MORE 0x02
```

#### 4.3.2.5 GAP_ADTYPE_32BIT_COMPLETE

```
#define GAP_ADTYPE_32BIT_COMPLETE 0x05
```

#### 4.3.2.6 GAP_ADTYPE_32BIT_MORE

```
#define GAP_ADTYPE_32BIT_MORE 0x04
```

#### 4.3.2.7 GAP_ADTYPE_3D_INFO_DATA

```
#define GAP_ADTYPE_3D_INFO_DATA 0x3D
```

#### 4.3.2.8 GAP_ADTYPE_ADV_INTERVAL

```
#define GAP_ADTYPE_ADV_INTERVAL 0x1A
```

#### 4.3.2.9 GAP_ADTYPE_APPEARANCE

```
#define GAP_ADTYPE_APPEARANCE 0x19
```

#### 4.3.2.10 GAP_ADTYPE_FLAGS

```
#define GAP_ADTYPE_FLAGS 0x01
```

#### 4.3.2.11 GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED

```
#define GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED 0x04
```

#### 4.3.2.12 GAP_ADTYPE_FLAGS_GENERAL

```
#define GAP_ADTYPE_FLAGS_GENERAL 0x02
```

#### 4.3.2.13 GAP_ADTYPE_FLAGS_LIMITED

```
#define GAP_ADTYPE_FLAGS_LIMITED 0x01
```

#### 4.3.2.14 GAP_ADTYPE_LE_BD_ADDR

```
#define GAP_ADTYPE_LE_BD_ADDR 0x1B
```

#### 4.3.2.15 GAP_ADTYPE_LE_ROLE

```
#define GAP_ADTYPE_LE_ROLE 0x1C
```

#### 4.3.2.16 GAP_ADTYPE_LOCAL_NAME_COMPLETE

```
#define GAP_ADTYPE_LOCAL_NAME_COMPLETE 0x09
```

#### 4.3.2.17 GAP_ADTYPE_LOCAL_NAME_SHORT

```
#define GAP_ADTYPE_LOCAL_NAME_SHORT 0x08
```

#### 4.3.2.18 GAP_ADTYPE_MANUFACTURER_SPECIFIC

```
#define GAP_ADTYPE_MANUFACTURER_SPECIFIC 0xFF
```

#### 4.3.2.19 GAP_ADTYPE_OOB_CLASS_OF_DEVICE

```
#define GAP_ADTYPE_OOB_CLASS_OF_DEVICE 0x0D
```

#### 4.3.2.20 GAP_ADTYPE_OOB_SIMPLE_PAIRING_HASHC

```
#define GAP_ADTYPE_OOB_SIMPLE_PAIRING_HASHC 0x0E
```

#### 4.3.2.21 GAP_ADTYPE_OOB_SIMPLE_PAIRING_RANDR

```
#define GAP_ADTYPE_OOB_SIMPLE_PAIRING_RANDR 0x0F
```

#### 4.3.2.22 GAP_ADTYPE_POWER_LEVEL

```
#define GAP_ADTYPE_POWER_LEVEL 0x0A
```

#### 4.3.2.23 GAP_ADTYPE_PUBLIC_TARGET_ADDR

```
#define GAP_ADTYPE_PUBLIC_TARGET_ADDR 0x17
```

#### 4.3.2.24 GAP_ADTYPE_RANDOM_TARGET_ADDR

```
#define GAP_ADTYPE_RANDOM_TARGET_ADDR 0x18
```

#### 4.3.2.25 GAP_ADTYPE_SERVICE_DATA

```
#define GAP_ADTYPE_SERVICE_DATA 0x16
```

#### 4.3.2.26 GAP_ADTYPE_SERVICE_DATA_128BIT

```
#define GAP_ADTYPE_SERVICE_DATA_128BIT 0x21
```

#### 4.3.2.27 GAP_ADTYPE_SERVICE_DATA_32BIT

```
#define GAP_ADTYPE_SERVICE_DATA_32BIT 0x20
```

#### 4.3.2.28 GAP_ADTYPE_SERVICES_LIST_128BIT

```
#define GAP_ADTYPE_SERVICES_LIST_128BIT 0x15
```

#### 4.3.2.29 GAP_ADTYPE_SERVICES_LIST_16BIT

```
#define GAP_ADTYPE_SERVICES_LIST_16BIT 0x14
```

#### 4.3.2.30 GAP_ADTYPE_SIGNED_DATA

```
#define GAP_ADTYPE_SIGNED_DATA 0x13
```

#### 4.3.2.31 GAP_ADTYPE_SIMPLE_PAIRING_HASHC_256

```
#define GAP_ADTYPE_SIMPLE_PAIRING_HASHC_256 0x1D
```

#### 4.3.2.32 GAP_ADTYPE_SIMPLE_PAIRING_RANDR_256

```
#define GAP_ADTYPE_SIMPLE_PAIRING_RANDR_256 0x1E
```

#### 4.3.2.33 GAP_ADTYPE_SLAVE_CONN_INTERVAL_RANGE

```
#define GAP_ADTYPE_SLAVE_CONN_INTERVAL_RANGE 0x12
```

#### 4.3.2.34 GAP_ADTYPE_SM_OOB_FLAG

```
#define GAP_ADTYPE_SM_OOB_FLAG 0x11
```

#### 4.3.2.35 GAP_ADTYPE_SM_TK

```
#define GAP_ADTYPE_SM_TK 0x10
```

#### 4.3.2.36 GAP_PUBLIC_ADDR

```
#define GAP_PUBLIC_ADDR 0
```

#### 4.3.2.37 GAP_RAND_ADDR_NRPA

```
#define GAP_RAND_ADDR_NRPA 2
```

#### 4.3.2.38 GAP_RAND_ADDR_RPA

```
#define GAP_RAND_ADDR_RPA 3
```

#### 4.3.2.39 GAP_RAND_ADDR_STATIC

```
#define GAP_RAND_ADDR_STATIC 1
```

#### 4.3.2.40 GAP_SCAN_TYPE_ACTIVE

```
#define GAP_SCAN_TYPE_ACTIVE 1
```

#### 4.3.2.41 GAP_SCAN_TYPE_PASSIVE

```
#define GAP_SCAN_TYPE_PASSIVE 0
```

#### 4.3.2.42 GAP_TX_PWR_CURR_VAL

```
#define GAP_TX_PWR_CURR_VAL 0
```

#### 4.3.2.43 GAP_TX_PWR_MAX_VAL

```
#define GAP_TX_PWR_MAX_VAL 1
```

#### 4.3.2.44 GAPBOND_IO_CAP_DISPLAY_ONLY

```
#define GAPBOND_IO_CAP_DISPLAY_ONLY 0x00
```

#### 4.3.2.45 GAPBOND_IO_CAP_DISPLAY_YES_NO

```
#define GAPBOND_IO_CAP_DISPLAY_YES_NO 0x01
```

#### 4.3.2.46 GAPBOND_IO_CAP_KEYBOARD_DISPLAY

```
#define GAPBOND_IO_CAP_KEYBOARD_DISPLAY 0x04
```

#### 4.3.2.47 GAPBOND_IO_CAP_KEYBOARD_ONLY

```
#define GAPBOND_IO_CAP_KEYBOARD_ONLY 0x02
```

#### 4.3.2.48 GAPBOND_IO_CAP_NO_INPUT_NO_OUTPUT

```
#define GAPBOND_IO_CAP_NO_INPUT_NO_OUTPUT 0x03
```

#### 4.3.2.49 GAPBOND_PAIRING_MODE_INITIATE

```
#define GAPBOND_PAIRING_MODE_INITIATE 0x02
```

#### 4.3.2.50 GAPBOND_PAIRING_MODE_NO_PAIRING

```
#define GAPBOND_PAIRING_MODE_NO_PAIRING 0x00
```

#### 4.3.2.51 GAPBOND_PAIRING_MODE_WAIT_FOR_REQ

```
#define GAPBOND_PAIRING_MODE_WAIT_FOR_REQ 0x01
```

#### 4.3.2.52 LE_GAP_ADV_MAX_SIZE

```
#define LE_GAP_ADV_MAX_SIZE 31
```

### 4.3.3 Function Documentation

#### 4.3.3.1 LeGapAddToResolvingList()

```
LE_ERR_STATE LeGapAddToResolvingList (
            LE_BT_ADDR_T * bt_addr,
            UINT8 * irk )
```

Add device to resolving-list.

**Parameters**

| | |
|---|---|
| *bt_addr* | BT device address. |
| *irk* | IRK, Identity Resolving Key |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.2 LeGapAddToWhiteList()

```
LE_ERR_STATE LeGapAddToWhiteList (
            LE_BT_ADDR_T * bt_addr )
```

Add device to whitelist.

**Parameters**

| | |
|---|---|
| *bt_addr* | BT device address. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.3 LeGapAdvertisingEnable()

```
LE_ERR_STATE LeGapAdvertisingEnable (
             BOOL start )
```

Enable or disable advertising function.

**Parameters**

| | |
|---|---|
| *start* | TRUE is enable, FALSE is disable. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.4 LeGapCentralConnectReq()

```
LE_ERR_STATE LeGapCentralConnectReq (
             LE_BT_ADDR_T * taddr,
             UINT8 own_addr_type )
```

Central connect request.

**Parameters**

| | |
|---|---|
| *taddr* | advertisers device address. |
| *own_addr_type* | owner address type. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.5 LeGapCentralSetDataChannel()

```
LE_ERR_STATE LeGapCentralSetDataChannel (
             UINT8 * ch )
```

Central set data channel.

**Parameters**

| | |
|---|---|
| *ch* | data channel. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.6 LeGapClearResolvingList()

```
LE_ERR_STATE LeGapClearResolvingList (
            void  )
```

Clear the resolving-list in the controller.

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.7 LeGapClearWhiteList()

```
LE_ERR_STATE LeGapClearWhiteList (
            void  )
```

Clear whitelist in the controller.

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.8 LeGapConnectCancelReq()

```
LE_ERR_STATE LeGapConnectCancelReq (
            void  )
```

Cancel connect request.

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.9 LeGapConnParaRequestRsp()

```
void LeGapConnParaRequestRsp (
            UINT16 conn_hdl,
            BOOL accept )
```

Connection parameters request response.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *accept* | TRUE is accept, FALSE is not. |

**Returns**

None.

#### 4.3.3.10 LeGapConnUpdateRequest()

```
void LeGapConnUpdateRequest (
            UINT16 conn_hdl,
            LE_CONN_PARA_T * para )
```

Connection parameters update request.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *para* | update connection parameters. |

**Returns**

None.

#### 4.3.3.11 LeGapConnUpdateResponse()

```
void LeGapConnUpdateResponse (
            UINT16 conn_hdl,
            UINT8 identifier,
            BOOL accept )
```

Connection parameters update response.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *identifier* | TBD |
| *accept* | accept request, or not. |

**Returns**

None.

#### 4.3.3.12 LeGapDisconnectReq()

```
LE_ERR_STATE LeGapDisconnectReq (
             UINT16 conn_hdl )
```

Disconnect the physical connection.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.13 LeGapGenRandAddr()

```
LE_ERR_STATE LeGapGenRandAddr (
             UINT8 type,
             BD_ADDR addr )
```

Called to generation random address.

**Parameters**

| | |
|---|---|
| *type* | address type. |
| *addr* | address. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.14 LeGapGetBtAddr()

```
void LeGapGetBtAddr (
             void  )
```

Get owner device address.

#### 4.3.3.15 LeGapReadAdvChannelTxPower()

```
void LeGapReadAdvChannelTxPower (
            void  )
```

Read ADV channel txpower.

#### 4.3.3.16 LeGapReadChannelMap()

```
LE_ERR_STATE LeGapReadChannelMap (
            UINT16 conn_hdl )
```

Read channel map.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.17 LeGapReadResolvingListSize()

```
void LeGapReadResolvingListSize (
            void  )
```

Read the resolving-list size in the controller.

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.18 LeGapReadRssi()

```
LE_ERR_STATE LeGapReadRssi (
            UINT16 conn_hdl )
```

Read RSSI value from controller.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.19 LeGapReadTxPower()

```
LE_ERR_STATE LeGapReadTxPower (
             UINT16 conn_hdl,
             UINT8 type )
```

Read tx power value for the specified connection.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *type* | current tx power, or maxinum tx power. Don't support. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.20 LeGapReadWhiteListSize()

```
void LeGapReadWhiteListSize (
             void  )
```

Read whitelist size in the controller.

#### 4.3.3.21 LeGapRemoveFromWhiteList()

```
LE_ERR_STATE LeGapRemoveFromWhiteList (
             LE_BT_ADDR_T * bt_addr )
```

Remove device from whitelist.

Remove device from resolving-list.

**Parameters**

| | |
|---|---|
| *bt_addr* | BT device address. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.22 LeGapScanningReq()

```
LE_ERR_STATE LeGapScanningReq (
            BOOL start,
            BOOL filter )
```

Request scanning start.

**Parameters**

| | |
|---|---|
| *start* | TRUE is start, FALSE is not. |
| *filter* | scan policy, refer to LE_HCI_SCAN_FILT_* in ble_hci_if.h |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.23 LeGapSetAdvData()

```
LE_ERR_STATE LeGapSetAdvData (
            UINT8 len,
            UINT8 * data )
```

Called to set ADV data.

**Parameters**

| | |
|---|---|
| *len* | ADV data length. |
| *data* | ADV data. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.24 LeGapSetAdvParameter()

```
LE_ERR_STATE LeGapSetAdvParameter (
            LE_GAP_ADVERTISING_PARAM_T * params )
```

Called to set ADV parameters.

**Parameters**

| | |
|---|---|
| *params* | advertising params. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.25 LeGapSetConnParameter()

```
LE_ERR_STATE LeGapSetConnParameter (
             UINT16 interval_min,
             UINT16 interval_max,
             UINT16 slave_latency,
             UINT16 supervision_timeout )
```

Called to set connection parameters.

**Parameters**

| | |
|---|---|
| *interval_min* | mininum connection interval. |
| *interval_max* | maxinum connection interval. |
| *slave_letency* | slave letency. |
| *supervision_timeout* | supervison timeout. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.26 LeGapSetDataChannelPduLen()

```
LE_ERR_STATE LeGapSetDataChannelPduLen (
             UINT16 conn_hdl,
```

```
            UINT16 tx_octets,
            UINT16 tx_time )
```

Set data channel PDU length.

**Parameters**

| | |
|---|---|
| *tx_octets* | the maximum number of octets in the Payload field that the local device will send to the remote device. |
| *tx_time* | the maximum number of microseconds that the local device will take to transmit a PDU to the remote device. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.27 LeGapSetRandAddr()

```
LE_ERR_STATE LeGapSetRandAddr (
            BD_ADDR addr )
```

Called to set random address.

**Parameters**

| | |
|---|---|
| *addr* | the random address which should be set. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.28 LeGapSetRpaTimeout()

```
LE_ERR_STATE LeGapSetRpaTimeout (
            UINT16 timeout )
```

Set resolvable private address timeout.

**Parameters**

| | |
|---|---|
| *timeout* | RPA_Timeout, measured in seconds. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.29 LeGapSetStaticAddr()

```
LE_ERR_STATE LeGapSetStaticAddr (
             BD_ADDR addr )
```

Called to set static address.

**Parameters**

| | |
|---|---|
| *addr* | the static address which should be set. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.30 LeSetScanParameter()

```
LE_ERR_STATE LeSetScanParameter (
             LE_GAP_SCAN_PARAM_T * params )
```

Called to set scan parameters.

**Parameters**

| | |
|---|---|
| *params* | scan parameters. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.3.3.31 LeSetScanRspData()

```
LE_ERR_STATE LeSetScanRspData (
             UINT8 len,
             UINT8 * data )
```

Called to set scan response data.

**Parameters**

| | |
|---|---|
| *len* | scan response data length. |
| *data* | scan response data. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

## 4.4 BLE GATT APIs

### Data Structures

- struct LE_GATT_ATTR_T
- struct LE_GATT_MSG_ACCESS_READ_IND_T
- struct LE_GATT_MSG_ACCESS_WRITE_IND_T
- struct LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND_T
- struct LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND_T
- struct LE_GATT_MSG_CHARACTERISTIC_VAL_IND_T
- struct LE_GATT_MSG_CONFIRMATION_CFM_T
- struct LE_GATT_MSG_EXCHANGE_MTU_CFM_T
- struct LE_GATT_MSG_EXCHANGE_MTU_IND_T
- struct LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM_T
- struct LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM_T
- struct LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_CFM_T
- struct LE_GATT_MSG_FIND_CHARACTERISTIC_CFM_T
- struct LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM_T
- struct LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM_T
- struct LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T
- struct LE_GATT_MSG_INDICATE_IND_T
- struct LE_GATT_MSG_NOTIFY_CFM_T
- struct LE_GATT_MSG_NOTIFY_IND_T
- struct LE_GATT_MSG_OPERATION_TIMEOUT_T
- struct LE_GATT_MSG_PREPARE_WRITE_RELIABLE_CFM_T
- struct LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_CFM_T
- struct LE_GATT_MSG_READ_CHARACTERISTIC_VALUE_CFM_T
- struct LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM_T
- struct LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL_CFM_T
- struct LE_GATT_MSG_SERVICE_INFO_IND_T
- struct LE_GATT_MSG_SIGNED_WRITE_CFM_T
- struct LE_GATT_MSG_WRITE_CHAR_VAL_RELIABLE_CFM_T
- struct LE_GATT_MSG_WRITE_CHAR_VALUE_CFM_T
- struct LE_GATT_MSG_WRITE_LONG_CHAR_VALUE_CFM_T
- struct LE_GATT_MSG_WRITE_NO_RSP_CFM_T
- struct LE_GATT_SERVICE_T

### Macros

- #define CHAR_AGGREGATE_DESCRIPTOR(len, pVal) {0, LE_GATT_UUID16, (UINT16 *)&gcCharAggregateUuid, LE_GATT_PERMIT_READ, 0, len, (UINT8 *)(pVal)}
- #define CHAR_CLIENT_CONFIG_DESCRIPTOR(permit, pVal) {0, LE_GATT_UUID16, (UINT16 *)&gcClientCharConfigUuid, LE_GATT_PERMIT_READ | permit, 0, 2, (UINT8 *)(pVal)}
- #define CHAR_DECL_UUID16_ATTR_VAL(prop, type) {(prop), 0, 0, UINT16_LO(type), UINT16_HI(type)}
- #define CHAR_EXT_PROP_DESCRIPTOR(pVal) {0, LE_GATT_UUID16, (UINT16 *)&gcCharExtPropUuid, LE_GATT_PERMIT_READ, 0, 2, (UINT8 *)(pVal)}
- #define CHAR_PRESENT_FORMAT_DESCRIPTOR(pVal) {0, LE_GATT_UUID16, (UINT16 *)&gcCharFormatUuid, LE_GATT_PERMIT_READ, 0, 7, (UINT8 *)(pVal)}
- #define CHAR_SERVER_CONFIG_DESCRIPTOR(permit, pVal) {0, LE_GATT_UUID16, (UINT16 *)&gcServerCharConfigUuid, LE_GATT_PERMIT_READ | permit, 0, 2, (UINT8 *)(pVal)}
- #define CHAR_USER_DESC_DESCRIPTOR(permit, maxLen, len, pVal) {0, LE_GATT_UUID16, (UINT16 *)&gcCharUserDescUuid, permit, maxLen, len, (UINT8 *)(pVal)}

- #define CHARACTERISTIC_DECL_UUID128(pVal) {0, LE_GATT_UUID16, (UINT16 *)&gcCharacteristicUuid, LE_GATT_PERMIT_READ, 0, 19, (UINT8 *)(pVal)}
- #define CHARACTERISTIC_DECL_UUID16(pVal) {0, LE_GATT_UUID16, (UINT16 *)&gcCharacteristicUuid, LE_GATT_PERMIT_READ, 0, 5, (UINT8 *)(pVal)}
- #define CHARACTERISTIC_UUID128(pUuid, permit, maxLen, len, pVal) {0, LE_GATT_UUID128, (UINT16 *)pUuid, permit, maxLen, len, (UINT8 *)(pVal)}
- #define CHARACTERISTIC_UUID16(pUuid, permit, maxLen, len, pVal) {0, LE_GATT_UUID16, (UINT16 *)pUuid, permit, maxLen, len, (UINT8 *)(pVal)}
- #define GATT_CHAR_AGG_FORMAT_UUID 0x2905
- #define GATT_CHAR_EXT_PROPS_UUID 0x2900
- #define GATT_CHAR_FORMAT_UUID 0x2904
- #define GATT_CHAR_USER_DESC_UUID 0x2901
- #define GATT_CHARACTERISTIC_UUID 0x2803
- #define GATT_CLIENT_CHAR_CFG_UUID 0x2902
- #define GATT_EXT_REPORT_REF_UUID 0x2907
- #define GATT_INCLUDE_UUID 0x2802
- #define GATT_PRIMARY_SERVICE_UUID 0x2800
- #define GATT_REPORT_REF_UUID 0x2908
- #define GATT_SECONDARY_SERVICE_UUID 0x2801
- #define GATT_SERV_CHAR_CFG_UUID 0x2903
- #define GATT_VALID_RANGE_UUID 0x2906
- #define INCLUDE_DECL_UUID128(pVal) {0, LE_GATT_UUID16, (UINT16 *)&gcIncludeUuid, LE_GATT_PERMIT_READ, 0, 4, (UINT8 *)(pVal)}
- #define INCLUDE_DECL_UUID128_ATTR_VAL() {0, 0, 0, 0}
- #define INCLUDE_DECL_UUID16_ATTR_VAL(uuid) {0, 0, 0, 0, UINT16_LO(uuid), UINT16_HI(uuid)}
- #define INCLUDE_DECL_UUINT16(pVal) {0, LE_GATT_UUID16, (UINT16 *)&gcIncludeUuid, LE_GATT_PERMIT_READ, 0, 6, (UINT8 *)(pVal)}
- #define LE_ATT_UUID_SIZE 2
- #define LE_GATT_CHAR_PROP_AUTH 0x40
- #define LE_GATT_CHAR_PROP_BCAST 0x01

  *Characteristic Properties Bit.*
- #define LE_GATT_CHAR_PROP_EXT_PROP 0x80
- #define LE_GATT_CHAR_PROP_IND 0x20
- #define LE_GATT_CHAR_PROP_NTF 0x10
- #define LE_GATT_CHAR_PROP_RD 0x02
- #define LE_GATT_CHAR_PROP_WR 0x08
- #define LE_GATT_CHAR_PROP_WR_NO_RESP 0x04
- #define LE_GATT_CLIENT_CFG_INDICATION 0x02
- #define LE_GATT_CLIENT_CFG_NOTIFICATION 0x01
- #define LE_GATT_EXT_PROP_RELIABLE_WR 0x0001
- #define LE_GATT_EXT_PROP_WR_AUX 0x0002
- #define LE_GATT_FLAG_PREPARE_WRITE 0x02
- #define LE_GATT_FLAG_WRITE_CMD 0x01
- #define LE_GATT_FLAG_WRITE_REQ 0x00
- #define LE_GATT_PERM_AUTH_READABLE (0x1<<4)
- #define LE_GATT_PERM_AUTH_WRITABLE (0x1<<6)
- #define LE_GATT_PERM_NONE (0x00)
- #define LE_GATT_PERM_READ (0x1<<1)
- #define LE_GATT_PERM_RELIABLE_WRITE (0x1<<5)
- #define LE_GATT_PERM_WRITE_CMD (0x1<<2)
- #define LE_GATT_PERM_WRITE_REQ (0x1<<3)
- #define LE_GATT_PERMIT_AUTHEN_READ (0x0040)
- #define LE_GATT_PERMIT_AUTHEN_WRITE (0x0080)
- #define LE_GATT_PERMIT_AUTHOR_READ (0x0004)
- #define LE_GATT_PERMIT_AUTHOR_WRITE (0x0008)

- #define LE_GATT_PERMIT_ENCRYPT_READ (0x0010)
- #define LE_GATT_PERMIT_ENCRYPT_WRITE (0x0020)
- #define LE_GATT_PERMIT_READ (0x0001)
- #define LE_GATT_PERMIT_READABLE (LE_GATT_PERMIT_READ | LE_GATT_PERMIT_AUTHEN_READ | LE_GATT_PERMIT_AUTHOR_READ | LE_GATT_PERMIT_ENCRYPT_READ | LE_GATT_PERMIT_SC_AUTHEN_READ)
- #define LE_GATT_PERMIT_SC_AUTHEN_READ (0x0100)
- #define LE_GATT_PERMIT_SC_AUTHEN_WRITE (0x0200)
- #define LE_GATT_PERMIT_WRITABLE (LE_GATT_PERMIT_WRITE | LE_GATT_PERMIT_AUTHEN_WRITE | LE_GATT_PERMIT_AUTHOR_WRITE | LE_GATT_PERMIT_ENCRYPT_WRITE | LE_GATT_PERMIT_SC_AUTHEN_WRITE)
- #define LE_GATT_PERMIT_WRITE (0x0002)
- #define PRIMARY_SERVICE_DECL_UUID128(pUuid) {0, LE_GATT_UUID16, (UINT16 *)&gcPrimaryServiceUuid, LE_GATT_PERMIT_READ, 0, 16, (UINT8 *)(pUuid)}
- #define PRIMARY_SERVICE_DECL_UUID16(pUuid) {0, LE_GATT_UUID16, (UINT16 *)&gcPrimaryServiceUuid, LE_GATT_PERMIT_READ, 0, 2, (UINT8 *)(pUuid)}
- #define SECONDARY_SERVICE_DECL_UUID128(pUuid) {0, LE_GATT_UUID16, (UINT16 *)&gcSecondaryServiceUuid, LE_GATT_PERMIT_READ, 0, 16, (UINT8 *)(pUuid)}
- #define SECONDARY_SERVICE_DECL_UUID16(pUuid) {0, LE_GATT_UUID16, (UINT16 *)&gcSecondaryServiceUuid, LE_GATT_PERMIT_READ, 0, 2, (UINT8 *)(pUuid)}

## Enumerations

- enum {
  LE_GATT_MSG_INIT_CFM = LE_GATT_MSG_BASE, LE_GATT_MSG_EXCHANGE_MTU_IND,
  LE_GATT_MSG_EXCHANGE_MTU_CFM,
  LE_GATT_MSG_ACCESS_READ_IND,
  LE_GATT_MSG_ACCESS_WRITE_IND, LE_GATT_MSG_SERVICE_INFO_IND,
  LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_CFM,
  LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM,
  LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM, LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND,
  LE_GATT_MSG_FIND_CHARACTERISTIC_CFM, LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND,
  LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM, LE_GATT_MSG_CHARACTERISTIC_VAL_IND,
  LE_GATT_MSG_READ_CHARACTERISTIC_VALUE_CFM LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_CFM,
  LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM, LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL_CFM,
  LE_GATT_MSG_WRITE_CHAR_VALUE_CFM, LE_GATT_MSG_WRITE_LONG_CHAR_VALUE_CFM,
  LE_GATT_MSG_WRITE_CHAR_VAL_RELIABLE_CFM, LE_GATT_MSG_PREPARE_WRITE_RELIABLE_CFM,
  LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM, LE_GATT_MSG_WRITE_NO_RSP_CFM,
  LE_GATT_MSG_SIGNED_WRITE_CFM, LE_GATT_MSG_NOTIFY_IND, LE_GATT_MSG_NOTIFY_CFM,
  LE_GATT_MSG_INDICATE_IND,
  LE_GATT_MSG_CONFIRMATION_CFM, LE_GATT_MSG_OPERATION_TIMEOUT,
  LE_GATT_MSG_SIGN_RESOLUTION_FAIL,
  LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND,
  LE_GATT_MSG_TOP }

  *BLE GATT message id.*

## Functions

- LE_ERR_STATE LeGattAccessReadRsp (UINT16 conn_hdl, UINT16 handle, UINT8 att_err)

  *Gatt access read response.*
- LE_ERR_STATE LeGattAccessWriteRsp (UINT16 conn_hdl, UINT8 method, UINT16 handle, UINT8 att_err)

  *Gatt access write response.*
- LE_ERR_STATE LeGattChangeAttrVal (LE_GATT_SERVICE_T *svc, UINT16 attrId, UINT16 len, void *val)

  *Change attribute value.*
- LE_ERR_STATE LeGattCharValConfirmation (UINT16 conn_hdl)

  *Prepare write characteristic value response.*
- LE_ERR_STATE LeGattCharValIndicate (UINT16 conn_hdl, UINT16 hdl, UINT16 len, UINT8 *pval)

  *Gatt characteristic value indication.*

- LE_ERR_STATE LeGattCharValNotify (UINT16 conn_hdl, UINT16 hdl, UINT16 len, UINT8 *pval)

  *Gatt characteristic value notification.*
- LE_ERR_STATE LeGattExchangeMtuReq (UINT16 conn_hdl, UINT16 mtu)

  *Exchange MTU request.*
- LE_ERR_STATE LeGattExchangeMtuRsp (UINT16 conn_hdl, UINT16 mtu)

  *Exchange MTU response.*
- LE_ERR_STATE LeGattExecuteWriteCharValReliable (UINT16 conn_hdl, BOOL yesno)

  *Execute write characteristic value request.*
- LE_ERR_STATE LeGattFindAllCharacteristic (UINT16 conn_hdl, UINT16 start_hdl, UINT16 end_hdl)

  *Find all characteristic.*
- LE_ERR_STATE LeGattFindAllCharDescriptor (UINT16 conn_hdl, UINT16 start_hdl, UINT16 end_hdl)

  *Find all characteristic description.*
- LE_ERR_STATE LeGattFindAllPrimaryService (UINT16 conn_hdl)

  *Find all primary service.*
- LE_ERR_STATE LeGattFindCharacteristicByUuid (UINT16 conn_hdl, UINT16 start_hdl, UINT16 end_hdl, UINT8 format, UINT16 *uuid)

  *Find characteristic by UUID.*
- LE_ERR_STATE LeGattFindIncludedService (UINT16 conn_hdl, UINT16 start_hdl, UINT16 end_hdl)

  *Find include service.*
- LE_ERR_STATE LeGattFindPrimaryServiceByUuid (UINT16 conn_hdl, UINT8 format, UINT16 *uuid)

  *Find primary service by UUID.*
- UINT16 LeGattGetAttrHandle (LE_GATT_SERVICE_T *svc, UINT16 attrId)

  *Get attribute handle.*
- LE_ERR_STATE LeGattGetAttrVal (LE_GATT_SERVICE_T *svc, UINT16 attrId, UINT16 *len, void *val)

  *Get attribute value.*
- UINT16 LeGattGetAttrValLen (LE_GATT_SERVICE_T *svc, UINT16 attrId)

  *Get the length of attribute value.*
- UINT16 LeGattGetAttrValMaxLen (LE_GATT_SERVICE_T *svc, UINT16 attrId)

  *Get the max length of attribute value.*
- void LeGattInit (TASK appTask)

  *BLE Gatt module init.*
- LE_ERR_STATE LeGattModifyAttrVal (LE_GATT_SERVICE_T *svc, UINT16 attrId, UINT16 offset, UINT16 len, void *val)

  *Modify attribute value.*
- LE_ERR_STATE LeGattPrepareWriteCharValReliable (UINT16 conn_hdl, UINT16 handle, UINT16 offset, UINT16 len, UINT8 *val)

  *Prepare write characteristic value request.*
- LE_ERR_STATE LeGattReadCharValByUuid (UINT16 conn_hdl, UINT16 start_hdl, UINT16 end_hdl, UINT8 format, UINT16 *uuid)

  *Read a characteristic value by UUID.*
- LE_ERR_STATE LeGattReadCharValue (UINT16 conn_hdl, UINT16 handle)

  *Read a characteristic value.*
- LE_ERR_STATE LeGattReadLongCharVal (UINT16 conn_hdl, UINT16 handle, UINT16 offset)

  *Read a long characteristic value.*
- LE_ERR_STATE LeGattReadMultipleCharVal (UINT16 conn_hdl, UINT16 count, UINT16 *handle)

  *Read Multiple characteristic values.*
- LE_ERR_STATE LeGattRegisterIncludeService (UINT16 inc_hdl, UINT16 start_hdl, UINT16 end_hdl, UINT16 uuid)

  *Called to register an include service.*
- LE_GATT_SERVICE_T * LeGattRegisterService (LE_GATT_ATTR_T *attrTable, UINT16 numAttr)

  *Called to register a service.*

- LE_ERR_STATE LeGattSignedWriteNoRsp (UINT16 conn_hdl, UINT16 handle, UINT16 len, UINT8 *val)

  *Signed write without response.*
- void LeGattStopCurrentProcedure (UINT16 conn_hdl)

  *Stop current procedure.*
- LE_ERR_STATE LeGattWriteCharVal (UINT16 conn_hdl, UINT16 handle, UINT16 len, UINT8 *val)

  *Write characteristic value.*
- LE_ERR_STATE LeGattWriteCharValReliable (UINT16 conn_hdl, UINT16 handle, UINT16 offset, UINT16 len, UINT8 *val)

  *Write characteristic value reliable.*
- LE_ERR_STATE LeGattWriteLongCharVal (UINT16 conn_hdl, UINT16 handle, UINT16 offset, UINT16 len, UINT8 *val)

  *Write long characteristic value.*
- LE_ERR_STATE LeGattWriteNoRsp (UINT16 conn_hdl, UINT16 handle, UINT16 len, UINT8 *val)

  *Write without response.*

## Variables

- const UINT16 gcCharacteristicUuid
- const UINT16 gcCharAggregateUuid
- const UINT16 gcCharExtPropUuid
- const UINT16 gcCharFormatUuid
- const UINT16 gcCharUserDescUuid
- const UINT16 gcClientCharConfigUuid
- const UINT16 gcExtReportRefUuid
- const UINT16 gcIncludeUuid
- const UINT16 gcPrimaryServiceUuid
- const UINT16 gcReportRefUuid
- const UINT16 gcSecondaryServiceUuid
- const UINT16 gcServerCharConfigUuid
- const UINT16 gcValidRangeUuid

### 4.4.1 Detailed Description

### 4.4.2 Macro Definition Documentation

#### 4.4.2.1 CHAR_AGGREGATE_DESCRIPTOR

```
#define CHAR_AGGREGATE_DESCRIPTOR(
              len,
              pVal ) {0, LE_GATT_UUID16, (UINT16 *)&gcCharAggregateUuid, LE_GATT_PERMIT_READ,
0, len, (UINT8 *)(pVal)}
```

#### 4.4.2.2 CHAR_CLIENT_CONFIG_DESCRIPTOR

```
#define CHAR_CLIENT_CONFIG_DESCRIPTOR(
              permit,
              pVal ) {0, LE_GATT_UUID16, (UINT16 *)&gcClientCharConfigUuid, LE_GATT_PERMIT_READ
| permit, 0, 2, (UINT8 *)(pVal)}
```

#### 4.4.2.3 CHAR_DECL_UUID16_ATTR_VAL

```
#define CHAR_DECL_UUID16_ATTR_VAL(
              prop,
              type ) {(prop), 0, 0, UINT16_LO(type), UINT16_HI(type)}
```

#### 4.4.2.4 CHAR_EXT_PROP_DESCRIPTOR

```
#define CHAR_EXT_PROP_DESCRIPTOR(
              pVal ) {0, LE_GATT_UUID16, (UINT16 *)&gcCharExtPropUuid, LE_GATT_PERMIT_READ, 0,
2, (UINT8 *)(pVal)}
```

#### 4.4.2.5 CHAR_PRESENT_FORMAT_DESCRIPTOR

```
#define CHAR_PRESENT_FORMAT_DESCRIPTOR(
              pVal ) {0, LE_GATT_UUID16, (UINT16 *)&gcCharFormatUuid, LE_GATT_PERMIT_READ, 0,
7, (UINT8 *)(pVal)}
```

#### 4.4.2.6 CHAR_SERVER_CONFIG_DESCRIPTOR

```
#define CHAR_SERVER_CONFIG_DESCRIPTOR(
              permit,
              pVal ) {0, LE_GATT_UUID16, (UINT16 *)&gcServerCharConfigUuid, LE_GATT_PERMIT_READ
| permit, 0, 2, (UINT8 *)(pVal)}
```

#### 4.4.2.7 CHAR_USER_DESC_DESCRIPTOR

```
#define CHAR_USER_DESC_DESCRIPTOR(
              permit,
              maxLen,
              len,
              pVal ) {0, LE_GATT_UUID16, (UINT16 *)&gcCharUserDescUuid, permit, maxLen, len,
(UINT8 *)(pVal)}
```

#### 4.4.2.8 CHARACTERISTIC_DECL_UUID128

```
#define CHARACTERISTIC_DECL_UUID128(
              pVal ) {0, LE_GATT_UUID16, (UINT16 *)&gcCharacteristicUuid, LE_GATT_PERMIT_READ,
0, 19, (UINT8 *)(pVal)}
```

#### 4.4.2.9 CHARACTERISTIC_DECL_UUID16

```
#define CHARACTERISTIC_DECL_UUID16(
              pVal ) {0, LE_GATT_UUID16, (UINT16 *)&gcCharacteristicUuid, LE_GATT_PERMIT_READ,
0, 5, (UINT8 *)(pVal)}
```

#### 4.4.2.10 CHARACTERISTIC_UUID128

```
#define CHARACTERISTIC_UUID128(
              pUuid,
              permit,
              maxLen,
              len,
              pVal ) {0, LE_GATT_UUID128, (UINT16 *)pUuid, permit, maxLen, len, (UINT8 *)(p↩
Val)}
```

#### 4.4.2.11 CHARACTERISTIC_UUID16

```
#define CHARACTERISTIC_UUID16(
              pUuid,
              permit,
              maxLen,
              len,
              pVal ) {0, LE_GATT_UUID16, (UINT16 *)pUuid, permit, maxLen, len, (UINT8 *)(pVal)}
```

#### 4.4.2.12 GATT_CHAR_AGG_FORMAT_UUID

```
#define GATT_CHAR_AGG_FORMAT_UUID 0x2905
```

#### 4.4.2.13 GATT_CHAR_EXT_PROPS_UUID

```
#define GATT_CHAR_EXT_PROPS_UUID 0x2900
```

#### 4.4.2.14 GATT_CHAR_FORMAT_UUID

```
#define GATT_CHAR_FORMAT_UUID 0x2904
```

#### 4.4.2.15 GATT_CHAR_USER_DESC_UUID

```
#define GATT_CHAR_USER_DESC_UUID 0x2901
```

#### 4.4.2.16 GATT_CHARACTERISTIC_UUID

```
#define GATT_CHARACTERISTIC_UUID 0x2803
```

#### 4.4.2.17 GATT_CLIENT_CHAR_CFG_UUID

```
#define GATT_CLIENT_CHAR_CFG_UUID 0x2902
```

#### 4.4.2.18 GATT_EXT_REPORT_REF_UUID

```
#define GATT_EXT_REPORT_REF_UUID 0x2907
```

#### 4.4.2.19 GATT_INCLUDE_UUID

```
#define GATT_INCLUDE_UUID 0x2802
```

#### 4.4.2.20 GATT_PRIMARY_SERVICE_UUID

```
#define GATT_PRIMARY_SERVICE_UUID 0x2800
```

#### 4.4.2.21 GATT_REPORT_REF_UUID

```
#define GATT_REPORT_REF_UUID 0x2908
```

#### 4.4.2.22 GATT_SECONDARY_SERVICE_UUID

```
#define GATT_SECONDARY_SERVICE_UUID 0x2801
```

#### 4.4.2.23 GATT_SERV_CHAR_CFG_UUID

```
#define GATT_SERV_CHAR_CFG_UUID 0x2903
```

#### 4.4.2.24 GATT_VALID_RANGE_UUID

```
#define GATT_VALID_RANGE_UUID 0x2906
```

#### 4.4.2.25 INCLUDE_DECL_UUID128

```
#define INCLUDE_DECL_UUID128(
              pVal ) {0, LE_GATT_UUID16, (UINT16 *)&gcIncludeUuid, LE_GATT_PERMIT_READ, 0, 4,
(UINT8 *)(pVal)}
```

#### 4.4.2.26 INCLUDE_DECL_UUID128_ATTR_VAL

```
#define INCLUDE_DECL_UUID128_ATTR_VAL( ) {0, 0, 0, 0}
```

#### 4.4.2.27 INCLUDE_DECL_UUID16_ATTR_VAL

```
#define INCLUDE_DECL_UUID16_ATTR_VAL(
              uuid ) {0, 0, 0, 0, UINT16_LO(uuid), UINT16_HI(uuid)}
```

#### 4.4.2.28 INCLUDE_DECL_UUINT16

```
#define INCLUDE_DECL_UUINT16(
              pVal ) {0, LE_GATT_UUID16, (UINT16 *)&gcIncludeUuid, LE_GATT_PERMIT_READ, 0, 6,
(UINT8 *)(pVal)}
```

#### 4.4.2.29 LE_ATT_UUID_SIZE

```
#define LE_ATT_UUID_SIZE 2
```

#### 4.4.2.30 LE_GATT_CHAR_PROP_AUTH

```
#define LE_GATT_CHAR_PROP_AUTH 0x40
```

#### 4.4.2.31 LE_GATT_CHAR_PROP_BCAST

```
#define LE_GATT_CHAR_PROP_BCAST 0x01
```

Characteristic Properties Bit.

#### 4.4.2.32 LE_GATT_CHAR_PROP_EXT_PROP

```
#define LE_GATT_CHAR_PROP_EXT_PROP 0x80
```

#### 4.4.2.33 LE_GATT_CHAR_PROP_IND

```
#define LE_GATT_CHAR_PROP_IND 0x20
```

#### 4.4.2.34 LE_GATT_CHAR_PROP_NTF

```
#define LE_GATT_CHAR_PROP_NTF 0x10
```

#### 4.4.2.35 LE_GATT_CHAR_PROP_RD

```
#define LE_GATT_CHAR_PROP_RD 0x02
```

#### 4.4.2.36 LE_GATT_CHAR_PROP_WR

```
#define LE_GATT_CHAR_PROP_WR 0x08
```

#### 4.4.2.37 LE_GATT_CHAR_PROP_WR_NO_RESP

```
#define LE_GATT_CHAR_PROP_WR_NO_RESP 0x04
```

#### 4.4.2.38 LE_GATT_CLIENT_CFG_INDICATION

```
#define LE_GATT_CLIENT_CFG_INDICATION 0x02
```

#### 4.4.2.39 LE_GATT_CLIENT_CFG_NOTIFICATION

```
#define LE_GATT_CLIENT_CFG_NOTIFICATION 0x01
```

#### 4.4.2.40 LE_GATT_EXT_PROP_RELIABLE_WR

```
#define LE_GATT_EXT_PROP_RELIABLE_WR 0x0001
```

#### 4.4.2.41 LE_GATT_EXT_PROP_WR_AUX

```
#define LE_GATT_EXT_PROP_WR_AUX 0x0002
```

#### 4.4.2.42 LE_GATT_FLAG_PREPARE_WRITE

```
#define LE_GATT_FLAG_PREPARE_WRITE 0x02
```

#### 4.4.2.43 LE_GATT_FLAG_WRITE_CMD

```
#define LE_GATT_FLAG_WRITE_CMD 0x01
```

#### 4.4.2.44 LE_GATT_FLAG_WRITE_REQ

```
#define LE_GATT_FLAG_WRITE_REQ 0x00
```

#### 4.4.2.45 LE_GATT_PERM_AUTH_READABLE

```
#define LE_GATT_PERM_AUTH_READABLE (0x1<<4)
```

#### 4.4.2.46 LE_GATT_PERM_AUTH_WRITABLE

```
#define LE_GATT_PERM_AUTH_WRITABLE (0x1<<6)
```

#### 4.4.2.47 LE_GATT_PERM_NONE

```
#define LE_GATT_PERM_NONE (0x00)
```

#### 4.4.2.48 LE_GATT_PERM_READ

```
#define LE_GATT_PERM_READ (0x1<<1)
```

#### 4.4.2.49 LE_GATT_PERM_RELIABLE_WRITE

```
#define LE_GATT_PERM_RELIABLE_WRITE (0x1<<5)
```

#### 4.4.2.50 LE_GATT_PERM_WRITE_CMD

```
#define LE_GATT_PERM_WRITE_CMD (0x1<<2)
```

#### 4.4.2.51 LE_GATT_PERM_WRITE_REQ

```
#define LE_GATT_PERM_WRITE_REQ (0x1<<3)
```

#### 4.4.2.52 LE_GATT_PERMIT_AUTHEN_READ

```
#define LE_GATT_PERMIT_AUTHEN_READ (0x0040)
```

#### 4.4.2.53 LE_GATT_PERMIT_AUTHEN_WRITE

```
#define LE_GATT_PERMIT_AUTHEN_WRITE (0x0080)
```

#### 4.4.2.54 LE_GATT_PERMIT_AUTHOR_READ

```
#define LE_GATT_PERMIT_AUTHOR_READ (0x0004)
```

#### 4.4.2.55 LE_GATT_PERMIT_AUTHOR_WRITE

```
#define LE_GATT_PERMIT_AUTHOR_WRITE (0x0008)
```

#### 4.4.2.56 LE_GATT_PERMIT_ENCRYPT_READ

```
#define LE_GATT_PERMIT_ENCRYPT_READ (0x0010)
```

#### 4.4.2.57 LE_GATT_PERMIT_ENCRYPT_WRITE

```
#define LE_GATT_PERMIT_ENCRYPT_WRITE (0x0020)
```

#### 4.4.2.58 LE_GATT_PERMIT_READ

```
#define LE_GATT_PERMIT_READ (0x0001)
```

#### 4.4.2.59 LE_GATT_PERMIT_READABLE

```
#define LE_GATT_PERMIT_READABLE (LE_GATT_PERMIT_READ | LE_GATT_PERMIT_AUTHEN_READ | LE_GATT_PERMIT_AUTHOR_READ
| LE_GATT_PERMIT_ENCRYPT_READ | LE_GATT_PERMIT_SC_AUTHEN_READ)
```

#### 4.4.2.60 LE_GATT_PERMIT_SC_AUTHEN_READ

```
#define LE_GATT_PERMIT_SC_AUTHEN_READ (0x0100)
```

#### 4.4.2.61 LE_GATT_PERMIT_SC_AUTHEN_WRITE

```
#define LE_GATT_PERMIT_SC_AUTHEN_WRITE (0x0200)
```

#### 4.4.2.62 LE_GATT_PERMIT_WRITABLE

```
#define LE_GATT_PERMIT_WRITABLE (LE_GATT_PERMIT_WRITE | LE_GATT_PERMIT_AUTHEN_WRITE |
LE_GATT_PERMIT_AUTHOR_WRITE| LE_GATT_PERMIT_ENCRYPT_WRITE | LE_GATT_PERMIT_SC_AUTHEN_WRITE)
```

#### 4.4.2.63 LE_GATT_PERMIT_WRITE

```
#define LE_GATT_PERMIT_WRITE (0x0002)
```

#### 4.4.2.64 PRIMARY_SERVICE_DECL_UUID128

```
#define PRIMARY_SERVICE_DECL_UUID128(
              pUuid ) {0, LE_GATT_UUID16, (UINT16 *)&gcPrimaryServiceUuid, LE_GATT_PERMIT_READ,
0, 16, (UINT8 *)(pUuid)}
```

#### 4.4.2.65 PRIMARY_SERVICE_DECL_UUID16

```
#define PRIMARY_SERVICE_DECL_UUID16(
              pUuid ) {0, LE_GATT_UUID16, (UINT16 *)&gcPrimaryServiceUuid, LE_GATT_PERMIT_READ,
0, 2, (UINT8 *)(pUuid)}
```

#### 4.4.2.66 SECONDARY_SERVICE_DECL_UUID128

```
#define SECONDARY_SERVICE_DECL_UUID128(
              pUuid ) {0, LE_GATT_UUID16, (UINT16 *)&gcSecondaryServiceUuid, LE_GATT_PERMIT_READ,
0, 16, (UINT8 *)(pUuid)}
```

#### 4.4.2.67 SECONDARY_SERVICE_DECL_UUID16

```
#define SECONDARY_SERVICE_DECL_UUID16(
              pUuid ) {0, LE_GATT_UUID16, (UINT16 *)&gcSecondaryServiceUuid, LE_GATT_PERMIT_READ,
0, 2, (UINT8 *)(pUuid)}
```

### 4.4.3 Enumeration Type Documentation

#### 4.4.3.1 anonymous enum

```
anonymous enum
```

BLE GATT message id.

**Enumerator**

| | |
|---|---|
| LE_GATT_MSG_INIT_CFM | initialize confirm message |
| LE_GATT_MSG_EXCHANGE_MTU_IND | exchange MTU indication |
| LE_GATT_MSG_EXCHANGE_MTU_CFM | exchange MTU confirm |
| LE_GATT_MSG_ACCESS_READ_IND | access read indication |
| LE_GATT_MSG_ACCESS_WRITE_IND | access write indication |
| LE_GATT_MSG_SERVICE_INFO_IND | service infomation indication |
| LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_CFM | find all primary service confirm |
| LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM | find primary service by UUID fonfirm |
| LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM | find include service confirm |
| LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND | characteristic declaration info indication |
| LE_GATT_MSG_FIND_CHARACTERISTIC_CFM | find characteristic confirm |
| LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND | characteristic descriptor info indication |
| LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM | find all characteristic descriptors confirm |
| LE_GATT_MSG_CHARACTERISTIC_VAL_IND | characteristic value, indication message |
| LE_GATT_MSG_READ_CHARACTERISTIC_VALUE_CFM | read characteristic value, confirm message |
| LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_CFM | read characteristic value by UUID confirm message |
| LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM | read long characteristic value confirm mesage |
| LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL_CFM | read multiple characteristic value confirm |
| LE_GATT_MSG_WRITE_CHAR_VALUE_CFM | write characteristic value confirm |
| LE_GATT_MSG_WRITE_LONG_CHAR_VALUE_CFM | write long characteristic value confirm |
| LE_GATT_MSG_WRITE_CHAR_VAL_RELIABLE_CFM | write characteristic value reliable confirm |
| LE_GATT_MSG_PREPARE_WRITE_RELIABLE_CFM | prepare write reliable confirm |
| LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM | execute write reliable confirm |

**Enumerator**

| | |
|---|---|
| LE_GATT_MSG_WRITE_NO_RSP_CFM | write no response confirm |
| LE_GATT_MSG_SIGNED_WRITE_CFM | signed write confirm |
| LE_GATT_MSG_NOTIFY_IND | notify indication |
| LE_GATT_MSG_NOTIFY_CFM | notify confirm |
| LE_GATT_MSG_INDICATE_IND | indicate indication |
| LE_GATT_MSG_CONFIRMATION_CFM | confirmation confirm |
| LE_GATT_MSG_OPERATION_TIMEOUT | operation timeout |
| LE_GATT_MSG_SIGN_RESOLUTION_FAIL | sign resolution fail |
| LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND | include service infomation |
| LE_GATT_MSG_TOP | top of GATT message id |

### 4.4.4 Function Documentation

#### 4.4.4.1 LeGattAccessReadRsp()

```
LE_ERR_STATE LeGattAccessReadRsp (
             UINT16 conn_hdl,
             UINT16 handle,
             UINT8 att_err )
```

Gatt access read response.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *handle* | attribute handle. |
| *att_err* | 0 is OK, others refer to LE_ATT_ERR_∗ in ble_att_if.h. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.2 LeGattAccessWriteRsp()

```
LE_ERR_STATE LeGattAccessWriteRsp (
             UINT16 conn_hdl,
             UINT8 method,
             UINT16 handle,
             UINT8 att_err )
```

Gatt access write response.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *method* | refer to LE_GATT_FLAG_* in ble_gatt_if.h |
| *handle* | attribute handle. |
| *att_err* | 0 is OK, others refer to LE_ATT_ERR_* in ble_att_if.h. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.3 LeGattChangeAttrVal()

```
LE_ERR_STATE LeGattChangeAttrVal (
             LE_GATT_SERVICE_T * svc,
             UINT16 attrId,
             UINT16 len,
             void * val )
```

Change attribute value.

**Parameters**

| | | |
|---|---|---|
| | *svc* | service. |
| | *attr↩ Id* | attribute index of service. |
| in | *len* | attribute value length. |
| in | *val* | attribute value. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.4 LeGattCharValConfirmation()

```
LE_ERR_STATE LeGattCharValConfirmation (
             UINT16 conn_hdl )
```

Prepare write characteristic value response.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.5 LeGattCharValIndicate()

```
LE_ERR_STATE LeGattCharValIndicate (
             UINT16 conn_hdl,
             UINT16 hdl,
             UINT16 len,
             UINT8 * pval )
```

Gatt characteristic value indication.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *hdl* | characteristic value handle. |
| *len* | value length. |
| *pval* | value. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.6 LeGattCharValNotify()

```
LE_ERR_STATE LeGattCharValNotify (
             UINT16 conn_hdl,
             UINT16 hdl,
             UINT16 len,
             UINT8 * pval )
```

Gatt characteristic value notification.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *hdl* | characteristic value handle. |
| *len* | value length. |
| *pval* | value. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.7 LeGattExchangeMtuReq()

```
LE_ERR_STATE LeGattExchangeMtuReq (
             UINT16 conn_hdl,
             UINT16 mtu )
```

Exchange MTU request.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *mtu* | MTU. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.8 LeGattExchangeMtuRsp()

```
LE_ERR_STATE LeGattExchangeMtuRsp (
             UINT16 conn_hdl,
             UINT16 mtu )
```

Exchange MTU response.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *mtu* | MTU. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.9 LeGattExecuteWriteCharValReliable()

```
LE_ERR_STATE LeGattExecuteWriteCharValReliable (
             UINT16 conn_hdl,
             BOOL yesno )
```

Execute write characteristic value request.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *yesno* | execute write or not. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.10 LeGattFindAllCharacteristic()

```
LE_ERR_STATE LeGattFindAllCharacteristic (
             UINT16 conn_hdl,
             UINT16 start_hdl,
             UINT16 end_hdl )
```

Find all characteristic.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *start_hdl* | start handle. |
| *end_hdl* | end handle. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.11 LeGattFindAllCharDescriptor()

```
LE_ERR_STATE LeGattFindAllCharDescriptor (
             UINT16 conn_hdl,
             UINT16 start_hdl,
             UINT16 end_hdl )
```

Find all characteristic description.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *start_hdl* | start handle. |
| *end_hdl* | end handle. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.12 LeGattFindAllPrimaryService()

```
LE_ERR_STATE LeGattFindAllPrimaryService (
             UINT16 conn_hdl )
```

Find all primary service.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.13 LeGattFindCharacteristicByUuid()

```
LE_ERR_STATE LeGattFindCharacteristicByUuid (
             UINT16 conn_hdl,
             UINT16 start_hdl,
             UINT16 end_hdl,
             UINT8 format,
             UINT16 * uuid )
```

Find characteristic by UUID.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *start_hdl* | start handle. |
| *end_hdl* | end handle. |
| *format* | UUID type. |
| *uuid* | UUID. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.14 LeGattFindIncludedService()

```
LE_ERR_STATE LeGattFindIncludedService (
             UINT16 conn_hdl,
             UINT16 start_hdl,
             UINT16 end_hdl )
```

Find include service.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *start_hdl* | start handle. |
| *end_hdl* | end handle. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.15 LeGattFindPrimaryServiceByUuid()

```
LE_ERR_STATE LeGattFindPrimaryServiceByUuid (
             UINT16 conn_hdl,
             UINT8 format,
             UINT16 * uuid )
```

Find primary service by UUID.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *format* | UUID type. |
| *uuid* | UUID. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.16 LeGattGetAttrHandle()

```
UINT16 LeGattGetAttrHandle (
            LE_GATT_SERVICE_T * svc,
            UINT16 attrId )
```

Get attribute handle.

**Parameters**

| | |
|---|---|
| *svc* | service. |
| *attr↩ Id* | attribute index of service. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.17 LeGattGetAttrVal()

```
LE_ERR_STATE LeGattGetAttrVal (
            LE_GATT_SERVICE_T * svc,
            UINT16 attrId,
            UINT16 * len,
            void * val )
```

Get attribute value.

**Parameters**

| | | |
|---|---|---|
| | *svc* | service. |
| | *attr↩ Id* | attribute index of service. |
| out | *len* | attribute value length. |
| out | *val* | attribute value. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.18 LeGattGetAttrValLen()

```
UINT16 LeGattGetAttrValLen (
            LE_GATT_SERVICE_T * svc,
            UINT16 attrId )
```

Get the length of attribute value.

**Parameters**

| | |
|---|---|
| *svc* | service. |
| *attr↩ Id* | attribute index of service. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.19 LeGattGetAttrValMaxLen()

```
UINT16 LeGattGetAttrValMaxLen (
            LE_GATT_SERVICE_T * svc,
            UINT16 attrId )
```

Get the max length of attribute value.

**Parameters**

| | |
|---|---|
| *svc* | service. |
| *attr↩ Id* | attribute index of service. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.20 LeGattInit()

```
void LeGattInit (
            TASK appTask )
```

BLE Gatt module init.

**Parameters**

| | |
|---|---|
| *appTask* | the reference of BLE task. |

**Returns**

None.

#### 4.4.4.21 LeGattModifyAttrVal()

```
LE_ERR_STATE LeGattModifyAttrVal (
             LE_GATT_SERVICE_T * svc,
             UINT16 attrId,
             UINT16 offset,
             UINT16 len,
             void * val )
```

Modify attribute value.

**Parameters**

| | |
|---|---|
| *svc* | servie. |
| *attrId* | attribute index of service. |
| *offset* | modify offset. |
| *len* | modify length. |
| *val* | modify value. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.22 LeGattPrepareWriteCharValReliable()

```
LE_ERR_STATE LeGattPrepareWriteCharValReliable (
             UINT16 conn_hdl,
             UINT16 handle,
             UINT16 offset,
             UINT16 len,
             UINT8 * val )
```

Prepare write characteristic value request.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *handle* | characteristic value handle. |
| *offset* | offset written. |
| *len* | length written. |
| *val* | value. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.23 LeGattReadCharValByUuid()

```
LE_ERR_STATE LeGattReadCharValByUuid (
             UINT16 conn_hdl,
             UINT16 start_hdl,
             UINT16 end_hdl,
             UINT8 format,
             UINT16 * uuid )
```

Read a characteristic value by UUID.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *start_hdl* | start handle. |
| *end_hdl* | end handle. |
| *format* | UUID type. |
| *uuid* | UUID. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.24 LeGattReadCharValue()

```
LE_ERR_STATE LeGattReadCharValue (
             UINT16 conn_hdl,
             UINT16 handle )
```

Read a characteristic value.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *handle* | characteristic value handle. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.25 LeGattReadLongCharVal()

```
LE_ERR_STATE LeGattReadLongCharVal (
            UINT16 conn_hdl,
            UINT16 handle,
            UINT16 offset )
```

Read a long characteristic value.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *handle* | characteristic value handle. |
| *offset* | characteristic value offset. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.26 LeGattReadMultipleCharVal()

```
LE_ERR_STATE LeGattReadMultipleCharVal (
            UINT16 conn_hdl,
            UINT16 count,
            UINT16 * handle )
```

Read Multiple characteristic values.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *count* | handle count. |
| *handle* | handle table. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.27 LeGattRegisterIncludeService()

```
LE_ERR_STATE LeGattRegisterIncludeService (
            UINT16 inc_hdl,
```

```
            UINT16 start_hdl,
            UINT16 end_hdl,
            UINT16 uuid )
```

Called to register an include service.

**Parameters**

| | |
|---|---|
| *inc_hdl* | include service handle. |
| *start_hdl* | start handle. |
| *end_hdl* | end handle. |
| *uuid* | include service UUID. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.28 LeGattRegisterService()

```
LE_GATT_SERVICE_T* LeGattRegisterService (
            LE_GATT_ATTR_T * attrTable,
            UINT16 numAttr )
```

Called to register a service.

**Parameters**

| | |
|---|---|
| *attrTable* | service attribute table. |
| *numAttr* | the attribute number of service. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.29 LeGattSignedWriteNoRsp()

```
LE_ERR_STATE LeGattSignedWriteNoRsp (
            UINT16 conn_hdl,
            UINT16 handle,
            UINT16 len,
            UINT8 * val )
```

Signed write without response.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *handle* | characteristic value handle. |
| *len* | length of the data to be written. |
| *val* | the value to be written. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.30 LeGattStopCurrentProcedure()

```
void LeGattStopCurrentProcedure (
            UINT16 conn_hdl )
```

Stop current procedure.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.31 LeGattWriteCharVal()

```
LE_ERR_STATE LeGattWriteCharVal (
            UINT16 conn_hdl,
            UINT16 handle,
            UINT16 len,
            UINT8 * val )
```

Write characteristic value.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *handle* | characteristic value handle. |
| *len* | length of the data to be written. |
| *val* | the value to be written. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.32 LeGattWriteCharValReliable()

```
LE_ERR_STATE LeGattWriteCharValReliable (
             UINT16 conn_hdl,
             UINT16 handle,
             UINT16 offset,
             UINT16 len,
             UINT8 * val )
```

Write characteristic value reliable.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *handle* | characteristic value handle. |
| *offset* | offset written. |
| *len* | length written. |
| *val* | value. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.33 LeGattWriteLongCharVal()

```
LE_ERR_STATE LeGattWriteLongCharVal (
             UINT16 conn_hdl,
             UINT16 handle,
             UINT16 offset,
             UINT16 len,
             UINT8 * val )
```

Write long characteristic value.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *handle* | characteristic value handle. |
| *offset* | value position offset. |
| *len* | length of the data to be written. |
| *val* | the value to be written. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

#### 4.4.4.34 LeGattWriteNoRsp()

```
LE_ERR_STATE LeGattWriteNoRsp (
             UINT16 conn_hdl,
             UINT16 handle,
             UINT16 len,
             UINT8 * val )
```

Write without response.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *handle* | characteristic value handle. |
| *len* | length of the data to be written. |
| *val* | the value to be written. |

**Returns**

- SYS_ERR_SUCCESS: success.
- others: refer to error code in ble_err.h.

### 4.4.5 Variable Documentation

#### 4.4.5.1 gcCharacteristicUuid

```
const UINT16 gcCharacteristicUuid
```

#### 4.4.5.2 gcCharAggregateUuid

```
const UINT16 gcCharAggregateUuid
```

#### 4.4.5.3 gcCharExtPropUuid

```
const UINT16 gcCharExtPropUuid
```

#### 4.4.5.4 gcCharFormatUuid

```
const UINT16 gcCharFormatUuid
```

#### 4.4.5.5 gcCharUserDescUuid

```
const UINT16 gcCharUserDescUuid
```

#### 4.4.5.6 gcClientCharConfigUuid

```
const UINT16 gcClientCharConfigUuid
```

#### 4.4.5.7 gcExtReportRefUuid

```
const UINT16 gcExtReportRefUuid
```

#### 4.4.5.8 gcIncludeUuid

```
const UINT16 gcIncludeUuid
```

#### 4.4.5.9 gcPrimaryServiceUuid

```
const UINT16 gcPrimaryServiceUuid
```

#### 4.4.5.10 gcReportRefUuid

```
const UINT16 gcReportRefUuid
```

#### 4.4.5.11 gcSecondaryServiceUuid

```
const UINT16 gcSecondaryServiceUuid
```

#### 4.4.5.12 gcServerCharConfigUuid

```
const UINT16 gcServerCharConfigUuid
```

#### 4.4.5.13 gcValidRangeUuid

```
const UINT16 gcValidRangeUuid
```

## 4.5 BLE MSG APIs

### Data Structures

- struct LE_SYS_MSG_BUF_OVERFLOW_T

### Macros

- #define LE_ATT_MSG_BASE 0x1400
- #define LE_CM_MSG_BASE 0x1100
- #define LE_GATT_MSG_BASE 0x1500
- #define LE_HCI_MSG_BASE 0x1000
- #define LE_L2CAP_MSG_BASE 0x1200
- #define LE_SMP_MSG_BASE 0x1300
- #define LE_SYS_MSG_BASE 0x8000
- #define MESSAGE_ALLOCATE(M, S) PanicUnlessMalloc(sizeof(M##_T) + S)
- #define MESSAGE_BULID(M) M##_T *msg = PanicUnlessMalloc(sizeof(M##_T))
- #define MESSAGE_DATA_BULID(M, S) M##_T *msg = PanicUnlessMalloc(sizeof(M##_T) + S)
- #define MESSAGE_OFFSET(M) ((UINT8 *)msg + sizeof(M##_T))
- #define T_HOUR(h) ((UINT32)((h) * (UINT32)1000 * (UINT32)60) * (UINT32)60)
- #define T_MIN(m) ((UINT32)((m) * (UINT32)1000 * (UINT32)60))
- #define T_SEC(s) ((UINT32)((s) * (UINT32)1000))

### Typedefs

- typedef MsgData MESSAGE
- typedef UINT16 MESSAGEID
- typedef void const * MsgData
- typedef const UINT8 * MsgLock
- typedef MsgLock MSGLOCK
- typedef UINT16 MSGSUBID
- typedef UINT32 MSGTIMER
- typedef TASKPACK * Task
- typedef Task TASK
- typedef void(* TASKHANDLER) (Task, UINT16, MsgData)
- typedef void ** TASKPACK

### Enumerations

- enum { LE_SYS_MSG_BUF_OVERFLOW = (LE_SYS_MSG_BASE + 1), LE_SYS_MSG_TOP }

  *BLE system message id.*

### Functions

- UINT16 LeCancelAllMessage (TASK task, MESSAGEID id)

  *Cancel all message in queue.*
- UINT16 LeCancelAllSubMessage (TASK task, MESSAGEID id, MSGSUBID subId)

  *Cancel all sub message in queue.*
- BOOL LeCancelFirstMessage (TASK task, MESSAGEID id)

  *Cancel the first message in queue.*
- BOOL LeCancelFirstSubMessage (TASK task, MESSAGEID id, MSGSUBID subId)

  *Cancel the first sub message in queue.*
- UINT16 LeGetSubMsgId (UINT16 *s)

  *Get sub message id.*
- BOOL LeHostCreateTask (TASK task, TASKHANDLER hdl)

  *Create BLE task.*
- void LeHostMessageLoop (void)

  *message loop run.*
- void LeSendMessage (TASK task, MESSAGEID msgId, MESSAGE msg)

  *Send message to BLE task.*
- void LeSendMessageAfter (TASK task, MESSAGEID msgId, MESSAGE msg, UINT32 delay)

  *Delay, then send message to BLE task.*
- void LeSendMessageUnlock (TASK task, MESSAGEID id, MESSAGE msg, MSGLOCK lock)

  *Send message until lock is 0.*
- void LeSendSubMessage (TASK task, MESSAGEID msgId, MSGSUBID subId, MESSAGE msg)

  *Send sub message.*
- void LeSendSubMessageAfter (TASK task, MESSAGEID msgId, MSGSUBID subId, MESSAGE msg, UIN↩ T32 delay)

  *Delay, then send sub message.*
- void LeSendSubMessageUnlock (TASK task, MESSAGEID id, MSGSUBID subId, MESSAGE msg, MSGLOCK lock)

  *Send sub message until lock is 0.*

## 4.5.1 Detailed Description

## 4.5.2 Macro Definition Documentation

### 4.5.2.1 LE_ATT_MSG_BASE

```
#define LE_ATT_MSG_BASE 0x1400
```

### 4.5.2.2 LE_CM_MSG_BASE

```
#define LE_CM_MSG_BASE 0x1100
```

#### 4.5.2.3 LE_GATT_MSG_BASE

```
#define LE_GATT_MSG_BASE 0x1500
```

#### 4.5.2.4 LE_HCI_MSG_BASE

```
#define LE_HCI_MSG_BASE 0x1000
```

#### 4.5.2.5 LE_L2CAP_MSG_BASE

```
#define LE_L2CAP_MSG_BASE 0x1200
```

#### 4.5.2.6 LE_SMP_MSG_BASE

```
#define LE_SMP_MSG_BASE 0x1300
```

#### 4.5.2.7 LE_SYS_MSG_BASE

```
#define LE_SYS_MSG_BASE 0x8000
```

#### 4.5.2.8 MESSAGE_ALLOCATE

```
#define MESSAGE_ALLOCATE(
              M,
              S ) PanicUnlessMalloc(sizeof(M##_T) + S)
```

#### 4.5.2.9 MESSAGE_BULID

```
#define MESSAGE_BULID(
              M ) M##_T *msg = PanicUnlessMalloc(sizeof(M##_T))
```

#### 4.5.2.10 MESSAGE_DATA_BULID

```
#define MESSAGE_DATA_BULID(
              M,
              S ) M##_T *msg = PanicUnlessMalloc(sizeof(M##_T) + S)
```

#### 4.5.2.11 MESSAGE_OFFSET

```
#define MESSAGE_OFFSET(
              M ) ((UINT8 *)msg + sizeof(M##_T))
```

#### 4.5.2.12 T_HOUR

```
#define T_HOUR(
              h ) ((UINT32)((h) * (UINT32)1000 * (UINT32)60) * (UINT32)60)
```

#### 4.5.2.13 T_MIN

```
#define T_MIN(
              m ) ((UINT32)((m) * (UINT32)1000 * (UINT32)60))
```

#### 4.5.2.14 T_SEC

```
#define T_SEC(
              s ) ((UINT32)((s) * (UINT32)1000))
```

### 4.5.3 Typedef Documentation

#### 4.5.3.1 MESSAGE

```
typedef MsgData MESSAGE
```

#### 4.5.3.2 MESSAGEID

```
typedef UINT16 MESSAGEID
```

#### 4.5.3.3 MsgData

```
typedef void const* MsgData
```

#### 4.5.3.4 MsgLock

```
typedef const UINT8* MsgLock
```

#### 4.5.3.5 MSGLOCK

```
typedef MsgLock MSGLOCK
```

#### 4.5.3.6 MSGSUBID

```
typedef UINT16 MSGSUBID
```

#### 4.5.3.7 MSGTIMER

```
typedef UINT32 MSGTIMER
```

#### 4.5.3.8 Task

```
typedef TASKPACK* Task
```

#### 4.5.3.9 TASK

```
typedef Task TASK
```

#### 4.5.3.10 TASKHANDLER

```
typedef void(* TASKHANDLER) (Task, UINT16, MsgData)
```

#### 4.5.3.11 TASKPACK

```
typedef void** TASKPACK
```

### 4.5.4 Enumeration Type Documentation

#### 4.5.4.1 anonymous enum

```
anonymous enum
```

BLE system message id.

**Enumerator**

| | |
|---|---|
| LE_SYS_MSG_BUF_OVERFLOW | message buffer overflow |
| LE_SYS_MSG_TOP | top of system message id |

### 4.5.5 Function Documentation

#### 4.5.5.1 LeCancelAllMessage()

```
UINT16 LeCancelAllMessage (
            TASK task,
            MESSAGEID id )
```

Cancel all message in queue.

**Parameters**

| | |
|---|---|
| *task* | task. |
| *id* | message id. |

**Returns**

0 is ok, others is error.

#### 4.5.5.2 LeCancelAllSubMessage()

```
UINT16 LeCancelAllSubMessage (
            TASK task,
            MESSAGEID id,
            MSGSUBID subId )
```

Cancel all sub message in queue.

**Parameters**

| | |
|---|---|
| *task* | the task of recvice message. |
| *id* | message id. |
| *subId* | sub message id. |

**Returns**

0 is ok, others is error.

#### 4.5.5.3 LeCancelFirstMessage()

```
BOOL LeCancelFirstMessage (
            TASK task,
            MESSAGEID id )
```

Cancel the first message in queue.

**Parameters**

| | |
|---|---|
| *task* | task. |
| *id* | message id. |

**Returns**

True is ok, false is error.

#### 4.5.5.4 LeCancelFirstSubMessage()

```
BOOL LeCancelFirstSubMessage (
            TASK task,
            MESSAGEID id,
            MSGSUBID subId )
```

Cancel the first sub message in queue.

**Parameters**

| | |
|---|---|
| *task* | the task of recvice message. |
| *id* | message id. |
| *subId* | sub message id. |

**Returns**

True is ok, false is error.

#### 4.5.5.5 LeGetSubMsgId()

```
UINT16 LeGetSubMsgId (
            UINT16 * s )
```

Get sub message id.

**Parameters**

| | |
|---|---|
| *sub* | message id. |

**Returns**

0 is ok, others is error.

#### 4.5.5.6 LeHostCreateTask()

```
BOOL LeHostCreateTask (
            TASK task,
            TASKHANDLER hdl )
```

Create BLE task.

**Parameters**

| | |
|---|---|
| *task* | the reference of BLE task. |
| *hdl* | callback handle of BLE task. |

**Returns**

TRUE is success, FALSE is failed.

#### 4.5.5.7 LeHostMessageLoop()

```
void LeHostMessageLoop (
            void  )
```

message loop run.

**Returns**

None.

#### 4.5.5.8 LeSendMessage()

```
void LeSendMessage (
            TASK task,
            MESSAGEID msgId,
            MESSAGE msg )
```

Send message to BLE task.

**Parameters**

| | |
|---|---|
| *task* | reference of BLE task. |
| *msg↩ Id* | message ID. |
| *msg* | message. |

**Returns**

None.

#### 4.5.5.9 LeSendMessageAfter()

```
void LeSendMessageAfter (
            TASK task,
```

```
            MESSAGEID msgId,
            MESSAGE msg,
            UINT32 delay )
```

Delay, then send message to BLE task.

**Parameters**

| | |
|---|---|
| *task* | reference of BLE task. |
| *msg↩ Id* | message ID. |
| *msg* | message. |
| *delay* | delay time, ms. |

**Returns**

None.

#### 4.5.5.10 LeSendMessageUnlock()

```
void LeSendMessageUnlock (
            TASK task,
            MESSAGEID id,
            MESSAGE msg,
            MSGLOCK lock )
```

Send message until lock is 0.

**Parameters**

| | |
|---|---|
| *task* | the task of recvice message. |
| *id* | message id. |
| *msg* | message. |
| *lock* | lock number. |

**Returns**

None.

#### 4.5.5.11 LeSendSubMessage()

```
void LeSendSubMessage (
            TASK task,
            MESSAGEID msgId,
            MSGSUBID subId,
            MESSAGE msg )
```

Send sub message.

**Parameters**

| | |
|---|---|
| *task* | the task of recvice message. |
| *msg↩ Id* | message id. |
| *subId* | sub message id. |
| *msg* | message. |

**Returns**

None.

#### 4.5.5.12 LeSendSubMessageAfter()

```
void LeSendSubMessageAfter (
            TASK task,
            MESSAGEID msgId,
            MSGSUBID subId,
            MESSAGE msg,
            UINT32 delay )
```

Delay, then send sub message.

**Parameters**

| | |
|---|---|
| *task* | the task of recvice message. |
| *msg↩ Id* | message id. |
| *subId* | sub message id. |
| *msg* | message. |
| *delay* | delay time. |

**Returns**

None.

#### 4.5.5.13 LeSendSubMessageUnlock()

```
void LeSendSubMessageUnlock (
            TASK task,
            MESSAGEID id,
            MSGSUBID subId,
            MESSAGE msg,
            MSGLOCK lock )
```

Send sub message until lock is 0.

**Parameters**

| | |
|---|---|
| *task* | the task of recvice message. |
| *id* | message id. |
| *sub↩ Id* | sub message id. |
| *msg* | message. |
| *lock* | lock number. |

**Returns**

None.

## 4.6 BLE SMP APIs

### Data Structures

- struct LE_SMP_MSG_ENCRYPTION_CHANGE_IND_T
- struct LE_SMP_MSG_ENCRYPTION_REFRESH_IND_T
- struct LE_SMP_MSG_OOB_DATA_REQUEST_IND_T
- struct LE_SMP_MSG_PAIRING_ACTION_IND_T
- struct LE_SMP_MSG_PAIRING_COMPLETE_IND_T
- struct LE_SMP_MSG_PASSKEY_DISPLAY_IND_T
- struct LE_SMP_MSG_PASSKEY_INPUT_IND_T
- struct LE_SMP_MSG_SC_OOB_DATA_REQUEST_IND_T
- struct LE_SMP_MSG_SLAVE_SECURITY_REQUEST_IND_T
- struct LE_SMP_MSG_USER_CONFIRM_IND_T
- struct LE_SMP_SC_OOB_DATA_T

### Macros

- #define LE_MAX_BOND_COUNT 8
- #define LE_SM_IO_CAP_DISP_ONLY 0x00
- #define LE_SM_IO_CAP_DISP_YES_NO 0x01
- #define LE_SM_IO_CAP_KEYBOARD_DISP 0x04
- #define LE_SM_IO_CAP_KEYBOARD_ONLY 0x02
- #define LE_SM_IO_CAP_NO_IO 0x03
- #define LE_SM_PAIR_MITM_NO 0x00
- #define LE_SM_PAIR_MITM_YES 0x01
- #define LE_SM_PAIR_OOB_NO 0x00
- #define LE_SM_PAIR_OOB_YES 0x01
- #define LE_SM_PAIR_SC_NO 0x00
- #define LE_SM_PAIR_SC_YES 0x01

### Enumerations

- enum {
  LE_SMP_MSG_SLAVE_SECURITY_REQUEST_IND = LE_SMP_MSG_BASE, LE_SMP_MSG_PAIRING_ACTION_IND,
  LE_SMP_MSG_PASSKEY_DISPLAY_IND, LE_SMP_MSG_PASSKEY_INPUT_IND,
  LE_SMP_MSG_OOB_DATA_REQUEST_IND, LE_SMP_MSG_SC_OOB_DATA_REQUEST_IND,
  LE_SMP_MSG_USER_CONFIRM_IND LE_SMP_MSG_ENCRYPTION_CHANGE_IND,
  LE_SMP_MSG_ENCRYPTION_REFRESH_IND, LE_SMP_MSG_PAIRING_COMPLETE_IND,
  LE_SMP_LONG_TERM_KEY_REQ,
  LE_SMP_KEYS_IND,
  LE_SMP_MSG_TOP }
  *BLE SMP message Id.*
- enum {
  LE_SMP_PAIR_JUST_WORK, LE_SMP_PAIR_OOB, LE_SMP_PAIR_PASSKEY_INPUT, LE_SMP_PAIR_DISPLAY,
  LE_SMP_PAIR_NUM_COMPARE }

### Functions

- void LeSmpInit (TASK appTask)

  *BLE SMP Module Init.*
- void LeSmpOobAuthDataRsp (UINT16 conn_hdl, UINT8 *data, UINT16 len)

  *SMP OOB authenticate data response.*
- UINT16 LeSmpOobPresent (UINT16 conn_hdl, BOOL oob_present)

  *SMP OOB present.*
- void LeSmpPasskeyInput (UINT16 conn_hdl, UINT32 passkey)

  *Input passkey.*
- UINT16 LeSmpScOobComputeConfirmVal (UINT8 *rand, UINT8 *confirm)

  *SMP secure connection OOB compute confirm value.*
- void LeSmpScOobDataRsp (UINT16 conn_hdl, UINT8 *our_rand, LE_SMP_SC_OOB_DATA_T *peer)

  *OOB data response.*
- UINT16 LeSmpSecurityReq (UINT16 conn_hdl)

  *BLE SMP security request.*
- UINT16 LeSmpSecurityRsp (UINT16 conn_hdl, BOOL accept)

  *BLE SMP security request.*
- UINT16 LeSmpSetDefaultConfig (UINT8 iocap, BOOL mitm, BOOL sc, BOOL bond)

  *Set default configure for pairing.*
- UINT16 LeSmpUserConfirmRsp (UINT16 conn_hdl, BOOL accept)

  *User confirm response.*

## 4.6.1 Detailed Description

## 4.6.2 Macro Definition Documentation

### 4.6.2.1 LE_MAX_BOND_COUNT

```
#define LE_MAX_BOND_COUNT 8
```

### 4.6.2.2 LE_SM_IO_CAP_DISP_ONLY

```
#define LE_SM_IO_CAP_DISP_ONLY 0x00
```

display only

### 4.6.2.3 LE_SM_IO_CAP_DISP_YES_NO

```
#define LE_SM_IO_CAP_DISP_YES_NO 0x01
```

display + yes or no

#### 4.6.2.4 LE_SM_IO_CAP_KEYBOARD_DISP

```
#define LE_SM_IO_CAP_KEYBOARD_DISP 0x04
```

display + keyboard

#### 4.6.2.5 LE_SM_IO_CAP_KEYBOARD_ONLY

```
#define LE_SM_IO_CAP_KEYBOARD_ONLY 0x02
```

keyboard only

#### 4.6.2.6 LE_SM_IO_CAP_NO_IO

```
#define LE_SM_IO_CAP_NO_IO 0x03
```

no input and output

#### 4.6.2.7 LE_SM_PAIR_MITM_NO

```
#define LE_SM_PAIR_MITM_NO 0x00
```

#### 4.6.2.8 LE_SM_PAIR_MITM_YES

```
#define LE_SM_PAIR_MITM_YES 0x01
```

#### 4.6.2.9 LE_SM_PAIR_OOB_NO

```
#define LE_SM_PAIR_OOB_NO 0x00
```

#### 4.6.2.10 LE_SM_PAIR_OOB_YES

```
#define LE_SM_PAIR_OOB_YES 0x01
```

#### 4.6.2.11 LE_SM_PAIR_SC_NO

```
#define LE_SM_PAIR_SC_NO 0x00
```

#### 4.6.2.12 LE_SM_PAIR_SC_YES

```
#define LE_SM_PAIR_SC_YES 0x01
```

### 4.6.3 Enumeration Type Documentation

#### 4.6.3.1 anonymous enum

```
anonymous enum
```

BLE SMP message id.

**Enumerator**

| | |
|---|---|
| LE_SMP_MSG_SLAVE_SECURITY_REQUEST_IND | slave security request |
| LE_SMP_MSG_PAIRING_ACTION_IND | pairing action indication |
| LE_SMP_MSG_PASSKEY_DISPLAY_IND | passkey display indication |
| LE_SMP_MSG_PASSKEY_INPUT_IND | passkey input indication |
| LE_SMP_MSG_OOB_DATA_REQUEST_IND | OOB date request indication |
| LE_SMP_MSG_SC_OOB_DATA_REQUEST_IND | SC OOB data request indication |
| LE_SMP_MSG_USER_CONFIRM_IND | user confirm indication |
| LE_SMP_MSG_ENCRYPTION_CHANGE_IND | encryption change indication |
| LE_SMP_MSG_ENCRYPTION_REFRESH_IND | encryption refresh indication |
| LE_SMP_MSG_PAIRING_COMPLETE_IND | pairing complete indication |
| LE_SMP_LONG_TERM_KEY_REQ | long term key request |
| LE_SMP_KEYS_IND | keys indication |
| LE_SMP_MSG_TOP | top of SMP message id |

#### 4.6.3.2 anonymous enum

```
anonymous enum
```

**Enumerator**

| | |
|---|---|
| LE_SMP_PAIR_JUST_WORK | just work |
| LE_SMP_PAIR_OOB | out of band |
| LE_SMP_PAIR_PASSKEY_INPUT | passkey entry |
| LE_SMP_PAIR_DISPLAY | display |
| LE_SMP_PAIR_NUM_COMPARE | number compare |

### 4.6.4 Function Documentation

#### 4.6.4.1 LeSmpInit()

```
void LeSmpInit (
            TASK appTask )
```

BLE SMP Module Init.

**Parameters**

| | |
|---|---|
| *appTask* | the reference of BLE task. |

**Returns**

None.

#### 4.6.4.2 LeSmpOobAuthDataRsp()

```
void LeSmpOobAuthDataRsp (
            UINT16 conn_hdl,
            UINT8 * data,
            UINT16 len )
```

SMP OOB authenticate data response.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *data* | response data. |
| *len* | data length. |

**Returns**

None.

#### 4.6.4.3 LeSmpOobPresent()

```
UINT16 LeSmpOobPresent (
            UINT16 conn_hdl,
            BOOL oob_present )
```

SMP OOB present.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *oob_present* | present or not. |

**Returns**

0 is Ok, others refer to SMP_ERR_∗ in ble_err.h.

#### 4.6.4.4 LeSmpPasskeyInput()

```
void LeSmpPasskeyInput (
            UINT16 conn_hdl,
            UINT32 passkey )
```

Input passkey.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *passkey* | passkey. |

**Returns**

None.

#### 4.6.4.5 LeSmpScOobComputeConfirmVal()

```
UINT16 LeSmpScOobComputeConfirmVal (
            UINT8 * rand,
            UINT8 * confirm )
```

SMP secure connection OOB compute confirm value.

**Parameters**

| | |
|---|---|
| *rand* | random data. |
| *confirm* | confirm data. |

**Returns**

0 is Ok, others refer to SMP_ERR_∗ in ble_err.h.

#### 4.6.4.6 LeSmpScOobDataRsp()

```
void LeSmpScOobDataRsp (
             UINT16 conn_hdl,
             UINT8 * our_rand,
             LE_SMP_SC_OOB_DATA_T * peer )
```

OOB data response.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handld. |
| *our_rand* | our random data. |
| *peer* | peer OOB data. |

**Returns**

None.

#### 4.6.4.7 LeSmpSecurityReq()

```
UINT16 LeSmpSecurityReq (
             UINT16 conn_hdl )
```

BLE SMP security request.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |

**Returns**

0 is Ok, others refer to SMP_ERR_∗ in ble_err.h.

#### 4.6.4.8 LeSmpSecurityRsp()

```
UINT16 LeSmpSecurityRsp (
             UINT16 conn_hdl,
             BOOL accept )
```

BLE SMP security request.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *accept* | TRUE is accept, FALSE is not. |

**Returns**

0 is Ok, others refer to SMP_ERR_∗ in ble_err.h.

#### 4.6.4.9 LeSmpSetDefaultConfig()

```
UINT16 LeSmpSetDefaultConfig (
            UINT8 iocap,
            BOOL mitm,
            BOOL sc,
            BOOL bond )
```

Set default configure for pairing.

**Parameters**

| | |
|---|---|
| *iocap* | IO capability. |
| *mitm* | TRUE is MITM protected, FALSE is not. |
| *sc* | TRUE is request BLE secure connection pairing, FALSE is not. |
| *bond* | TRUE: bonding, FALSE: no bonding. |

**Returns**

0 is Ok, others refer to SMP_ERR_∗ in ble_err.h.

#### 4.6.4.10 LeSmpUserConfirmRsp()

```
UINT16 LeSmpUserConfirmRsp (
            UINT16 conn_hdl,
            BOOL accept )
```

User confirm response.

**Parameters**

| | |
|---|---|
| *conn_hdl* | connection handle. |
| *accept* | yes or no. |

**Returns**

0 is Ok, others refer to SMP_ERR_∗ in ble_err.h.

## 4.7 WIFI APIs

WIFI APIs.

### Modules

- WIFI Common APIs
- WIFI STA APIs
- Enumeration

### Data Structures

- struct wifi_active_scan_time_t

  *Range of active scan times per channel.*
- struct wifi_ap_config_t

  *This structure is the Wi-Fi configuration for initialization for Soft-AP mode.*
- struct wifi_auto_connect_info_f

  *WiFi auto connect info parameters.*
- union wifi_config_t

  *Wi-Fi configuration for initialization.*
- struct wifi_fast_scan_threshold_t

  *Structure describing parameters for a Wi-Fi fast scan.*
- struct wifi_init_config_t

  *WiFi stack configuration parameters.*
- struct wifi_scan_config_t

  *Parameters for an SSID scan.*
- struct wifi_scan_info_t

  *This structure defines the inforamtion of scanned APs.*
- struct wifi_scan_list_t

  *This structure defines the list of scanned APs with their corresponding information.*
- union wifi_scan_time_t

  *Aggregate of active & passive scan time per channel.*
- struct wifi_sta_config_t

  *This structure is the Wi-Fi configuration for initialization for STA mode.*

### Macros

- #define WIFI_BEACON_INTERVAL_LENGTH (2)

  *Beacon interval length in a frame header.*
- #define WIFI_CAPABILITY_INFO_LENGTH (2)

  *Length of capability information in a frame header.*
- #define WIFI_LENGTH_802_11 (24)

  *Length of 802.11 MAC header.*
- #define WIFI_LENGTH_PASSPHRASE (64)

  *The maximum length of passphrase used in WPA-PSK and WPA2-PSK encryption types.*
- #define WIFI_MAC_ADDRESS_LENGTH (6)

  *MAC address length.*
- #define WIFI_MAX_LENGTH_OF_SSID (32+1)

  *The maximum length of SSID.*
- #define WIFI_MAX_SCAN_AP_NUM (16)

  *maximum number of ap list items which can stored*
- #define WIFI_MAX_SUPPORTED_RATES (8)

  *maximum number of supported rates which can used*

### Typedefs

- typedef int(∗ wifi_event_notify_cb_t) (void ∗data)

### Functions

- int wifi_event_process_handler (wifi_event_t event, uint8_t ∗payload, uint32_t length)

  *Default event handler for system events.*
- void wifi_install_default_event_handlers (void)

  *Set discoverability and connectability mode for legacy bluetooth. This function should.*
- int wifi_register_event_handler (wifi_event_t idx, wifi_event_handler_t handler)

  *Set discoverability and connectability mode for legacy bluetooth. This function should.*

## 4.7.1 Detailed Description

WIFI APIs.

## 4.7.2 Macro Definition Documentation

### 4.7.2.1 WIFI_BEACON_INTERVAL_LENGTH

```
#define WIFI_BEACON_INTERVAL_LENGTH (2)
```

Beacon interval length in a frame header.

### 4.7.2.2 WIFI_CAPABILITY_INFO_LENGTH

```
#define WIFI_CAPABILITY_INFO_LENGTH (2)
```

Length of capability information in a frame header.

### 4.7.2.3 WIFI_LENGTH_802_11

```
#define WIFI_LENGTH_802_11 (24)
```

Length of 802.11 MAC header.

#### 4.7.2.4 WIFI_LENGTH_PASSPHRASE

```
#define WIFI_LENGTH_PASSPHRASE (64)
```

The maximum length of passphrase used in WPA-PSK and WPA2-PSK encryption types.

#### 4.7.2.5 WIFI_MAC_ADDRESS_LENGTH

```
#define WIFI_MAC_ADDRESS_LENGTH (6)
```

MAC address length.

#### 4.7.2.6 WIFI_MAX_LENGTH_OF_SSID

```
#define WIFI_MAX_LENGTH_OF_SSID (32+1)
```

The maximum length of SSID.

#### 4.7.2.7 WIFI_MAX_SCAN_AP_NUM

```
#define WIFI_MAX_SCAN_AP_NUM (16)
```

maximum number of ap list items which can stored

#### 4.7.2.8 WIFI_MAX_SUPPORTED_RATES

```
#define WIFI_MAX_SUPPORTED_RATES (8)
```

maximum number of supported rates which can used

### 4.7.3 Typedef Documentation

#### 4.7.3.1 wifi_event_notify_cb_t

```
typedef int(* wifi_event_notify_cb_t) (void *data)
```

### 4.7.4 Function Documentation

#### 4.7.4.1 wifi_event_process_handler()

```
int wifi_event_process_handler (
            wifi_event_t event,
            uint8_t * payload,
            uint32_t length )
```

Default event handler for system events.

This function performs default handling of system events. When using event_loop APIs, it is called automatically before invoking the user-provided callback function.

Applications which implement a custom event loop must call this function as part of event processing.

**Parameters**

| | | |
|---|---|---|
| in | *event* | event type Set the event type,Options are<br>• WIFI_EVENT_INIT_COMPLETE<br>• WIFI_EVENT_SCAN_COMPLETE<br>• WIFI_EVENT_STA_START<br>• WIFI_EVENT_STA_STOP<br>• WIFI_EVENT_STA_CONNECTED<br>• WIFI_EVENT_STA_DISCONNECTED<br>• WIFI_EVENT_STA_CONNECTION_FAILED<br>• WIFI_EVENT_STA_GOT_IP |
| in | *payload* | Data block that transmitted to event |
| in | *length* | The length of data block |

**Returns**

0 : success
other : failed

#### 4.7.4.2 wifi_install_default_event_handlers()

```
void wifi_install_default_event_handlers (
            void  )
```

Set discoverability and connectability mode for legacy bluetooth. This function should.

#### 4.7.4.3 wifi_register_event_handler()

```
int wifi_register_event_handler (
            wifi_event_t idx,
            wifi_event_handler_t handler )
```

Set discoverability and connectability mode for legacy bluetooth. This function should.

**Parameters**

| | | |
|---|---|---|
| in | *idx* | one of the enums of bt_scan_mode_t |
| in | *handler* | the Wi-Fi event handler |

**Returns**

0 : success
other : failed

## 4.8 WIFI Common APIs

### Data Structures

- struct event_msg_t

  *Send information to event by event_msg_t.*
- union wifi_event_info_t

  *wifi_event_info_t*
- struct wifi_event_sta_connected_t

  *wifi_event_sta_connected_t*
- struct wifi_event_sta_disconnected_t

  *wifi_event_sta_disconnected_t*
- struct wifi_event_sta_got_ip_t

  *wifi_event_sta_got_ip_t*
- struct wifi_event_sta_scan_done_t

  *wifi_event_sta_scan_done_t*

### Typedefs

- typedef int(∗ wifi_event_cb_t) (wifi_event_id_t event, void ∗data, uint16_t length)

  *Application specified event callback function.*

### Functions

- int wifi_event_loop_init (wifi_event_cb_t cb)

  *Event Loop Initialization Create the event handler and call back funtion.*
- int wifi_event_loop_send (event_msg_t ∗msg)

  *Send an event to event task.*
- void wifi_event_loop_set_cb (wifi_event_cb_t cb, void ∗ctx)

  *Set application specified event callback function.*
- int wifi_event_process_handler (wifi_event_t event, uint8_t ∗payload, uint32_t length)

  *Default event handler for system events.*

### 4.8.1 Detailed Description

### 4.8.2 Typedef Documentation

#### 4.8.2.1 wifi_event_cb_t

```
typedef int(* wifi_event_cb_t) (wifi_event_id_t event, void *data, uint16_t length)
```

Application specified event callback function.

### 4.8.3 Function Documentation

#### 4.8.3.1 wifi_event_loop_init()

```
int wifi_event_loop_init (
            wifi_event_cb_t cb )
```

Event Loop Initialization Create the event handler and call back funtion.

**Parameters**

| | |
|---|---|
| *cb* | : application specified event callback |

**Returns**

0 : success
other : failed

#### 4.8.3.2 wifi_event_loop_send()

```
int wifi_event_loop_send (
            event_msg_t * msg )
```

Send an event to event task.

**Attention**

1. Other task/modules, such as the TCPIP module, can call this API to send an event to event task

**Parameters**

| | |
|---|---|
| *event_msg_t* | * msg: Send information to event by msg |

**Returns**

0 : success
other : failed

#### 4.8.3.3 wifi_event_loop_set_cb()

```
void wifi_event_loop_set_cb (
            wifi_event_cb_t cb,
            void * ctx )
```

Set application specified event callback function.

**Attention**

1. If cb is NULL, means application does not need to handle If cb is not NULL, it will be called when an event is received and after the default event callback is completed

**Parameters**

| | |
|---|---|
| *wifi_event_↩ cb_t* | cb : callback |
| *void* | ∗ctx : reserved for user |

#### 4.8.3.4 wifi_event_process_handler()

```
int wifi_event_process_handler (
            wifi_event_t event,
            uint8_t * payload,
            uint32_t length )
```

Default event handler for system events.

This function performs default handling of system events.

Applications which implement a custom event loop must call this function as part of event processing.

**Parameters**

| | | |
|---|---|---|
| in | *event* | event type Set the event type,Options are<br>• WIFI_EVENT_INIT_COMPLETE<br>• WIFI_EVENT_SCAN_COMPLETE<br>• WIFI_EVENT_STA_START<br>• WIFI_EVENT_STA_STOP<br>• WIFI_EVENT_STA_CONNECTED<br>• WIFI_EVENT_STA_DISCONNECTED<br>• WIFI_EVENT_STA_CONNECTION_FAILED<br>• WIFI_EVENT_STA_GOT_IP |
| in | *payload* | Data block transmitted to event |
| in | *length* | The length of the data block |

**Returns**

0 : success
other : failed

## 4.9 WIFI STA APIs

### Typedefs

- typedef int32_t(∗ wifi_event_handler_t) (wifi_event_t event, uint8_t ∗payload, uint32_t length)

  *This defines the Wi-Fi event handler. Call wifi_connection_register_event_handler() to register a handler, then the Wi-Fi driver generates an event and sends it to the handler.*
- typedef void(∗ wifi_init_complete_cb_t) (void ∗ctx)

  *Initialization of complete callback function.*
- typedef int32_t wifi_result_t

### Functions

- int wifi_auto_connect_del_ap_info (u8 index)

  *Delete automatically connected AP information stored in flash.*
- int wifi_auto_connect_get_ap_info (u8 index, wifi_auto_connect_info_f ∗info)

  *Get ap detailed information saved in flash.*
- u8 wifi_auto_connect_get_ap_num (void)

  *Get the number of automatically connected aps that have been saved in the flash.*
- u8 wifi_auto_connect_get_mode (void)

  *Get the status of the current automatic connection mode.*
- int wifi_auto_connect_init (void)

  *Initialize wifi automatic connection.*
- int wifi_auto_connect_set_ap_num (u8 num)

  *Save the number of automatically connected ap to flash.*
- int wifi_auto_connect_set_mode (u8 mode)

  *Set automatic connection mode.*
- int wifi_auto_connect_start (void)

  *Start wifi automatic connection process.*
- int wifi_config_get_bandwidth (wifi_mode_t interface, wifi_bandwidth_t ∗bandwidth)

  *Get the bandwidth of OPL1000 specified interface.*
- int wifi_config_get_bssid (uint8_t ∗bssid)

  *get bssid after scan*
- int wifi_config_get_channel (wifi_mode_t interface, uint8_t ∗channel)

  *Get the primary/secondary channel of OPL1000.*
- int wifi_config_get_mac_address (wifi_mode_t interface, uint8_t ∗address)

  *Get mac of specified interface.*
- int wifi_config_get_ssid (uint8_t ∗ssid, uint8_t ∗ssid_length)

  *Get ssid value of AP.*
- int wifi_config_set_bandwidth (wifi_mode_t interface, wifi_bandwidth_t bandwidth)

  *Set the bandwidth of OPL1000 specified interface.*
- int wifi_config_set_bssid (uint8_t ∗bssid)

  *config OPL1000 Wi-Fi bssid.*
- int wifi_config_set_channel (wifi_mode_t interface, uint8_t channel)

  *Set primary/secondary channel of OPL1000.*
- int wifi_config_set_mac_address (wifi_mode_t interface, uint8_t ∗address)

  *Set MAC address of OPL1000 Wi-Fi station or the soft-AP interface.*
- int wifi_config_set_ssid (wifi_mode_t interface, uint8_t ∗ssid, uint8_t ssid_length)

  *Set the ssid value of the current device.*
- int wifi_connection_connect (wifi_config_t ∗config)

*Connect OPL1000 Wi-Fi station to certain AP.*

- int wifi_connection_disconnect_ap (void)

  *Disconnect the link between OPL1000 and connected AP.*
- int wifi_connection_disconnect_sta (uint8_t *address)

  *Disconnect the link between the current device and the station.*
- int wifi_connection_get_rssi (int8_t *rssi)

  *get signal strength of AP*
- int wifi_connection_register_event_handler (wifi_event_t event, wifi_event_handler_t handler)

  *register wifi call back handler*
- int wifi_connection_unregister_event_handler (wifi_event_t event, wifi_event_handler_t handler)

  *unregister wifi call back handler*
- int wifi_deinit (void)

  *De-init Wi-Fi Initialization and Configuration functions.*
- u8 wifi_fast_connect_get_mode (u8 ap_index)

  *Get the status of AP fast connection.*
- int wifi_fast_connect_set_mode (u8 mode, u8 ap_index)

  *Set the fast connection type.*
- int wifi_fast_connect_start (void)

  *Start the fast connection process.*
- int wifi_get_config (wifi_mode_t interface, wifi_config_t *conf)

  *Get configuration of specified interface.*
- int wifi_get_fast_conn_mode (void)

  *quickly connect to the current AP if the currently scanned AP ID has been connected*
- int wifi_init (const wifi_init_config_t *config, wifi_init_complete_cb_t init_cb)

  *Init Wi-Fi Initializes the wifi according to the specified parameters in the config.*
- int wifi_scan_get_ap_list (wifi_scan_list_t *scan_list)

  *Get list of APs that found in last scan operation.*
- int wifi_scan_get_ap_num (uint16_t *number)

  *Get the number of scanned APs.*
- int wifi_scan_get_ap_records (uint16_t *number, wifi_scan_info_t *ap_records)

  *Get AP list found in last scan operation.*
- int wifi_scan_scan_stop (void)

  *Stop scanning process.*
- int wifi_scan_start (const wifi_scan_config_t *config, bool block)

  *Scan all available APs. After invoke the wifi_set_config() and wifi_start(), then call wifi_scan_start() to scan APs.*
- int wifi_set_config (wifi_mode_t interface, wifi_config_t *conf)

  *Set configuration of OPL1000 STA.*
- int wifi_sta_get_ap_info (wifi_scan_info_t *ap_info)

  *Get information of AP which OPL1000 station is associated with.*
- int wifi_start (void)

  *Start Wi-Fi working.*
- int wifi_stop (void)

  *Stop wifi working.*

### 4.9.1 Detailed Description

### 4.9.2 Typedef Documentation

#### 4.9.2.1 wifi_event_handler_t

```
typedef int32_t(* wifi_event_handler_t) (wifi_event_t event, uint8_t *payload, uint32_t length)
```

This defines the Wi-Fi event handler. Call wifi_connection_register_event_handler() to register a handler, then the Wi-Fi driver generates an event and sends it to the handler.

**Parameters**

| | | |
|---|---|---|
| in | *event* | is an optional event to register. For more details, please refer to wifi_event_t. |
| in | *payload* | is the payload for the event. When the event is WIFI_EVENT_IOT_CONNECTED in AP mode, payload is the connected STA's MAC address. When the event is WIFI_EVENT_IOT_CONNECTED in STA mode, payload is the connected AP's BSSID. |
| in | *length* | is the length of a packet. |

**Returns**

The return value is reserved and it is ignored.

#### 4.9.2.2 wifi_init_complete_cb_t

```
typedef void(* wifi_init_complete_cb_t) (void *ctx)
```

Initialization of complete callback function.

Invoked when Wi-Fi initialization is complete.

**Parameters**

| | |
|---|---|
| *ctx* | is context pointer that provided to wifi_init(). It will be passed back to the callback. |

#### 4.9.2.3 wifi_result_t

```
typedef int32_t wifi_result_t
```

### 4.9.3 Function Documentation

#### 4.9.3.1 wifi_auto_connect_del_ap_info()

```
int wifi_auto_connect_del_ap_info (
            u8 index )
```

Delete automatically connected AP information stored in flash.

**Parameters**

| | | |
|---|---|---|
| in | *index* | : Index of ap information,The range is 0 to 3 |

**Returns**

0 : success
other : failed

#### 4.9.3.2 wifi_auto_connect_get_ap_info()

```
int wifi_auto_connect_get_ap_info (
            u8 index,
            wifi_auto_connect_info_f * info )
```

Get ap detailed information saved in flash.

**Parameters**

| | | |
|---|---|---|
| in | *index* | : Index of ap information,The range is 0 to 3 |
| in | *info* | : wifi_auto_connect_info_f array to hold the found APs |

**Returns**

0 : success
other : failed

#### 4.9.3.3 wifi_auto_connect_get_ap_num()

```
u8 wifi_auto_connect_get_ap_num (
            void  )
```

Get the number of automatically connected aps that have been saved in the flash.

**Returns**

0-3 ap number

#### 4.9.3.4 wifi_auto_connect_get_mode()

```
u8 wifi_auto_connect_get_mode (
            void  )
```

Get the status of the current automatic connection mode.

**Returns**

0 : off
1 : on

#### 4.9.3.5 wifi_auto_connect_init()

```
int wifi_auto_connect_init (
            void  )
```

Initialize wifi automatic connection.

**Returns**

0 : success
other : failed

#### 4.9.3.6 wifi_auto_connect_set_ap_num()

```
int wifi_auto_connect_set_ap_num (
            u8 num )
```

Save the number of automatically connected ap to flash.

**Parameters**

| | | |
|---|---|---|
| in | *Connection* | Type |

**Returns**

0 : success
other : failed

#### 4.9.3.7 wifi_auto_connect_set_mode()

```
int wifi_auto_connect_set_mode (
            u8 mode )
```

Set automatic connection mode.

**Parameters**

| | | |
|---|---|---|
| in | *mode* | Configure the auto connect mode ,0 means disable automatic connection and 1 enable the automatic connection mode |

**Returns**

0 : success
other : failed

#### 4.9.3.8 wifi_auto_connect_start()

```
int wifi_auto_connect_start (
            void  )
```

Start wifi automatic connection process.

**Returns**

0 : success
other : failed

#### 4.9.3.9 wifi_config_get_bandwidth()

```
int wifi_config_get_bandwidth (
            wifi_mode_t interface,
            wifi_bandwidth_t * bandwidth )
```

Get the bandwidth of OPL1000 specified interface.

**Attention**

1. API returns false if try to get an interface which is not enable

**Parameters**

| | | |
|---|---|---|
| in | *interface* | Configure the current wifi working mode,The options are<br>• WIFI_MODE_STA<br>• WIFI_MODE_AP (currently not support) |
| out | *bandwidth* | Get the bandwidth value of the current wifi module working through the pointer |

**Returns**

0 : success
other : failed

#### 4.9.3.10 wifi_config_get_bssid()

```
int wifi_config_get_bssid (
            uint8_t * bssid )
```

get bssid after scan

**Parameters**

| | | |
|---|---|---|
| out | *bssid* | the string of bssid |

**Returns**

0 : success
other : failed

#### 4.9.3.11 wifi_config_get_channel()

```
int wifi_config_get_channel (
            wifi_mode_t interface,
            uint8_t * channel )
```

Get the primary/secondary channel of OPL1000.

**Attention**

1. API returns false if try to get an interface which is not enabled

**Parameters**

| | | |
|---|---|---|
| in | *interface* | Configure the current wifi working mode,The options are<br>• WIFI_MODE_STA<br>• WIFI_MODE_AP (currently not support) |
| out | *channel* | Get Current module wifi work channel number |

**Returns**

0 : success
other : failed

#### 4.9.3.12 wifi_config_get_mac_address()

```
int wifi_config_get_mac_address (
            wifi_mode_t interface,
            uint8_t * address )
```

Get mac of specified interface.

**Parameters**

| | | |
|---|---|---|
| in | *interface* | Configure the current wifi working mode,The options are<br>• WIFI_MODE_STA<br>• WIFI_MODE_AP (currently not support) |
| out | *address* | Get the MAC address of the device through this interface, The address is similar to this structure: xx:xx:xx:xx:xx:xx |

**Returns**

0 : success
other : failed

#### 4.9.3.13 wifi_config_get_ssid()

```
int wifi_config_get_ssid (
            uint8_t * ssid,
            uint8_t * ssid_length )
```

Get ssid value of AP.

**Parameters**

| | | |
|---|---|---|
| out | *ssid* | Get ssid by pointer |
| out | *ssid_length* | Get the length of the ssid character |

**Returns**

0 : success
other : failed

#### 4.9.3.14 wifi_config_set_bandwidth()

```
int wifi_config_set_bandwidth (
            wifi_mode_t interface,
            wifi_bandwidth_t bandwidth )
```

Set the bandwidth of OPL1000 specified interface.

**Parameters**

| | | |
|---|---|---|
| in | *interface* | Configure the current wifi working mode,The options are<br>• WIFI_MODE_STA<br>• WIFI_MODE_AP (currently not support) |
| in | *bandwidth* | Set the working bandwidth of wifi |

**Returns**

0 : success
other : failed

#### 4.9.3.15 wifi_config_set_bssid()

```
int wifi_config_set_bssid (
            uint8_t * bssid )
```

config OPL1000 Wi-Fi bssid.

**Parameters**

| | | |
|---|---|---|
| in | *bssid* | the string of bssid |

**Returns**

0 : success
other : failed

#### 4.9.3.16 wifi_config_set_channel()

```
int wifi_config_set_channel (
            wifi_mode_t interface,
            uint8_t channel )
```

Set primary/secondary channel of OPL1000.

**Attention**

1. This is a special API for sniffer
2. This API should be called after wifi_start()

**Parameters**

| | | |
|---|---|---|
| in | *interface* | Configure the current wifi working mode,The options are<br>• WIFI_MODE_STA<br>• WIFI_MODE_AP (currently not support) |
| in | *channel* | Set current Wi-Fi work channel number |

**Returns**

0 : success
other : failed

#### 4.9.3.17 wifi_config_set_mac_address()

```
int wifi_config_set_mac_address (
            wifi_mode_t interface,
            uint8_t * address )
```

Set MAC address of OPL1000 Wi-Fi station or the soft-AP interface.

**Attention**

1. This API can only be called when the interface is disabled
2. OPL1000 soft-AP and station have different MAC addresses, do not set them to be the same.

**Parameters**

| | | |
|---|---|---|
| in | *interface* | Configure the current wifi working mode,The options are<br>• WIFI_MODE_STA<br>• WIFI_MODE_AP (currently not support) |
| in | *address* | set MAC address |

**Returns**

0 : success
other : failed

#### 4.9.3.18 wifi_config_set_ssid()

```
int wifi_config_set_ssid (
            wifi_mode_t interface,
            uint8_t * ssid,
            uint8_t ssid_length )
```

Set the ssid value of the current device.

**Parameters**

| | | |
|---|---|---|
| in | *interface* | Configure the current wifi working mode,The options are<br>• WIFI_MODE_STA<br>• WIFI_MODE_AP (currently not support) |
| in | *ssid* | Set the value of ssid |
| in | *ssid_length* | The length of ssid parameter |

**Returns**

0 : success
other : failed

#### 4.9.3.19 wifi_connection_connect()

```
int wifi_connection_connect (
            wifi_config_t * config )
```

Connect OPL1000 Wi-Fi station to certain AP.

**Attention**

1. This API only impact WIFI_MODE_STA or WIFI_MODE_AP mode
2. If OPL1000 is connected to an AP, call wifi_disconnect to disconnect.

**Parameters**

| | | |
|---|---|---|
| in | *config* | Establish connection parameters |

**Returns**

0 : success
other : failed

#### 4.9.3.20 wifi_connection_disconnect_ap()

```
int wifi_connection_disconnect_ap (
            void  )
```

Disconnect the link between OPL1000 and connected AP.

**Returns**

0 : success
other : failed

#### 4.9.3.21 wifi_connection_disconnect_sta()

```
int wifi_connection_disconnect_sta (
            uint8_t * address )
```

Disconnect the link between the current device and the station.

**Parameters**

| | | |
|---|---|---|
| in | *address* | station address |

**Returns**

0 : success
other : failed

#### 4.9.3.22 wifi_connection_get_rssi()

```
int wifi_connection_get_rssi (
            int8_t * rssi )
```

get signal strength of AP

**Attention**

1. If the scan is successful, this API returns signal strength value, otherwise it will get wrong result

**Parameters**

| | | |
|---|---|---|
| out | *rssi* | rssi value |

**Returns**

0 : success
other : failed

#### 4.9.3.23 wifi_connection_register_event_handler()

```
int wifi_connection_register_event_handler (
            wifi_event_t event,
            wifi_event_handler_t handler )
```

register wifi call back handler

**Parameters**

| | | |
|---|---|---|
| in | *event* | The type of the registered event. Options are<br>• WIFI_EVENT_INIT_COMPLETE<br>• WIFI_EVENT_SCAN_COMPLETE<br>• WIFI_EVENT_STA_START<br>• WIFI_EVENT_STA_STOP<br>• WIFI_EVENT_STA_CONNECTED<br>• WIFI_EVENT_STA_DISCONNECTED<br>• WIFI_EVENT_STA_CONNECTION_FAILED<br>• WIFI_EVENT_STA_GOT_IP |
| in | *handler* | registered event handler |

**Returns**

0 : success
other : failed

#### 4.9.3.24 wifi_connection_unregister_event_handler()

```
int wifi_connection_unregister_event_handler (
            wifi_event_t event,
            wifi_event_handler_t handler )
```

unregister wifi call back handler

**Parameters**

| | | |
|---|---|---|
| in | *event* | The type of the unregistered event. Options please refer to wifi_connection_register_event_handler() |
| in | *handler* | unregistered event handler |

**Returns**

0 : success
other : failed

#### 4.9.3.25 wifi_deinit()

```
int wifi_deinit (
            void  )
```

De-init Wi-Fi Initialization and Configuration functions.

**Attention**

1. This API should be called if want to remove Wi-Fi driver from the system

**Returns**

0 : success
other : failed

#### 4.9.3.26 wifi_fast_connect_get_mode()

```
u8 wifi_fast_connect_get_mode (
            u8 ap_index )
```

Get the status of AP fast connection.

**Parameters**

| | | |
|---|---|---|
| in | *ap_index* | : Index of ap information,The range is 0 to 3 |

**Returns**

0 : success
other : failed

#### 4.9.3.27 wifi_fast_connect_set_mode()

```
int wifi_fast_connect_set_mode (
            u8 mode,
            u8 ap_index )
```

Set the fast connection type.

**Parameters**

| | | |
|---|---|---|
| in | *mode* | : Configure the fast connect mode ,0 means disable fast connection, and 1 enable the fast connection mode |
| in | *ap_index* | : Index of ap information,The range is 0 to 3 |

**Returns**

0 : success
other : failed

#### 4.9.3.28 wifi_fast_connect_start()

```
int wifi_fast_connect_start (
            void  )
```

Start the fast connection process.

**Returns**

0 : success
other : failed

#### 4.9.3.29 wifi_get_config()

```
int wifi_get_config (
            wifi_mode_t interface,
            wifi_config_t * conf )
```

Get configuration of specified interface.

**Parameters**

| | | |
|---|---|---|
| in | *interface* | Configure wifi working mode,The options are<br>• WIFI_MODE_STA<br>• WIFI_MODE_AP (currently not support) |
| out | *conf* | return wifi's current operating parameters |

**Returns**

0 : success
other : failed

#### 4.9.3.30 wifi_get_fast_conn_mode()

```
int wifi_get_fast_conn_mode (
            void  )
```

quickly connect to the current AP if the currently scanned AP ID has been connected

**Returns**

0 : success
other : failed

#### 4.9.3.31 wifi_init()

```
int wifi_init (
            const wifi_init_config_t * config,
            wifi_init_complete_cb_t init_cb )
```

Init Wi-Fi Initializes the wifi according to the specified parameters in the config.

**Attention**

1. This API must be called before other Wi-Fi APIs are invoked

**Parameters**

| | | |
|---|---|---|
| in | *config* | pointer to Wi-Fi init configuration structure; can point to a temporary variable. |
| in | *init_cb* | pointer to Wi-Fi init complete configuration structure; can point to a temporary variable. |

**Returns**

0 : success
other : failed

#### 4.9.3.32 wifi_scan_get_ap_list()

```
int wifi_scan_get_ap_list (
            wifi_scan_list_t * scan_list )
```

Get list of APs that found in last scan operation.

**Attention**

This API only be called when scan is completed, otherwise it may get wrong value.

**Parameters**

| | | |
|---|---|---|
| out | *scan_list* | store APs' informaton that found in last scan operation |

**Returns**

0 : success
other : failed

#### 4.9.3.33 wifi_scan_get_ap_num()

```
int wifi_scan_get_ap_num (
            uint16_t * number )
```

Get the number of scanned APs.

**Parameters**

| | | |
|---|---|---|
| out | *number* | store number of APs found in last scan operation |

**Attention**

This API only be called when scan is completed, otherwise it may get wrong value.

**Returns**

the scan result of AP number

#### 4.9.3.34 wifi_scan_get_ap_records()

```
int wifi_scan_get_ap_records (
            uint16_t * number,
            wifi_scan_info_t * ap_records )
```

Get AP list found in last scan operation.

**Parameters**

| | | |
|---|---|---|
| out | *number* | As input param, it stores max AP number that ap_records can hold. As output param, it receives the actual AP number that this API returns. |
| out | *ap_records* | wifi_scan_info_t array stores the found APs |

**Returns**

0 : success
other : failed

#### 4.9.3.35 wifi_scan_scan_stop()

```
int wifi_scan_scan_stop (
            void  )
```

Stop scanning process.

**Attention**

This API shall be called after wifi_scan_start()

**Returns**

0 : success
other : failed

#### 4.9.3.36 wifi_scan_start()

```
int wifi_scan_start (
            const wifi_scan_config_t * config,
            bool block )
```

Scan all available APs. After invoke the wifi_set_config() and wifi_start(), then call wifi_scan_start() to scan APs.

**Parameters**

| | | |
|---|---|---|
| in | *config* | Configure parameters for scan operation |
| in | *block* | if block is true, this API blocks the caller until scan operation is done, otherwise it returns immediately |

**Returns**

0 : success
other : failed

#### 4.9.3.37 wifi_set_config()

```
int wifi_set_config (
            wifi_mode_t interface,
            wifi_config_t * conf )
```

Set configuration of OPL1000 STA.

**Attention**

1. This API is called only when specified interface is enabled, otherwise API calling will be failed
2. For station configuration, bssid_set shall be set to 0; set to 1 menas user want to check MAC address of certain AP.
3. OPL1000 is limited to working on one channel.

**Parameters**

| | | |
|---|---|---|
| in | *interface* | Configure wifi working mode,The options are<br>• WIFI_MODE_STA<br>• WIFI_MODE_AP (currently not support) |
| in | *conf* | structure of configuration paremeters |

**Returns**

0 : success
other : failed

#### 4.9.3.38 wifi_sta_get_ap_info()

```
int wifi_sta_get_ap_info (
            wifi_scan_info_t * ap_info )
```

Get information of AP which OPL1000 station is associated with.

**Parameters**

| | | |
|---|---|---|
| out | *ap_info* | get AP information from list |

**Returns**

0 : success
other : failed

#### 4.9.3.39 wifi_start()

```
int wifi_start (
            void  )
```

Start Wi-Fi working.

- If mode is WIFI_MODE_STA, it creates station control block and starts station

**Returns**

0 : success
other : failed

#### 4.9.3.40 wifi_stop()

```
int wifi_stop (
            void  )
```

Stop wifi working.

- If mode is WIFI_MODE_STA, it stops station and releases station control block

**Returns**

0 : success
other : failed

## 4.10 Enumeration

### Enumerations

- enum wifi_auth_mode_t {
  WIFI_AUTH_OPEN = 0, WIFI_AUTH_WEP, WIFI_AUTH_WPA_PSK, WIFI_AUTH_WPA2_PSK,
  WIFI_AUTH_WPA_WPA2_PSK, WIFI_AUTH_WPA2_ENTERPRISE }

  *This enumeration defines the wireless authentication mode to indicate the Wi-Fi device authentication attribute.*
- enum wifi_bandwidth_t { WIFI_BW_HT20 = 1, WIFI_BW_HT40 }
- enum wifi_cipher_type_t {
  WIFI_CIPHER_TYPE_NONE = 0, WIFI_CIPHER_TYPE_WEP40, WIFI_CIPHER_TYPE_WEP104,
  WIFI_CIPHER_TYPE_TKIP,
  WIFI_CIPHER_TYPE_CCMP, WIFI_CIPHER_TYPE_TKIP_CCMP, WIFI_CIPHER_TYPE_UNKNOWN }

  *This enumeration defines wireless security cipher suits.*
- enum wifi_event_t {
  WIFI_EVENT_NONE = -1, WIFI_EVENT_INIT_COMPLETE = 0, WIFI_EVENT_SCAN_COMPLETE,
  WIFI_EVENT_STA_START,
  WIFI_EVENT_STA_STOP, WIFI_EVENT_STA_CONNECTED, WIFI_EVENT_STA_DISCONNECTED,
  WIFI_EVENT_STA_CONNECTION_FAILED,
  WIFI_EVENT_STA_GOT_IP, WIFI_EVENT_MAX }

  *This enumeration defines the supported events generated by the Wi-Fi driver. The event will be sent to the upper layer handler registered in wifi_register_event_handler().*
- enum wifi_mode_t { WIFI_MODE_NULL = 0, WIFI_MODE_STA, WIFI_MODE_AP, WIFI_MODE_MAX }
- enum wifi_reason_code_t {
  WIFI_REASON_CODE_SUCCESS, WIFI_REASON_CODE_FIND_AP_FAIL,
  WIFI_REASON_CODE_PREV_AUTH_INVALID, WIFI_REASON_CODE_DEAUTH_LEAVING_BSS,
  WIFI_REASON_CODE_DISASSOC_INACTIVITY, WIFI_REASON_CODE_DISASSOC_AP_OVERLOAD,
  WIFI_REASON_CODE_CLASS_2_ERR, WIFI_REASON_CODE_CLASS_3_ERR,
  WIFI_REASON_CODE_DISASSOC_LEAVING_BSS, WIFI_REASON_CODE_ASSOC_BEFORE_AUTH,
  WIFI_REASON_CODE_DISASSOC_PWR_CAP_UNACCEPTABLE, WIFI_REASON_CODE_DISASSOC_SUP_CHS_UN
  AC CEPTABLE, WIFI_REASON_CODE_INVALID_INFO_ELEM = 13, WIFI_REASON_CODE_MIC_FAILURE,
  WIFI_REASON_CODE_4_WAY_HANDSHAKE_TIMEOUT, WIFI_REASON_CODE_GROUP_KEY_UPDATE_TIMEOUT,
  WIFI_REASON_CODE_DIFFERENT_INFO_ELEM, WIFI_REASON_CODE_GROUP_CIPHER_INVALID_VALID,
  WIFI_REASON_CODE_PAIRWISE_CIPHER_INVALID, WIFI_REASON_CODE_AKMP_INVALID,
  WIFI_REASON_CODE_UNSUPPORTED_RSNE_VERSION, WIFI_REASON_CODE_INVALID_RSNE_CAPABILITIES,
  WIFI_REASON_CODE_IEEE_802_1X_AUTH_FAILED, WIFI_REASON_CODE_CIPHER_REJECTED,
  WIFI_REASON_CODE_AUTO_CONNECT_FAILED = 200 }

  *This enumeration defines the reason code of the WIFI_EVENT_STA_CONNECTION_FAILED event in wifi_event_t. Find the details for the reason code below.*
- enum wifi_scan_method_t { WIFI_FAST_SCAN = 0, WIFI_ALL_CHANNEL_SCAN }
- enum wifi_scan_type_t { WIFI_SCAN_TYPE_ACTIVE = 0, WIFI_SCAN_TYPE_PASSIVE }

  *This enumeration defines the wireless STA scan type.*
- enum wifi_sort_method_t { WIFI_CONNECT_AP_BY_SIGNAL = 0, WIFI_CONNECT_AP_BY_SECURITY }

### 4.10.1 Detailed Description

### 4.10.2 Enumeration Type Documentation

#### 4.10.2.1 wifi_auth_mode_t

`enum wifi_auth_mode_t`

This enumeration defines the wireless authentication mode to indicate the Wi-Fi device authentication attribute.

**Enumerator**

| | |
|---|---|
| WIFI_AUTH_OPEN | authenticate mode : open |
| WIFI_AUTH_WEP | authenticate mode : WEP |
| WIFI_AUTH_WPA_PSK | authenticate mode : WPA_PSK |
| WIFI_AUTH_WPA2_PSK | authenticate mode : WPA2_PSK |
| WIFI_AUTH_WPA_WPA2_PSK | authenticate mode : WPA_WPA2_PSK |
| WIFI_AUTH_WPA2_ENTERPRISE | authenticate mode : WPA2_ENTERPRISE |

#### 4.10.2.2 wifi_bandwidth_t

`enum wifi_bandwidth_t`

**Enumerator**

| | |
|---|---|
| WIFI_BW_HT20 | Bandwidth is HT20 |
| WIFI_BW_HT40 | Bandwidth is HT40 |

#### 4.10.2.3 wifi_cipher_type_t

`enum wifi_cipher_type_t`

This enumeration defines wireless security cipher suits.

**Enumerator**

| | |
|---|---|
| WIFI_CIPHER_TYPE_NONE | 0, the cipher type is none |
| WIFI_CIPHER_TYPE_WEP40 | 1, the cipher type is WEP40 |
| WIFI_CIPHER_TYPE_WEP104 | 2, the cipher type is WEP104 |
| WIFI_CIPHER_TYPE_TKIP | 3, the cipher type is TKIP |
| WIFI_CIPHER_TYPE_CCMP | 4, the cipher type is CCMP |
| WIFI_CIPHER_TYPE_TKIP_CCMP | 5, the cipher type is TKIP and CCMP |
| WIFI_CIPHER_TYPE_UNKNOWN | 6, the cipher type is unknown |

#### 4.10.2.4 wifi_event_t

`enum wifi_event_t`

This enumeration defines the supported events generated by the Wi-Fi driver. The event will be sent to the upper layer handler registered in wifi_register_event_handler().

**Enumerator**

| | |
|---|---|
| WIFI_EVENT_NONE | Reserved |
| WIFI_EVENT_INIT_COMPLETE | Wi-Fi initialization complete event. |
| WIFI_EVENT_SCAN_COMPLETE | Scan completed event |
| WIFI_EVENT_STA_START | station start |
| WIFI_EVENT_STA_STOP | station stop |
| WIFI_EVENT_STA_CONNECTED | station connected to AP event |
| WIFI_EVENT_STA_DISCONNECTED | station disconnected from AP |
| WIFI_EVENT_STA_CONNECTION_FAILED | Connection has failed. For the reason code, please refer to wifi_reason_code_t. |
| WIFI_EVENT_STA_GOT_IP | station got IP from connected AP |
| WIFI_EVENT_MAX | |

#### 4.10.2.5 wifi_mode_t

`enum wifi_mode_t`

**Enumerator**

| | |
|---|---|
| WIFI_MODE_NULL | null mode |
| WIFI_MODE_STA | Wi-Fi station mode |
| WIFI_MODE_AP | Wi-Fi soft-AP mode |
| WIFI_MODE_MAX | |

#### 4.10.2.6 wifi_reason_code_t

`enum wifi_reason_code_t`

This enumeration defines the reason code of the WIFI_EVENT_STA_CONNECTION_FAILED event in wifi_event_t. Find the details for the reason code below.

**Enumerator**

| | |
|---|---|
| WIFI_REASON_CODE_SUCCESS | 0 Reserved. |
| WIFI_REASON_CODE_FIND_AP_FAIL | 1 (Internal) No AP found. |
| WIFI_REASON_CODE_PREV_AUTH_INVALID | 2 Previous authentication is no longer valid. |
| WIFI_REASON_CODE_DEAUTH_LEAVING_BSS | 3 Deauthenticated because sending STA is leaving (or has left) IBSS or ES. |
| WIFI_REASON_CODE_DISASSOC_INACTIVITY | 4 Disassociated due to inactivity. |
| WIFI_REASON_CODE_DISASSOC_AP_OVERL↩ OAD | 5 Disassociated because AP is unable to handle all currently associated STAs. |
| WIFI_REASON_CODE_CLASS_2_ERR | 6 Class 2 frame received from nonauthenticated STA. |
| WIFI_REASON_CODE_CLASS_3_ERR | 7 Class 3 frame received from nonauthenticated STA. |

**Enumerator**

| | |
|---|---|
| WIFI_REASON_CODE_DISASSOC_LEAVING_BSS | 8 Disassociated because sending STA is leaving (or has left) BSS. |
| WIFI_REASON_CODE_ASSOC_BEFORE_AUTH | 9 STA requesting (re)association is not authenticated with responding STA. |
| WIFI_REASON_CODE_DISASSOC_PWR_CAP_↩ UNACCEPTABLE | 10 Disassociated because the information in the Power Capability element is unacceptable. |
| WIFI_REASON_CODE_DISASSOC_SUP_CHS_U↩ NACCEPTABLE | 11 Disassociated because the information in the Supported Channels element is unacceptable. |
| WIFI_REASON_CODE_INVALID_INFO_ELEM | 13 Invalid information element. |
| WIFI_REASON_CODE_MIC_FAILURE | 14 Message integrity code (MIC) failure. |
| WIFI_REASON_CODE_4_WAY_HANDSHAKE_TI↩ MEOUT | 15 4-Way Handshake time out. |
| WIFI_REASON_CODE_GROUP_KEY_UPDATE_↩ TIMEOUT | 16 Group Key Handshake time out. |
| WIFI_REASON_CODE_DIFFERENT_INFO_ELEM | 17 Information element in 4-Way Handshake different from (Re)Association Request/Probe Response/Beacon frame. |
| WIFI_REASON_CODE_GROUP_CIPHER_INVALI↩ D_VALID | 18 Invalid group cipher. |
| WIFI_REASON_CODE_PAIRWISE_CIPHER_INV↩ ALID | 19 Invalid pairwise cipher. |
| WIFI_REASON_CODE_AKMP_INVALID | 20 Invalid AKMP. |
| WIFI_REASON_CODE_UNSUPPORTED_RSNE_↩ VERSION | 21 Unsupported RSN information element version. |
| WIFI_REASON_CODE_INVALID_RSNE_CAPABI↩ LITIES | 22 Invalid RSN information element capabilities. |
| WIFI_REASON_CODE_IEEE_802_1X_AUTH_FAI↩ LED | 23 IEEE 802.1X authentication failed. |
| WIFI_REASON_CODE_CIPHER_REJECTED | 24 Cipher suite rejected because of the security policy. |
| WIFI_REASON_CODE_AUTO_CONNECT_FAILED | 200 Auto connect failed. |

#### 4.10.2.7 wifi_scan_method_t

```
enum wifi_scan_method_t
```

**Enumerator**

| | |
|---|---|
| WIFI_FAST_SCAN | Do fast scan, scan will end after find SSID match AP |
| WIFI_ALL_CHANNEL_SCAN | All channel scan, scan will end after scan all the channel |

#### 4.10.2.8 wifi_scan_type_t

```
enum wifi_scan_type_t
```

This enumeration defines the wireless STA scan type.

**Enumerator**

| | |
|---|---|
| WIFI_SCAN_TYPE_ACTIVE | Actively scan a network by sending 802.11 probe(s) |
| WIFI_SCAN_TYPE_PASSIVE | Passively scan a network by listening for beacons from APs |

#### 4.10.2.9 wifi_sort_method_t

```
enum wifi_sort_method_t
```

**Enumerator**

| | |
|---|---|
| WIFI_CONNECT_AP_BY_SIGNAL | Sort match AP in scan list by RSSI |
| WIFI_CONNECT_AP_BY_SECURITY | Sort match AP in scan list by security mode |

# Chapter 5

# Data Structure Documentation

## 5.1 auto_conn_info_t Struct Reference

```
#include <controller_wifi_com_patch.h>
```

### Data Fields

- u8 ap_channel
- u16 beacon_interval
- u8 bssid [MAC_ADDR_LEN]
- u16 capabilities
- u8 dtim_prod
- u8 fast_connect
- bool free_ocpy
- s8 hid_ssid [IEEE80211_MAX_SSID_LEN+1]
- u64 latest_beacon_rx_time
- s8 passphrase [MAX_LEN_OF_PASSPHRASE]
- u8 psk [32]
- u8 rsn_ie [100]
- s8 rssi
- s8 ssid [IEEE80211_MAX_SSID_LEN+1]
- u8 supported_rates [SUPPORTED_RATES_MAX]
- wpa_ie_data_t wpa_data
- u8 wpa_ie [100]

### 5.1.1 Field Documentation

#### 5.1.1.1 ap_channel

```
u8 ap_channel
```

#### 5.1.1.2 beacon_interval

```
u16 beacon_interval
```

#### 5.1.1.3 bssid

```
u8 bssid[MAC_ADDR_LEN]
```

#### 5.1.1.4 capabilities

```
u16 capabilities
```

#### 5.1.1.5 dtim_prod

```
u8 dtim_prod
```

#### 5.1.1.6 fast_connect

```
u8 fast_connect
```

#### 5.1.1.7 free_ocpy

```
bool free_ocpy
```

#### 5.1.1.8 hid_ssid

```
s8 hid_ssid[IEEE80211_MAX_SSID_LEN+1]
```

#### 5.1.1.9 latest_beacon_rx_time

```
u64 latest_beacon_rx_time
```

#### 5.1.1.10 passphrase

```
s8 passphrase[MAX_LEN_OF_PASSPHRASE]
```

#### 5.1.1.11 psk

```
u8 psk[32]
```

#### 5.1.1.12 rsn_ie

```
u8 rsn_ie[100]
```

#### 5.1.1.13 rssi

```
s8 rssi
```

#### 5.1.1.14 ssid

```
s8 ssid[IEEE80211_MAX_SSID_LEN+1]
```

#### 5.1.1.15 supported_rates

```
u8 supported_rates[SUPPORTED_RATES_MAX]
```

#### 5.1.1.16 wpa_data

```
wpa_ie_data_t wpa_data
```

#### 5.1.1.17 wpa_ie

```
u8 wpa_ie[100]
```

## 5.2 auto_connect_cfg_t Struct Reference

`#include <controller_wifi_com_patch.h>`

### Data Fields

- bool flag
- s8 front
- u8 max_save_num
- auto_conn_info_t * pFCInfo
- s8 rear
- u8 retryCount
- u8 targetIdx
- u32 uFCApNum

### 5.2.1 Field Documentation

#### 5.2.1.1 flag

```
bool flag
```

#### 5.2.1.2 front

```
s8 front
```

#### 5.2.1.3 max_save_num

```
u8 max_save_num
```

#### 5.2.1.4 pFCInfo

```
auto_conn_info_t* pFCInfo
```

#### 5.2.1.5 rear

```
s8 rear
```

#### 5.2.1.6 retryCount

```
u8 retryCount
```

#### 5.2.1.7 targetIdx

```
u8 targetIdx
```

#### 5.2.1.8 uFCApNum

```
u32 uFCApNum
```

## 5.3 event_msg_t Struct Reference

Send information to event by event_msg_t.

```
#include <event_loop.h>
```

### Data Fields

- uint32_t event
- uint32_t length
- uint8_t * param

### 5.3.1 Detailed Description

Send information to event by event_msg_t.

### 5.3.2 Field Documentation

#### 5.3.2.1 event

```
uint32_t event
```

event type

#### 5.3.2.2 length

```
uint32_t length
```

Packet length

#### 5.3.2.3 param

```
uint8_t* param
```

event parament

## 5.4 LE_BT_ADDR_T Struct Reference

```
#include <ble.h>
```

### Data Fields

- BD_ADDR addr
- UINT8 type

### 5.4.1 Field Documentation

#### 5.4.1.1 addr

```
BD_ADDR addr
```

address

#### 5.4.1.2 type

```
UINT8 type
```

address type

## 5.5 LE_CM_CONNECTION_COMPLETE_IND_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 conn_interval
- UINT16 conn_latency
- UINT16 dev_id
- BD_ADDR peer_addr
- UINT8 peer_addr_type
- UINT8 role
- UINT16 status
- UINT16 supervison_timeout

### 5.5.1 Field Documentation

#### 5.5.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.5.1.2 conn_interval

```
UINT16 conn_interval
```

connection interval

#### 5.5.1.3 conn_latency

```
UINT16 conn_latency
```

connection latency

#### 5.5.1.4 dev_id

```
UINT16 dev_id
```

device ID

#### 5.5.1.5 peer_addr

```
BD_ADDR peer_addr
```

perr address

#### 5.5.1.6 peer_addr_type

```
UINT8 peer_addr_type
```

peer address type

#### 5.5.1.7 role

```
UINT8 role
```

master or slave

#### 5.5.1.8 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

#### 5.5.1.9 supervison_timeout

```
UINT16 supervison_timeout
```

supervision timeout

## 5.6 LE_CM_MSG_ADVERTISE_REPORT_IND_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- BD_ADDR addr
- UINT8 addr_type
- UINT8 data [1]
- UINT8 event_type
- UINT8 len
- INT8 rssi

### 5.6.1 Field Documentation

#### 5.6.1.1 addr

```
BD_ADDR addr
```

address

#### 5.6.1.2 addr_type

```
UINT8 addr_type
```

address type

#### 5.6.1.3 data

```
UINT8 data[1]
```

#### 5.6.1.4 event_type

```
UINT8 event_type
```

#### 5.6.1.5 len

```
UINT8 len
```

#### 5.6.1.6 rssi

```
INT8 rssi
```

RSSI

## 5.7 LE_CM_MSG_CONN_PARA_REQ_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 itv_max
- UINT16 itv_min
- UINT16 latency
- UINT32 sv_tmo

### 5.7.1 Field Documentation

#### 5.7.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.7.1.2 itv_max

```
UINT16 itv_max
```

maxinum connection interval

#### 5.7.1.3 itv_min

```
UINT16 itv_min
```

mininum connection interval

#### 5.7.1.4 latency

```
UINT16 latency
```

slave latency

#### 5.7.1.5 sv_tmo

```
UINT32 sv_tmo
```

supervision timeout

## 5.8 LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 interval
- UINT16 latency
- UINT16 status
- UINT32 supervision_timeout

### 5.8.1 Field Documentation

#### 5.8.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.8.1.2 interval

```
UINT16 interval
```

connection interval

#### 5.8.1.3 latency

```
UINT16 latency
```

slave letency

#### 5.8.1.4 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

#### 5.8.1.5 supervision_timeout

```
UINT32 supervision_timeout
```

supervision timeout

## 5.9 LE_CM_MSG_DATA_LEN_CHANGE_IND_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 max_rx_octets
- UINT16 max_rx_time
- UINT16 max_tx_octets
- UINT16 max_tx_time

### 5.9.1 Field Documentation

#### 5.9.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.9.1.2 max_rx_octets

```
UINT16 max_rx_octets
```

connMaxRxOctets

#### 5.9.1.3 max_rx_time

```
UINT16 max_rx_time
```

connMaxRxTime

#### 5.9.1.4 max_tx_octets

```
UINT16 max_tx_octets
```

connMaxTxOctets

#### 5.9.1.5 max_tx_time

```
UINT16 max_tx_time
```

connMaxTxTime

## 5.10 LE_CM_MSG_DIRECT_ADV_REPORT_IND_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- BD_ADDR direct_addr
- UINT8 direct_addr_type
- BD_ADDR peer_addr
- UINT8 peer_addr_type
- INT8 rssi

### 5.10.1 Field Documentation

#### 5.10.1.1 direct_addr

```
BD_ADDR direct_addr
```

direct address

#### 5.10.1.2 direct_addr_type

```
UINT8 direct_addr_type
```

direct address type

#### 5.10.1.3 peer_addr

```
BD_ADDR peer_addr
```

peer address

#### 5.10.1.4 peer_addr_type

```
UINT8 peer_addr_type
```

peer address type

#### 5.10.1.5 rssi

```
INT8 rssi
```

RSSI

## 5.11 LE_CM_MSG_DISCONNECT_COMPLETE_IND_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT8 reason
- UINT16 status

### 5.11.1 Field Documentation

#### 5.11.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.11.1.2 reason

```
UINT8 reason
```

disconnect reason

#### 5.11.1.3 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.12 LE_CM_MSG_ENCRYPTION_CHANGE_IND_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 devid
- UINT8 enabled
- UINT16 status

### 5.12.1 Field Documentation

#### 5.12.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.12.1.2 devid

```
UINT16 devid
```

device ID

#### 5.12.1.3 enabled

```
UINT8 enabled
```

#### 5.12.1.4 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.13 LE_CM_MSG_ENCRYPTION_REFRESH_IND_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 devid
- BOOL enabled
- UINT16 status

### 5.13.1 Field Documentation

#### 5.13.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.13.1.2 devid

```
UINT16 devid
```

device ID

#### 5.13.1.3 enabled

```
BOOL enabled
```

enable or disable

#### 5.13.1.4 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.14 LE_CM_MSG_INIT_COMPLETE_CFM_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- UINT16 status

### 5.14.1 Field Documentation

#### 5.14.1.1 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.15 LE_CM_MSG_LTK_REQ_IND_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 devid
- UINT16 ediv
- UINT8 rand [8]

### 5.15.1 Field Documentation

#### 5.15.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.15.1.2 devid

```
UINT16 devid
```

device ID

#### 5.15.1.3 ediv

```
UINT16 ediv
```

#### 5.15.1.4 rand

```
UINT8 rand[8]
```

## 5.16 LE_CM_MSG_READ_ADV_TX_POWER_CFM_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- INT8 pwr_level
- UINT16 status

### 5.16.1 Field Documentation

#### 5.16.1.1 pwr_level

```
INT8 pwr_level
```

power level

#### 5.16.1.2 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.17 LE_CM_MSG_READ_BD_ADDR_CFM_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- BD_ADDR bd_addr
- UINT16 status

### 5.17.1 Field Documentation

#### 5.17.1.1 bd_addr

```
BD_ADDR bd_addr
```

#### 5.17.1.2 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.18 LE_CM_MSG_READ_CHANNEL_MAP_CFM_T Struct Reference

`#include <ble_cm_if.h>`

### Data Fields

- UINT8 ch_map [5]
- UINT16 conn_hdl
- UINT16 status

### 5.18.1 Field Documentation

#### 5.18.1.1 ch_map

`UINT8 ch_map[5]`

channel map

#### 5.18.1.2 conn_hdl

`UINT16 conn_hdl`

connection handle

#### 5.18.1.3 status

`UINT16 status`

refer to LE_ERR_STATE in ble_err.h

## 5.19 LE_CM_MSG_READ_RESOLVING_LIST_SIZE_CFM_T Struct Reference

`#include <ble_cm_if.h>`

### Data Fields

- UINT8 size
- UINT16 status

### 5.19.1 Field Documentation

#### 5.19.1.1 size

```
UINT8 size
```

resolving list size

#### 5.19.1.2 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.20 LE_CM_MSG_READ_RSSI_CFM_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- UINT16 conn_hdl
- INT8 rssi
- UINT16 status

### 5.20.1 Field Documentation

#### 5.20.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.20.1.2 rssi

```
INT8 rssi
```

RSSI

#### 5.20.1.3 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.21 LE_CM_MSG_READ_TX_POWER_CFM_T Struct Reference

`#include <ble_cm_if.h>`

### Data Fields

- UINT16 conn_hdl
- UINT16 status
- INT8 tx_power

### 5.21.1 Field Documentation

#### 5.21.1.1 conn_hdl

`UINT16 conn_hdl`

connection handle

#### 5.21.1.2 status

`UINT16 status`

refer to LE_ERR_STATE in ble_err.h

#### 5.21.1.3 tx_power

`INT8 tx_power`

tx power

## 5.22 LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM_T Struct Reference

`#include <ble_cm_if.h>`

### Data Fields

- UINT8 size
- UINT16 status

### 5.22.1 Field Documentation

#### 5.22.1.1 size

```
UINT8 size
```

white list size

#### 5.22.1.2 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.23 LE_CM_MSG_SET_DATA_LENGTH_CFM_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 status

### 5.23.1 Field Documentation

#### 5.23.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.23.1.2 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.24 LE_CM_MSG_SET_DISCONNECT_CFM_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- UINT16 handle
- UINT16 status

### 5.24.1 Field Documentation

#### 5.24.1.1 handle

```
UINT16 handle
```

connection handle

#### 5.24.1.2 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.25 LE_CM_MSG_SIGNAL_UPDATE_REQ_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 identifier
- UINT16 interval_max
- UINT16 interval_min
- UINT16 slave_latency
- UINT32 timeout_multiplier

### 5.25.1 Field Documentation

#### 5.25.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.25.1.2 identifier

```
UINT16 identifier
```

#### 5.25.1.3 interval_max

```
UINT16 interval_max
```

maxinum connection interval

#### 5.25.1.4 interval_min

```
UINT16 interval_min
```

mininum connection interval

#### 5.25.1.5 slave_latency

```
UINT16 slave_latency
```

slave latency

#### 5.25.1.6 timeout_multiplier

```
UINT32 timeout_multiplier
```

## 5.26 LE_CM_REQ_STATUS_T Struct Reference

```
#include <ble_cm_if.h>
```

### Data Fields

- UINT16 status

### 5.26.1 Field Documentation

#### 5.26.1.1 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.27 LE_CONN_PARA_T Struct Reference

```
#include <ble.h>
```

### Data Fields

- UINT16 itv_max
- UINT16 itv_min
- UINT16 latency
- UINT16 sv_timeout

### 5.27.1 Field Documentation

#### 5.27.1.1 itv_max

```
UINT16 itv_max
```

maxinum connection interval

#### 5.27.1.2 itv_min

```
UINT16 itv_min
```

mininum connection interval

#### 5.27.1.3 latency

```
UINT16 latency
```

slave latency

#### 5.27.1.4 sv_timeout

```
UINT16 sv_timeout
```

supervision timeout

## 5.28 LE_GAP_ADVERTISING_PARAM_T Struct Reference

`#include <ble_gap_if.h>`

### Data Fields

- UINT8 channel_map
- UINT8 filter_policy
- UINT16 interval_max
- UINT16 interval_min
- UINT8 own_addr_type
- BD_ADDR peer_addr
- UINT8 peer_addr_type
- UINT8 type

### 5.28.1 Field Documentation

#### 5.28.1.1 channel_map

`UINT8 channel_map`

advertising channel map

#### 5.28.1.2 filter_policy

`UINT8 filter_policy`

advertising filter policy

#### 5.28.1.3 interval_max

`UINT16 interval_max`

maxinum advertising interval

#### 5.28.1.4 interval_min

`UINT16 interval_min`

mininum advertising interval

#### 5.28.1.5 own_addr_type

```
UINT8 own_addr_type
```

owner address type

#### 5.28.1.6 peer_addr

```
BD_ADDR peer_addr
```

peer address

#### 5.28.1.7 peer_addr_type

```
UINT8 peer_addr_type
```

peer address type

#### 5.28.1.8 type

```
UINT8 type
```

advertising type

## 5.29 LE_GAP_CONN_PARAM_T Struct Reference

```
#include <ble_gap_if.h>
```

### Data Fields

- UINT16 interval_max
- UINT16 interval_min
- UINT16 latency
- UINT16 supervision_timeout

### 5.29.1 Field Documentation

#### 5.29.1.1 interval_max

```
UINT16 interval_max
```

maxinum connection interval

#### 5.29.1.2 interval_min

```
UINT16 interval_min
```

mininum connection interval

#### 5.29.1.3 latency

```
UINT16 latency
```

slave latency

#### 5.29.1.4 supervision_timeout

```
UINT16 supervision_timeout
```

supervision timeout for the LE Link

## 5.30 LE_GAP_SCAN_PARAM_T Struct Reference

```
#include <ble_gap_if.h>
```

### Data Fields

- UINT8 filter_policy
- UINT16 interval
- UINT8 own_addr_type
- UINT8 type
- UINT16 window

### 5.30.1 Field Documentation

#### 5.30.1.1 filter_policy

```
UINT8 filter_policy
```

scan filter policy

#### 5.30.1.2 interval

```
UINT16 interval
```

scan interval

#### 5.30.1.3 own_addr_type

```
UINT8 own_addr_type
```

owner address type

#### 5.30.1.4 type

```
UINT8 type
```

scan type

#### 5.30.1.5 window

```
UINT16 window
```

scan window

## 5.31 LE_GATT_ATTR_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT8 format
- UINT16 handle
- UINT16 len
- UINT16 maxLen
- UINT16 permit
- UINT16 *const pUuid
- UINT8 *const pVal

### 5.31.1 Field Documentation

#### 5.31.1.1 format

```
UINT8 format
```

UUID type

#### 5.31.1.2 handle

```
UINT16 handle
```

handle

#### 5.31.1.3 len

```
UINT16 len
```

value length

#### 5.31.1.4 maxLen

```
UINT16 maxLen
```

maxinum value length

#### 5.31.1.5 permit

```
UINT16 permit
```

permit

#### 5.31.1.6 pUuid

```
UINT16* const pUuid
```

UUID

#### 5.31.1.7 pVal

```
UINT8* const pVal
```

value

## 5.32 LE_GATT_MSG_ACCESS_READ_IND_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 offset

### 5.32.1 Field Documentation

#### 5.32.1.1 conn_hdl

`UINT16 conn_hdl`

connection handle

#### 5.32.1.2 devid

`UINT16 devid`

device index

#### 5.32.1.3 handle

`UINT16 handle`

attribute handle

#### 5.32.1.4 offset

`UINT16 offset`

attribute handle value

## 5.33 LE_GATT_MSG_ACCESS_WRITE_IND_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 devid
- UINT8 flag
- UINT16 handle
- UINT16 len
- UINT16 offset
- UINT8 * pVal

### 5.33.1 Field Documentation

#### 5.33.1.1 conn_hdl

`UINT16 conn_hdl`

connection handle

#### 5.33.1.2 devid

`UINT16 devid`

device ID

#### 5.33.1.3 flag

`UINT8 flag`

refer to LE_GATT_FLAG_* in ble_gatt_if.h

#### 5.33.1.4 handle

`UINT16 handle`

attribute handle

#### 5.33.1.5 len

`UINT16 len`

length written

#### 5.33.1.6 offset

`UINT16 offset`

attribute handle value

#### 5.33.1.7 pVal

`UINT8* pVal`

value written

## 5.34 LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND_T Struct Reference

`#include <ble_gatt_if.h>`

### Data Fields

- UINT16 conn_hdl
- UINT16 devid
- UINT8 format
- UINT16 handle
- UINT16 uuid [8]

### 5.34.1 Field Documentation

#### 5.34.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.34.1.2 devid

```
UINT16 devid
```

device ID

#### 5.34.1.3 format

```
UINT8 format
```

UUID type

#### 5.34.1.4 handle

```
UINT16 handle
```

characteristic descriptor handle

#### 5.34.1.5 uuid

```
UINT16 uuid[8]
```

UUID

## 5.35 LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 devid
- UINT8 format
- UINT16 handle
- UINT8 property
- UINT16 uuid [8]
- UINT16 val_hdl

## 5.35.1 Field Documentation

#### 5.35.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.35.1.2 devid

```
UINT16 devid
```

device ID

#### 5.35.1.3 format

```
UINT8 format
```

UUID type

#### 5.35.1.4 handle

```
UINT16 handle
```

characteristic declaration handle

#### 5.35.1.5 property

```
UINT8 property
```

property

#### 5.35.1.6 uuid

```
UINT16 uuid[8]
```

UUID

#### 5.35.1.7 val_hdl

```
UINT16 val_hdl
```

characteristic value handle

## 5.36 LE_GATT_MSG_CHARACTERISTIC_VAL_IND_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT8 att_err
- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 len
- UINT16 offset
- UINT8 * val

### 5.36.1 Field Documentation

#### 5.36.1.1 att_err

```
UINT8 att_err
```

0 is ok, others refer to LE_ATT_ERR_* in ble_att_if.h

#### 5.36.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.36.1.3 devid

```
UINT16 devid
```

device ID

#### 5.36.1.4 handle

```
UINT16 handle
```

characteristic value handle

#### 5.36.1.5 len

```
UINT16 len
```

value length

#### 5.36.1.6 offset

```
UINT16 offset
```

value position offset

#### 5.36.1.7 val

```
UINT8* val
```

value

## 5.37 LE_GATT_MSG_CONFIRMATION_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle

### 5.37.1 Field Documentation

#### 5.37.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.37.1.2 devid

```
UINT16 devid
```

device ID

#### 5.37.1.3 handle

```
UINT16 handle
```

attribute handle

## 5.38 LE_GATT_MSG_EXCHANGE_MTU_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 current_rx_mtu
- UINT16 devid

### 5.38.1 Field Documentation

#### 5.38.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.38.1.2 current_rx_mtu

```
UINT16 current_rx_mtu
```

current receive MTU

#### 5.38.1.3 devid

```
UINT16 devid
```

device ID

## 5.39 LE_GATT_MSG_EXCHANGE_MTU_IND_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT16 client_rx_mtu
- UINT16 conn_hdl
- UINT16 devid

### 5.39.1 Field Documentation

#### 5.39.1.1 client_rx_mtu

```
UINT16 client_rx_mtu
```

client receive MTU

#### 5.39.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.39.1.3 devid

```
UINT16 devid
```

device ID

## 5.40 LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT8 att_err
- UINT16 conn_hdl
- UINT16 devid
- UINT16 err_hdl
- UINT16 status

### 5.40.1 Field Documentation

#### 5.40.1.1 att_err

```
UINT8 att_err
```

0 is ok, others refer to LE_ATT_ERR_∗ in ble_att_if.h

#### 5.40.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.40.1.3 devid

```
UINT16 devid
```

device ID

#### 5.40.1.4 err_hdl

```
UINT16 err_hdl
```

TBD

#### 5.40.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.41 LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT8 att_err
- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 status

### 5.41.1 Field Documentation

#### 5.41.1.1 att_err

```
UINT8 att_err
```

0 is ok, others refer to LE_ATT_ERR_∗ in ble_att_if.h

#### 5.41.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.41.1.3 devid

```
UINT16 devid
```

device ID

#### 5.41.1.4 handle

```
UINT16 handle
```

characteristic descriptor handle

#### 5.41.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.42 LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT8 att_err
- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 status

### 5.42.1 Field Documentation

#### 5.42.1.1 att_err

```
UINT8 att_err
```

0 is ok, others refer to LE_ATT_ERR_∗ in ble_att_if.h

#### 5.42.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.42.1.3 devid

```
UINT16 devid
```

device ID

#### 5.42.1.4 handle

```
UINT16 handle
```

#### 5.42.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.43 LE_GATT_MSG_FIND_CHARACTERISTIC_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT8 att_err
- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 status

### 5.43.1 Field Documentation

#### 5.43.1.1 att_err

```
UINT8 att_err
```

0 is ok, others refer to LE_ATT_ERR_∗ in ble_att_if.h

#### 5.43.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.43.1.3 devid

```
UINT16 devid
```

device ID

#### 5.43.1.4 handle

```
UINT16 handle
```

characteristic descriptor handle

#### 5.43.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.44 LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT8 att_err
- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 status

### 5.44.1 Field Documentation

#### 5.44.1.1 att_err

```
UINT8 att_err
```

0 is ok, others refer to LE_ATT_ERR_∗ in ble_att_if.h

#### 5.44.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.44.1.3 devid

```
UINT16 devid
```

device ID

#### 5.44.1.4 handle

```
UINT16 handle
```

include service start handle

#### 5.44.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.45 LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT8 att_err
- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 status

### 5.45.1 Field Documentation

#### 5.45.1.1 att_err

```
UINT8 att_err
```

0 is ok, others refer to LE_ATT_ERR_∗ in ble_att_if.h

#### 5.45.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.45.1.3 devid

```
UINT16 devid
```

device ID

#### 5.45.1.4 handle

```
UINT16 handle
```

service start handle

#### 5.45.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.46 LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 devid
- UINT16 end_hdl
- UINT8 format
- UINT16 handle
- UINT16 start_hdl
- UINT16 uuid [8]

## 5.46.1 Field Documentation

#### 5.46.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.46.1.2 devid

```
UINT16 devid
```

device ID

#### 5.46.1.3 end_hdl

```
UINT16 end_hdl
```

end handle

#### 5.46.1.4 format

```
UINT8 format
```

UUID type

#### 5.46.1.5 handle

```
UINT16 handle
```

include servie handle

#### 5.46.1.6 start_hdl

```
UINT16 start_hdl
```

start handle

#### 5.46.1.7 uuid

```
UINT16 uuid[8]
```

UUID

## 5.47 LE_GATT_MSG_INDICATE_IND_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 len
- UINT8 * val

### 5.47.1 Field Documentation

#### 5.47.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.47.1.2 devid

```
UINT16 devid
```

device ID

#### 5.47.1.3 handle

```
UINT16 handle
```

attribute handle

#### 5.47.1.4 len

```
UINT16 len
```

value length

#### 5.47.1.5 val

```
UINT8* val
```

value

## 5.48 LE_GATT_MSG_NOTIFY_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 status

### 5.48.1 Field Documentation

#### 5.48.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.48.1.2 devid

```
UINT16 devid
```

device ID

#### 5.48.1.3 handle

```
UINT16 handle
```

attribute handle

#### 5.48.1.4 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.49 LE_GATT_MSG_NOTIFY_IND_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 len
- UINT8 * val

### 5.49.1 Field Documentation

#### 5.49.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.49.1.2 devid

```
UINT16 devid
```

device ID

#### 5.49.1.3 handle

```
UINT16 handle
```

attribute handle

#### 5.49.1.4 len

```
UINT16 len
```

value length

#### 5.49.1.5 val

```
UINT8* val
```

value

## 5.50 LE_GATT_MSG_OPERATION_TIMEOUT_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT8 att_op
- UINT16 conn_hdl
- UINT16 devid

### 5.50.1 Field Documentation

#### 5.50.1.1 att_op

```
UINT8 att_op
```

refer to LE_ATT_OP_* in ble_att_if.h

#### 5.50.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.50.1.3 devid

```
UINT16 devid
```

device ID

## 5.51 LE_GATT_MSG_PREPARE_WRITE_RELIABLE_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT8 att_err
- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 status

### 5.51.1 Field Documentation

#### 5.51.1.1 att_err

```
UINT8 att_err
```

0 is ok, others refer to LE_ATT_ERR_∗ in ble_att_if.h

#### 5.51.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.51.1.3 devid

```
UINT16 devid
```

device ID

#### 5.51.1.4 handle

```
UINT16 handle
```

attribute handle

#### 5.51.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.52 LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT8 att_err
- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 status

### 5.52.1 Field Documentation

#### 5.52.1.1 att_err

```
UINT8 att_err
```

0 is ok, others refer to LE_ATT_ERR_∗ in ble_att_if.h

#### 5.52.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.52.1.3 devid

```
UINT16 devid
```

device ID

#### 5.52.1.4 handle

```
UINT16 handle
```

characteristic value handle

#### 5.52.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.53 LE_GATT_MSG_READ_CHARACTERISTIC_VALUE_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT8 att_err
- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 status

### 5.53.1 Field Documentation

#### 5.53.1.1 att_err

```
UINT8 att_err
```

0 is ok, others refer to LE_ATT_ERR_∗ in ble_att_if.h

#### 5.53.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.53.1.3 devid

```
UINT16 devid
```

device ID

#### 5.53.1.4 handle

```
UINT16 handle
```

characteristic value handle

#### 5.53.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.54 LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT8 att_err
- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 status

### 5.54.1 Field Documentation

#### 5.54.1.1 att_err

```
UINT8 att_err
```

0 is ok, others refer to LE_ATT_ERR_∗ in ble_att_if.h

#### 5.54.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.54.1.3 devid

```
UINT16 devid
```

device ID

#### 5.54.1.4 handle

```
UINT16 handle
```

characteristic value handle

#### 5.54.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.55 LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT8 att_err
- UINT16 conn_hdl
- UINT16 devid
- UINT16 err_hdl
- UINT16 len
- UINT16 status
- UINT8 * val

## 5.55.1 Field Documentation

#### 5.55.1.1 att_err

```
UINT8 att_err
```

0 is ok, others refer to LE_ATT_ERR_* in ble_att_if.h

#### 5.55.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.55.1.3 devid

```
UINT16 devid
```

device ID

#### 5.55.1.4 err_hdl

```
UINT16 err_hdl
```

TBD

#### 5.55.1.5 len

```
UINT16 len
```

value length

#### 5.55.1.6 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

#### 5.55.1.7 val

```
UINT8* val
```

value

## 5.56 LE_GATT_MSG_SERVICE_INFO_IND_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 devid
- UINT16 end_hdl
- UINT8 format
- UINT16 start_hdl
- UINT16 uuid [8]

### 5.56.1 Field Documentation

#### 5.56.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.56.1.2 devid

```
UINT16 devid
```

device ID

#### 5.56.1.3 end_hdl

```
UINT16 end_hdl
```

end handle

#### 5.56.1.4 format

```
UINT8 format
```

UUID type

#### 5.56.1.5 start_hdl

```
UINT16 start_hdl
```

start handle

#### 5.56.1.6 uuid

```
UINT16 uuid[8]
```

UUID

## 5.57 LE_GATT_MSG_SIGNED_WRITE_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 status

### 5.57.1 Field Documentation

#### 5.57.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.57.1.2 devid

```
UINT16 devid
```

device ID

#### 5.57.1.3 handle

```
UINT16 handle
```

attribute handle

#### 5.57.1.4 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.58 LE_GATT_MSG_WRITE_CHAR_VAL_RELIABLE_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT8 att_err
- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 status

### 5.58.1 Field Documentation

#### 5.58.1.1 att_err

```
UINT8 att_err
```

0 is ok, others refer to LE_ATT_ERR_∗ in ble_att_if.h

#### 5.58.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.58.1.3 devid

```
UINT16 devid
```

device ID

#### 5.58.1.4 handle

```
UINT16 handle
```

characteristic value handle

#### 5.58.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.59 LE_GATT_MSG_WRITE_CHAR_VALUE_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT8 att_err
- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 status

### 5.59.1 Field Documentation

#### 5.59.1.1 att_err

```
UINT8 att_err
```

0 is ok, others refer to LE_ATT_ERR_* in ble_att_if.h

#### 5.59.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.59.1.3 devid

```
UINT16 devid
```

device ID

#### 5.59.1.4 handle

```
UINT16 handle
```

attribute handle

#### 5.59.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.60 LE_GATT_MSG_WRITE_LONG_CHAR_VALUE_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT8 att_err
- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 status

### 5.60.1 Field Documentation

#### 5.60.1.1 att_err

```
UINT8 att_err
```

0 is ok, others refer to LE_ATT_ERR_∗ in ble_att_if.h

#### 5.60.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.60.1.3 devid

```
UINT16 devid
```

device ID

#### 5.60.1.4 handle

```
UINT16 handle
```

characteristic value handle

#### 5.60.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.61 LE_GATT_MSG_WRITE_NO_RSP_CFM_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle
- UINT16 status

### 5.61.1 Field Documentation

#### 5.61.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.61.1.2 devid

```
UINT16 devid
```

device ID

#### 5.61.1.3 handle

```
UINT16 handle
```

attribute handle

#### 5.61.1.4 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.62 LE_GATT_SERVICE_T Struct Reference

```
#include <ble_gatt_if.h>
```

### Data Fields

- UINT16 endHdl
- LE_GATT_ATTR_T * pAttr
- UINT16 startHdl
- UINT16 svc_id

### 5.62.1 Field Documentation

#### 5.62.1.1 endHdl

```
UINT16 endHdl
```

end handle

#### 5.62.1.2 pAttr

```
LE_GATT_ATTR_T* pAttr
```

pointer attribute table

#### 5.62.1.3 startHdl

```
UINT16 startHdl
```

start handle

#### 5.62.1.4 svc_id

```
UINT16 svc_id
```

service ID

## 5.63 LE_SMP_MSG_ENCRYPTION_CHANGE_IND_T Struct Reference

`#include <ble_smp_if.h>`

### Data Fields

- UINT16 conn_hdl
- BOOL enable

### 5.63.1 Field Documentation

#### 5.63.1.1 conn_hdl

`UINT16 conn_hdl`

connection handle

#### 5.63.1.2 enable

`BOOL enable`

enable or disable

## 5.64 LE_SMP_MSG_ENCRYPTION_REFRESH_IND_T Struct Reference

`#include <ble_smp_if.h>`

### Data Fields

- UINT16 conn_hdl
- UINT16 status

### 5.64.1 Field Documentation

#### 5.64.1.1 conn_hdl

`UINT16 conn_hdl`

connection handle

#### 5.64.1.2 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.65 LE_SMP_MSG_OOB_DATA_REQUEST_IND_T Struct Reference

```
#include <ble_smp_if.h>
```

### Data Fields

- UINT16 conn_hdl

### 5.65.1 Field Documentation

#### 5.65.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

## 5.66 LE_SMP_MSG_PAIRING_ACTION_IND_T Struct Reference

```
#include <ble_smp_if.h>
```

### Data Fields

- UINT8 action
- UINT16 conn_hdl
- BOOL lost_bond
- UINT8 sc

### 5.66.1 Field Documentation

#### 5.66.1.1 action

```
UINT8 action
```

refer to LE_SM_IO_CAP_∗ in ble_smp_if.h

#### 5.66.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.66.1.3 lost_bond

```
BOOL lost_bond
```

remote lost bond

#### 5.66.1.4 sc

```
UINT8 sc
```

secure connection

## 5.67 LE_SMP_MSG_PAIRING_COMPLETE_IND_T Struct Reference

```
#include <ble_smp_if.h>
```

### Data Fields

- UINT8 authenticated
- UINT8 bonded
- UINT16 conn_hdl
- LE_BT_ADDR_T peer_id_addr
- UINT8 sc
- UINT16 status

### 5.67.1 Field Documentation

#### 5.67.1.1 authenticated

```
UINT8 authenticated
```

authenticated

#### 5.67.1.2 bonded

```
UINT8 bonded
```

bonded

#### 5.67.1.3 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.67.1.4 peer_id_addr

```
LE_BT_ADDR_T peer_id_addr
```

peer device address

#### 5.67.1.5 sc

```
UINT8 sc
```

secure connection

#### 5.67.1.6 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

## 5.68 LE_SMP_MSG_PASSKEY_DISPLAY_IND_T Struct Reference

```
#include <ble_smp_if.h>
```

### Data Fields

- UINT16 conn_hdl
- UINT32 passkey

### 5.68.1 Field Documentation

#### 5.68.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

#### 5.68.1.2 passkey

```
UINT32 passkey
```

passkey

## 5.69 LE_SMP_MSG_PASSKEY_INPUT_IND_T Struct Reference

```
#include <ble_smp_if.h>
```

### Data Fields

- UINT16 conn_hdl

### 5.69.1 Field Documentation

#### 5.69.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

## 5.70 LE_SMP_MSG_SC_OOB_DATA_REQUEST_IND_T Struct Reference

```
#include <ble_smp_if.h>
```

### Data Fields

- UINT16 conn_hdl

### 5.70.1 Field Documentation

#### 5.70.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

## 5.71 LE_SMP_MSG_SLAVE_SECURITY_REQUEST_IND_T Struct Reference

`#include <ble_smp_if.h>`

### Data Fields

- UINT8 bondable
- UINT16 conn_hdl
- UINT8 keypress
- UINT8 mitm
- UINT8 sc

### 5.71.1 Field Documentation

#### 5.71.1.1 bondable

`UINT8 bondable`

bonding

#### 5.71.1.2 conn_hdl

`UINT16 conn_hdl`

connection handle

#### 5.71.1.3 keypress

`UINT8 keypress`

keypress status

#### 5.71.1.4 mitm

`UINT8 mitm`

MITM

#### 5.71.1.5 sc

`UINT8 sc`

secure connection

## 5.72 LE_SMP_MSG_USER_CONFIRM_IND_T Struct Reference

`#include <ble_smp_if.h>`

### Data Fields

- UINT32 confirm_num
- UINT16 conn_hdl

### 5.72.1 Field Documentation

#### 5.72.1.1 confirm_num

`UINT32 confirm_num`

confirm number

#### 5.72.1.2 conn_hdl

`UINT16 conn_hdl`

connection handle

## 5.73 LE_SMP_SC_OOB_DATA_T Struct Reference

`#include <ble_smp_if.h>`

### Data Fields

- UINT8 confirm [16]
- UINT8 rand [16]

### 5.73.1 Field Documentation

#### 5.73.1.1 confirm

`UINT8 confirm[16]`

confirm data

#### 5.73.1.2 rand

```
UINT8 rand[16]
```

random data

## 5.74 LE_SYS_MSG_BUF_OVERFLOW_T Struct Reference

```
#include <ble_msg.h>
```

### Data Fields

- UINT16 conn_hdl

### 5.74.1 Field Documentation

#### 5.74.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

## 5.75 mw_wifi_auto_connect_ap_info_t Struct Reference

```
#include <controller_wifi_com_patch.h>
```

### Data Fields

- u8 ap_channel
- u16 beacon_interval
- u8 bssid [MAC_ADDR_LEN]
- u16 capabilities
- u8 dtim_prod
- u8 fast_connect
- bool free_ocpy
- s8 hid_ssid [IEEE80211_MAX_SSID_LEN+1]
- u64 latest_beacon_rx_time
- s8 passphrase [64]
- u8 psk [32]
- u8 rsn_ie [100]
- s8 rssi
- s8 ssid [IEEE80211_MAX_SSID_LEN+1]
- u8 supported_rates [SUPPORTED_RATES_MAX]
- wpa_ie_data_t wpa_data
- u8 wpa_ie [100]

### 5.75.1 Field Documentation

#### 5.75.1.1 ap_channel

```
u8 ap_channel
```

#### 5.75.1.2 beacon_interval

```
u16 beacon_interval
```

#### 5.75.1.3 bssid

```
u8 bssid[MAC_ADDR_LEN]
```

#### 5.75.1.4 capabilities

```
u16 capabilities
```

#### 5.75.1.5 dtim_prod

```
u8 dtim_prod
```

#### 5.75.1.6 fast_connect

```
u8 fast_connect
```

#### 5.75.1.7 free_ocpy

```
bool free_ocpy
```

#### 5.75.1.8 hid_ssid

```
s8 hid_ssid[IEEE80211_MAX_SSID_LEN+1]
```

#### 5.75.1.9 latest_beacon_rx_time

```
u64 latest_beacon_rx_time
```

#### 5.75.1.10 passphrase

```
s8 passphrase[64]
```

#### 5.75.1.11 psk

```
u8 psk[32]
```

#### 5.75.1.12 rsn_ie

```
u8 rsn_ie[100]
```

#### 5.75.1.13 rssi

```
s8 rssi
```

#### 5.75.1.14 ssid

```
s8 ssid[IEEE80211_MAX_SSID_LEN+1]
```

#### 5.75.1.15 supported_rates

```
u8 supported_rates[SUPPORTED_RATES_MAX]
```

#### 5.75.1.16 wpa_data

```
wpa_ie_data_t wpa_data
```

#### 5.75.1.17 wpa_ie

```
u8 wpa_ie[100]
```

## 5.76 MwFimAutoConnectCFG_t Struct Reference

```
#include <controller_wifi_com_patch.h>
```

### Data Fields

- bool flag
- s8 front
- u8 max_save_num
- s8 rear
- u8 targetIdx

### 5.76.1 Field Documentation

#### 5.76.1.1 flag

```
bool flag
```

#### 5.76.1.2 front

```
s8 front
```

#### 5.76.1.3 max_save_num

```
u8 max_save_num
```

#### 5.76.1.4 rear

```
s8 rear
```

#### 5.76.1.5 targetIdx

```
u8 targetIdx
```

## 5.77 T_RfCmd Struct Reference

```
#include <controller_wifi_patch.h>
```

### Data Fields

- int iArgc
- char * saArgv [RF_CMD_PARAM_NUM]
- uint32_t u32Type

### 5.77.1 Field Documentation

#### 5.77.1.1 iArgc

```
int iArgc
```

#### 5.77.1.2 saArgv

```
char* saArgv[RF_CMD_PARAM_NUM]
```

#### 5.77.1.3 u32Type

```
uint32_t u32Type
```

## 5.78 T_RfEvt Struct Reference

```
#include <controller_wifi_patch.h>
```

### Data Fields

- void * pParam
- uint16_t u16RfMode
- uint16_t u16RxCnt
- uint16_t u16RxCrcOkCnt
- uint32_t u32Freq
- uint32_t u32Mode
- uint32_t u32RfChannel
- uint32_t u32Type
- uint8_t u8Freq
- uint8_t u8IpcEnable
- uint8_t u8Len
- uint8_t u8Pkt
- uint8_t u8Reserved
- uint8_t u8Status
- uint8_t u8Unicast

## 5.78.1 Field Documentation

#### 5.78.1.1 pParam

```
void* pParam
```

#### 5.78.1.2 u16RfMode

```
uint16_t u16RfMode
```

#### 5.78.1.3 u16RxCnt

```
uint16_t u16RxCnt
```

#### 5.78.1.4 u16RxCrcOkCnt

```
uint16_t u16RxCrcOkCnt
```

#### 5.78.1.5 u32Freq

```
uint32_t u32Freq
```

#### 5.78.1.6 u32Mode

```
uint32_t u32Mode
```

#### 5.78.1.7 u32RfChannel

```
uint32_t u32RfChannel
```

#### 5.78.1.8 u32Type

```
uint32_t u32Type
```

#### 5.78.1.9 u8Freq

```
uint8_t u8Freq
```

#### 5.78.1.10 u8IpcEnable

```
uint8_t u8IpcEnable
```

#### 5.78.1.11 u8Len

```
uint8_t u8Len
```

#### 5.78.1.12 u8Pkt

```
uint8_t u8Pkt
```

#### 5.78.1.13 u8Reserved

```
uint8_t u8Reserved
```

#### 5.78.1.14 u8Status

```
uint8_t u8Status
```

#### 5.78.1.15 u8Unicast

```
uint8_t u8Unicast
```

## 5.79 wifi_active_scan_time_t Struct Reference

Range of active scan times per channel.

```
#include <wifi_types.h>
```

### Data Fields

- uint32_t max
- uint32_t min

### 5.79.1 Detailed Description

Range of active scan times per channel.

### 5.79.2 Field Documentation

#### 5.79.2.1 max

```
uint32_t max
```

maximum active scan time per channel, units: millisecond, values above 1500ms may cause station to disconnect from AP and are not recommended.

#### 5.79.2.2 min

```
uint32_t min
```

minimum active scan time per channel, units: millisecond

## 5.80 wifi_ap_config_t Struct Reference

This structure is the Wi-Fi configuration for initialization for Soft-AP mode.

```
#include <wifi_types.h>
```

### Data Fields

- wifi_auth_mode_t auth_mode
- uint16_t beacon_interval
- uint8_t channel
- wifi_cipher_type_t encrypt_type
- uint8_t max_connection
- uint8_t password [WIFI_LENGTH_PASSPHRASE]
- uint8_t password_length
- uint8_t ssid [WIFI_MAX_LENGTH_OF_SSID]
- uint8_t ssid_hidden
- uint8_t ssid_length

### 5.80.1 Detailed Description

This structure is the Wi-Fi configuration for initialization for Soft-AP mode.

### 5.80.2 Field Documentation

#### 5.80.2.1 auth_mode

```
wifi_auth_mode_t auth_mode
```

The authentication mode.

#### 5.80.2.2 beacon_interval

```
uint16_t beacon_interval
```

Beacon interval, 100 ~ 60000 ms, default 100 ms

#### 5.80.2.3 channel

```
uint8_t channel
```

The channel of Soft-AP.

#### 5.80.2.4 encrypt_type

```
wifi_cipher_type_t encrypt_type
```

The encryption mode.

#### 5.80.2.5 max_connection

```
uint8_t max_connection
```

Max number of stations allowed to connect in, default 4, max 4

#### 5.80.2.6 password

```
uint8_t password[WIFI_LENGTH_PASSPHRASE]
```

The password of the Soft-AP.

#### 5.80.2.7 password_length

```
uint8_t password_length
```

The length of the password.

#### 5.80.2.8 ssid

```
uint8_t ssid[WIFI_MAX_LENGTH_OF_SSID]
```

The SSID of the Soft-AP.

#### 5.80.2.9 ssid_hidden

```
uint8_t ssid_hidden
```

Broadcast SSID or not, default 0, broadcast the SSID

#### 5.80.2.10 ssid_length

```
uint8_t ssid_length
```

The length of the SSID.

## 5.81 wifi_auto_connect_info_f Struct Reference

WiFi auto connect info parameters.

```
#include <wifi_types.h>
```

### Data Fields

- uint8_t ap_channel
- uint16_t beacon_interval
- uint8_t bssid [WIFI_MAC_ADDRESS_LENGTH]
- uint16_t capabilities
- uint8_t dtim_prod
- uint8_t fast_connect
- bool free_ocpy
- int8_t hid_ssid [WIFI_MAX_LENGTH_OF_SSID]
- unsigned long latest_beacon_rx_time
- int8_t passphrase [WIFI_LENGTH_PASSPHRASE]
- uint8_t psk [32]
- uint8_t rsn_ie [100]
- int8_t rssi
- int8_t ssid [WIFI_MAX_LENGTH_OF_SSID]
- uint8_t supported_rates [WIFI_MAX_SUPPORTED_RATES]
- wpa_ie_data_t wpa_data
- uint8_t wpa_ie [100]

### 5.81.1 Detailed Description

WiFi auto connect info parameters.

### 5.81.2 Field Documentation

#### 5.81.2.1 ap_channel

```
uint8_t ap_channel
```

#### 5.81.2.2 beacon_interval

```
uint16_t beacon_interval
```

#### 5.81.2.3 bssid

```
uint8_t bssid[WIFI_MAC_ADDRESS_LENGTH]
```

#### 5.81.2.4 capabilities

```
uint16_t capabilities
```

#### 5.81.2.5 dtim_prod

```
uint8_t dtim_prod
```

#### 5.81.2.6 fast_connect

```
uint8_t fast_connect
```

#### 5.81.2.7 free_ocpy

```
bool free_ocpy
```

#### 5.81.2.8 hid_ssid

```
int8_t hid_ssid[WIFI_MAX_LENGTH_OF_SSID]
```

#### 5.81.2.9 latest_beacon_rx_time

```
unsigned long latest_beacon_rx_time
```

#### 5.81.2.10 passphrase

```
int8_t passphrase[WIFI_LENGTH_PASSPHRASE]
```

#### 5.81.2.11 psk

```
uint8_t psk[32]
```

#### 5.81.2.12 rsn_ie

```
uint8_t rsn_ie[100]
```

#### 5.81.2.13 rssi

```
int8_t rssi
```

#### 5.81.2.14 ssid

```
int8_t ssid[WIFI_MAX_LENGTH_OF_SSID]
```

#### 5.81.2.15 supported_rates

```
uint8_t supported_rates[WIFI_MAX_SUPPORTED_RATES]
```

#### 5.81.2.16 wpa_data

```
wpa_ie_data_t wpa_data
```

#### 5.81.2.17 wpa_ie

```
uint8_t wpa_ie[100]
```

## 5.82 wifi_config_t Union Reference

Wi-Fi configuration for initialization.

```
#include <wifi_types.h>
```

### Data Fields

- wifi_ap_config_t ap_config
- wifi_sta_config_t sta_config

#### 5.82.1 Detailed Description

Wi-Fi configuration for initialization.

#### 5.82.2 Field Documentation

##### 5.82.2.1 ap_config

```
wifi_ap_config_t ap_config
```

The configurations for certain AP. It should be set when the OPMODE is #WIFI_MODE_AP_ONLY .

##### 5.82.2.2 sta_config

```
wifi_sta_config_t sta_config
```

The configurations for the STA. It should be set when the OPMODE is #WIFI_MODE_STA_ONLY.

## 5.83 wifi_event_info_t Union Reference

wifi_event_info_t

```
#include <wifi_event.h>
```

### Data Fields

- wifi_event_sta_connected_t connected
- wifi_event_sta_disconnected_t disconnected
- wifi_event_sta_got_ip_t got_ip
- wifi_event_sta_scan_done_t scan_done

#### 5.83.1 Detailed Description

wifi_event_info_t

### 5.83.2 Field Documentation

#### 5.83.2.1 connected

```
wifi_event_sta_connected_t connected
```

station connected to AP

#### 5.83.2.2 disconnected

```
wifi_event_sta_disconnected_t disconnected
```

station disconnected to AP

#### 5.83.2.3 got_ip

```
wifi_event_sta_got_ip_t got_ip
```

station got IP, first time got IP or when IP is changed

#### 5.83.2.4 scan_done

```
wifi_event_sta_scan_done_t scan_done
```

station scan (APs) done

## 5.84 wifi_event_sta_connected_t Struct Reference

wifi_event_sta_connected_t

```
#include <wifi_event.h>
```

### Data Fields

- wifi_auth_mode_t authmode
- uint8_t bssid [6]
- uint8_t channel
- uint8_t ssid [32]
- uint8_t ssid_len

### 5.84.1 Detailed Description

wifi_event_sta_connected_t

### 5.84.2 Field Documentation

#### 5.84.2.1 authmode

```
wifi_auth_mode_t authmode
```

#### 5.84.2.2 bssid

```
uint8_t bssid[6]
```

BSSID of connected AP

#### 5.84.2.3 channel

```
uint8_t channel
```

channel of connected AP

#### 5.84.2.4 ssid

```
uint8_t ssid[32]
```

SSID of connected AP

#### 5.84.2.5 ssid_len

```
uint8_t ssid_len
```

SSID length of connected AP

## 5.85 wifi_event_sta_disconnected_t Struct Reference

wifi_event_sta_disconnected_t

```
#include <wifi_event.h>
```

### Data Fields

- uint8_t bssid [6]
- uint8_t reason
- uint8_t ssid [32]
- uint8_t ssid_len

### 5.85.1 Detailed Description

wifi_event_sta_disconnected_t

### 5.85.2 Field Documentation

#### 5.85.2.1 bssid

```
uint8_t bssid[6]
```

BSSID of disconnected AP

#### 5.85.2.2 reason

```
uint8_t reason
```

reason of disconnection

#### 5.85.2.3 ssid

```
uint8_t ssid[32]
```

SSID of disconnected AP

#### 5.85.2.4 ssid_len

```
uint8_t ssid_len
```

SSID length of disconnected AP

## 5.86 wifi_event_sta_got_ip_t Struct Reference

wifi_event_sta_got_ip_t

```
#include <wifi_event.h>
```

### Data Fields

- bool ip_changed

### 5.86.1 Detailed Description

wifi_event_sta_got_ip_t

### 5.86.2 Field Documentation

#### 5.86.2.1 ip_changed

```
bool ip_changed
```

## 5.87 wifi_event_sta_scan_done_t Struct Reference

wifi_event_sta_scan_done_t

```
#include <wifi_event.h>
```

### Data Fields

- uint8_t number
- uint8_t scan_id
- uint32_t status

### 5.87.1 Detailed Description

wifi_event_sta_scan_done_t

### 5.87.2 Field Documentation

#### 5.87.2.1 number

```
uint8_t number
```

The number of devices scanned

#### 5.87.2.2 scan_id

```
uint8_t scan_id
```

scan id

#### 5.87.2.3 status

```
uint32_t status
```

status of scanning APs

## 5.88 wifi_fast_scan_threshold_t Struct Reference

Structure describing parameters for a Wi-Fi fast scan.

```
#include <wifi_types.h>
```

### Data Fields

- wifi_auth_mode_t authmode
- int8_t rssi

### 5.88.1 Detailed Description

Structure describing parameters for a Wi-Fi fast scan.

### 5.88.2 Field Documentation

#### 5.88.2.1 authmode

```
wifi_auth_mode_t authmode
```

The weakest authmode to accept in the fast scan mode

#### 5.88.2.2 rssi

```
int8_t rssi
```

The minimum rssi to accept in the fast scan mode

## 5.89 wifi_init_config_t Struct Reference

WiFi stack configuration parameters.

```
#include <wifi_types.h>
```

### Data Fields

- wifi_event_notify_cb_t event_handler
- int magic

#### 5.89.1 Detailed Description

WiFi stack configuration parameters.

#### 5.89.2 Field Documentation

##### 5.89.2.1 event_handler

```
wifi_event_notify_cb_t event_handler
```

WiFi event handler

##### 5.89.2.2 magic

```
int magic
```

WiFi init magic number, it should be the last field

## 5.90 wifi_scan_config_t Struct Reference

Parameters for an SSID scan.

```
#include <wifi_types.h>
```

### Data Fields

- uint8_t * bssid
- uint8_t channel
- wifi_scan_time_t scan_time
- wifi_scan_type_t scan_type
- bool show_hidden
- uint8_t * ssid

#### 5.90.1 Detailed Description

Parameters for an SSID scan.

### 5.90.2 Field Documentation

#### 5.90.2.1 bssid

```
uint8_t* bssid
```

MAC address of AP

#### 5.90.2.2 channel

```
uint8_t channel
```

channel, scan the specific channel

#### 5.90.2.3 scan_time

```
wifi_scan_time_t scan_time
```

scan time per channel

#### 5.90.2.4 scan_type

```
wifi_scan_type_t scan_type
```

scan type, active or passive

#### 5.90.2.5 show_hidden

```
bool show_hidden
```

enable to scan AP whose SSID is hidden

#### 5.90.2.6 ssid

```
uint8_t* ssid
```

SSID of AP

## 5.91 wifi_scan_info_t Struct Reference

This structure defines the inforamtion of scanned APs.

```
#include <wifi_types.h>
```

### Data Fields

- wifi_auth_mode_t auth_mode
- uint16_t beacon_interval
- uint8_t bssid [WIFI_MAC_ADDRESS_LENGTH]
- uint16_t capability_info
- uint8_t channel
- wifi_cipher_type_t group_cipher
- wifi_cipher_type_t pairwise_cipher
- int rssi
- uint8_t ssid [WIFI_MAX_LENGTH_OF_SSID]
- uint8_t ssid_length

## 5.91.1 Detailed Description

This structure defines the inforamtion of scanned APs.

## 5.91.2 Field Documentation

#### 5.91.2.1 auth_mode

```
wifi_auth_mode_t auth_mode
```

Please refer to the definition of wifi_auth_mode_t.

#### 5.91.2.2 beacon_interval

```
uint16_t beacon_interval
```

Indicates the beacon interval.

#### 5.91.2.3 bssid

```
uint8_t bssid[WIFI_MAC_ADDRESS_LENGTH]
```

AP's MAC address.

#### 5.91.2.4 capability_info

```
uint16_t capability_info
```

The Capability Information field contains a number of subfields that are used to indicate requested or advertised optional capabilities.

#### 5.91.2.5 channel

```
uint8_t channel
```

The channel used.

#### 5.91.2.6 group_cipher

```
wifi_cipher_type_t group_cipher
```

group cipher of AP

#### 5.91.2.7 pairwise_cipher

```
wifi_cipher_type_t pairwise_cipher
```

pairwise cipher of AP, Please refer to the definition of #wifi_encrypt_type_t.

#### 5.91.2.8 rssi

```
int rssi
```

Records the RSSI value when probe response is received.

#### 5.91.2.9 ssid

```
uint8_t ssid[WIFI_MAX_LENGTH_OF_SSID]
```

Stores the predefined SSID.

#### 5.91.2.10 ssid_length

```
uint8_t ssid_length
```

Length of the SSID.

## 5.92 wifi_scan_list_t Struct Reference

This structure defines the list of scanned APs with their corresponding information.

```
#include <wifi_types.h>
```

### Data Fields

- wifi_scan_info_t ap_record [WIFI_MAX_SCAN_AP_NUM]
- int num

### 5.92.1 Detailed Description

This structure defines the list of scanned APs with their corresponding information.

### 5.92.2 Field Documentation

#### 5.92.2.1 ap_record

```
wifi_scan_info_t ap_record[WIFI_MAX_SCAN_AP_NUM]
```

The information about an AP obtained through the scan result is stored

#### 5.92.2.2 num

```
int num
```

number of AP in the list

## 5.93 wifi_scan_time_t Union Reference

Aggregate of active & passive scan time per channel.

```
#include <wifi_types.h>
```

### Data Fields

- wifi_active_scan_time_t active
- uint32_t passive

### 5.93.1 Detailed Description

Aggregate of active & passive scan time per channel.

### 5.93.2 Field Documentation

#### 5.93.2.1 active

```
wifi_active_scan_time_t active
```

active scan time per channel, units: millisecond.

#### 5.93.2.2 passive

```
uint32_t passive
```

passive scan time per channel, units: millisecond, values above 1500ms may cause station to disconnect from AP and are not recommended.

## 5.94 wifi_sta_config_t Struct Reference

This structure is the Wi-Fi configuration for initialization for STA mode.

```
#include <wifi_types.h>
```

### Data Fields

- uint8_t bssid [WIFI_MAC_ADDRESS_LENGTH]
- uint8_t bssid_present
- uint8_t password [WIFI_LENGTH_PASSPHRASE]
- uint8_t password_length
- wifi_scan_method_t scan_method
- wifi_sort_method_t sort_method
- uint8_t ssid [WIFI_MAX_LENGTH_OF_SSID]
- uint8_t ssid_length
- wifi_fast_scan_threshold_t threshold

### 5.94.1 Detailed Description

This structure is the Wi-Fi configuration for initialization for STA mode.

### 5.94.2 Field Documentation

#### 5.94.2.1 bssid

```
uint8_t bssid[WIFI_MAC_ADDRESS_LENGTH]
```

The MAC address of the target AP.

#### 5.94.2.2 bssid_present

```
uint8_t bssid_present
```

The BSSID is present if it is set to 1. Otherwise, it is set to 0.

#### 5.94.2.3 password

```
uint8_t password[WIFI_LENGTH_PASSPHRASE]
```

The password of the target AP.

#### 5.94.2.4 password_length

```
uint8_t password_length
```

The length of the password. If the length is 64, the password is regarded as PMK.

#### 5.94.2.5 scan_method

```
wifi_scan_method_t scan_method
```

do all channel scan or fast scan

#### 5.94.2.6 sort_method

```
wifi_sort_method_t sort_method
```

sort the connect AP in the list by rssi or security mode

#### 5.94.2.7 ssid

```
uint8_t ssid[WIFI_MAX_LENGTH_OF_SSID]
```

The SSID of the target AP.

#### 5.94.2.8 ssid_length

```
uint8_t ssid_length
```

The length of the SSID.

#### 5.94.2.9 threshold

```
wifi_fast_scan_threshold_t threshold
```

When scan_method is set to WIFI_FAST_SCAN, only APs which have an auth mode that is more secure than the selected auth mode and a signal stronger than the minimum RSSI will be used.

# Index