

# OPL1000\_WIFI\_BLE\_API\_GUIDE

**1.0.1.18**

Generated by Doxygen 1.8.14



# Contents

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>SDK PREVIEW</b>                                   | <b>1</b> |
| <b>2</b> | <b>Module Index</b>                                  | <b>3</b> |
| 2.1      | Modules . . . . .                                    | 3        |
| <b>3</b> | <b>Data Structure Index</b>                          | <b>5</b> |
| 3.1      | Data Structures . . . . .                            | 5        |
| <b>4</b> | <b>Module Documentation</b>                          | <b>9</b> |
| 4.1      | BLE ALL APIs . . . . .                               | 9        |
| 4.1.1    | Detailed Description . . . . .                       | 9        |
| 4.2      | BLE CM APIs . . . . .                                | 10       |
| 4.2.1    | Detailed Description . . . . .                       | 11       |
| 4.2.2    | Typedef Documentation . . . . .                      | 11       |
| 4.2.2.1  | LE_CM_MSG_ADD_TO_RESOLVING_LIST_CFM_T . . . . .      | 11       |
| 4.2.2.2  | LE_CM_MSG_ADD_TO_WHITE_LIST_CFM_T . . . . .          | 11       |
| 4.2.2.3  | LE_CM_MSG_CANCEL_CONNECTION_CFM_T . . . . .          | 12       |
| 4.2.2.4  | LE_CM_MSG_CLEAR_RESOLVING_LIST_CFM_T . . . . .       | 12       |
| 4.2.2.5  | LE_CM_MSG_CLEAR_WHITE_LIST_CFM_T . . . . .           | 12       |
| 4.2.2.6  | LE_CM_MSG_CREATE_CONNECTION_CFM_T . . . . .          | 12       |
| 4.2.2.7  | LE_CM_MSG_ENTER_ADVERTISING_CFM_T . . . . .          | 12       |
| 4.2.2.8  | LE_CM_MSG_ENTER_SCANNING_CFM_T . . . . .             | 12       |
| 4.2.2.9  | LE_CM_MSG_EXIT_ADVERTISING_CFM_T . . . . .           | 12       |
| 4.2.2.10 | LE_CM_MSG_EXIT_SCANNING_CFM_T . . . . .              | 12       |
| 4.2.2.11 | LE_CM_MSG_REMOVE_FROM_RESOLVING_LIST_CFM_T . . . . . | 13       |

|          |  |    |
|----------|--|----|
| 4.2.2.12 | LE_CM_MSG_REMOVE_FROM_WHITE_LIST_CFM_T . . . . . | 13 |
| 4.2.2.13 | LE_CM_MSG_SET_ADVERTISING_DATA_CFM_T . . . . .   | 13 |
| 4.2.2.14 | LE_CM_MSG_SET_ADVERTISING_PARAMS_CFM_T . . . . . | 13 |
| 4.2.2.15 | LE_CM_MSG_SET_CHANNEL_MAP_CFM_T . . . . .        | 13 |
| 4.2.2.16 | LE_CM_MSG_SET_RANDOM_ADDRESS_CFM_T . . . . .     | 13 |
| 4.2.2.17 | LE_CM_MSG_SET_RPA_TIMEOUT_CFM_T . . . . .        | 13 |
| 4.2.2.18 | LE_CM_MSG_SET_SCAN_PARAMS_CFM_T . . . . .        | 13 |
| 4.2.2.19 | LE_CM_MSG_SET_SCAN_RSP_DATA_CFM_T . . . . .      | 14 |
| 4.2.3    | Enumeration Type Documentation . . . . .         | 14 |
| 4.2.3.1  | anonymous enum . . . . .                         | 14 |
| 4.2.4    | Function Documentation . . . . .                 | 15 |
| 4.2.4.1  | LeCmInit() . . . . .                             | 15 |
| 4.3      | BLE GAP APIs . . . . .                           | 16 |
| 4.3.1    | Detailed Description . . . . .                   | 18 |
| 4.3.2    | Macro Definition Documentation . . . . .         | 18 |
| 4.3.2.1  | GAP_ADTYPE_128BIT_COMPLETE . . . . .             | 18 |
| 4.3.2.2  | GAP_ADTYPE_128BIT_MORE . . . . .                 | 18 |
| 4.3.2.3  | GAP_ADTYPE_16BIT_COMPLETE . . . . .              | 18 |
| 4.3.2.4  | GAP_ADTYPE_16BIT_MORE . . . . .                  | 19 |
| 4.3.2.5  | GAP_ADTYPE_32BIT_COMPLETE . . . . .              | 19 |
| 4.3.2.6  | GAP_ADTYPE_32BIT_MORE . . . . .                  | 19 |
| 4.3.2.7  | GAP_ADTYPE_3D_INFO_DATA . . . . .                | 19 |
| 4.3.2.8  | GAP_ADTYPE_ADV_INTERVAL . . . . .                | 19 |
| 4.3.2.9  | GAP_ADTYPE_APPEARANCE . . . . .                  | 19 |
| 4.3.2.10 | GAP_ADTYPE_FLAGS . . . . .                       | 19 |
| 4.3.2.11 | GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED . . . . .   | 19 |
| 4.3.2.12 | GAP_ADTYPE_FLAGS_GENERAL . . . . .               | 20 |
| 4.3.2.13 | GAP_ADTYPE_FLAGS_LIMITED . . . . .               | 20 |
| 4.3.2.14 | GAP_ADTYPE_LE_BD_ADDR . . . . .                  | 20 |
| 4.3.2.15 | GAP_ADTYPE_LE_ROLE . . . . .                     | 20 |

|          |  |    |
|----------|--|----|
| 4.3.2.16 | GAP_ADTYPE_LOCAL_NAME_COMPLETE . . . . .       | 20 |
| 4.3.2.17 | GAP_ADTYPE_LOCAL_NAME_SHORT . . . . .          | 20 |
| 4.3.2.18 | GAP_ADTYPE_MANUFACTURER_SPECIFIC . . . . .     | 20 |
| 4.3.2.19 | GAP_ADTYPE_OOB_CLASS_OF_DEVICE . . . . .       | 20 |
| 4.3.2.20 | GAP_ADTYPE_OOB_SIMPLE_PAIRING_HASHC . . . . .  | 21 |
| 4.3.2.21 | GAP_ADTYPE_OOB_SIMPLE_PAIRING_RANDR . . . . .  | 21 |
| 4.3.2.22 | GAP_ADTYPE_POWER_LEVEL . . . . .               | 21 |
| 4.3.2.23 | GAP_ADTYPE_PUBLIC_TARGET_ADDR . . . . .        | 21 |
| 4.3.2.24 | GAP_ADTYPE_RANDOM_TARGET_ADDR . . . . .        | 21 |
| 4.3.2.25 | GAP_ADTYPE_SERVICE_DATA . . . . .              | 21 |
| 4.3.2.26 | GAP_ADTYPE_SERVICE_DATA_128BIT . . . . .       | 21 |
| 4.3.2.27 | GAP_ADTYPE_SERVICE_DATA_32BIT . . . . .        | 21 |
| 4.3.2.28 | GAP_ADTYPE_SERVICES_LIST_128BIT . . . . .      | 22 |
| 4.3.2.29 | GAP_ADTYPE_SERVICES_LIST_16BIT . . . . .       | 22 |
| 4.3.2.30 | GAP_ADTYPE_SIGNED_DATA . . . . .               | 22 |
| 4.3.2.31 | GAP_ADTYPE_SIMPLE_PAIRING_HASHC_256 . . . . .  | 22 |
| 4.3.2.32 | GAP_ADTYPE_SIMPLE_PAIRING_RANDR_256 . . . . .  | 22 |
| 4.3.2.33 | GAP_ADTYPE_SLAVE_CONN_INTERVAL_RANGE . . . . . | 22 |
| 4.3.2.34 | GAP_ADTYPE_SM_OOB_FLAG . . . . .               | 22 |
| 4.3.2.35 | GAP_ADTYPE_SM_TK . . . . .                     | 22 |
| 4.3.2.36 | GAP_PUBLIC_ADDR . . . . .                      | 23 |
| 4.3.2.37 | GAP_RAND_ADDR_NRPA . . . . .                   | 23 |
| 4.3.2.38 | GAP_RAND_ADDR_RPA . . . . .                    | 23 |
| 4.3.2.39 | GAP_RAND_ADDR_STATIC . . . . .                 | 23 |
| 4.3.2.40 | GAP_SCAN_TYPE_ACTIVE . . . . .                 | 23 |
| 4.3.2.41 | GAP_SCAN_TYPE_PASSIVE . . . . .                | 23 |
| 4.3.2.42 | GAP_TX_PWR_CURR_VAL . . . . .                  | 23 |
| 4.3.2.43 | GAP_TX_PWR_MAX_VAL . . . . .                   | 23 |
| 4.3.2.44 | GAPBOND_IO_CAP_DISPLAY_ONLY . . . . .          | 24 |
| 4.3.2.45 | GAPBOND_IO_CAP_DISPLAY_YES_NO . . . . .        | 24 |

|          |   |    |
|----------|---|----|
| 4.3.2.46 | GAPBOND_IO_CAP_KEYBOARD_DISPLAY . . . . .   | 24 |
| 4.3.2.47 | GAPBOND_IO_CAP_KEYBOARD_ONLY . . . . .      | 24 |
| 4.3.2.48 | GAPBOND_IO_CAP_NO_INPUT_NO_OUTPUT . . . . . | 24 |
| 4.3.2.49 | GAPBOND_PAIRING_MODE_INITIATE . . . . .     | 24 |
| 4.3.2.50 | GAPBOND_PAIRING_MODE_NO_PAIRING . . . . .   | 24 |
| 4.3.2.51 | GAPBOND_PAIRING_MODE_WAIT_FOR_REQ . . . . . | 24 |
| 4.3.2.52 | LE_GAP_ADV_MAX_SIZE . . . . .               | 25 |
| 4.3.3    | Function Documentation . . . . .            | 25 |
| 4.3.3.1  | LeGapAddToResolvingList() . . . . .         | 25 |
| 4.3.3.2  | LeGapAddToWhiteList() . . . . .             | 25 |
| 4.3.3.3  | LeGapAdvertisingEnable() . . . . .          | 26 |
| 4.3.3.4  | LeGapCentralConnectReq() . . . . .          | 26 |
| 4.3.3.5  | LeGapCentralSetDataChannel() . . . . .      | 26 |
| 4.3.3.6  | LeGapClearResolvingList() . . . . .         | 27 |
| 4.3.3.7  | LeGapClearWhiteList() . . . . .             | 27 |
| 4.3.3.8  | LeGapConnectCancelReq() . . . . .           | 27 |
| 4.3.3.9  | LeGapConnParaRequestRsp() . . . . .         | 27 |
| 4.3.3.10 | LeGapConnUpdateRequest() . . . . .          | 28 |
| 4.3.3.11 | LeGapConnUpdateResponse() . . . . .         | 28 |
| 4.3.3.12 | LeGapDisconnectReq() . . . . .              | 29 |
| 4.3.3.13 | LeGapGenRandAddr() . . . . .                | 29 |
| 4.3.3.14 | LeGapGetBtAddr() . . . . .                  | 29 |
| 4.3.3.15 | LeGapReadAdvChannelTxPower() . . . . .      | 30 |
| 4.3.3.16 | LeGapReadChannelMap() . . . . .             | 30 |
| 4.3.3.17 | LeGapReadResolvingListSize() . . . . .      | 30 |
| 4.3.3.18 | LeGapReadRssi() . . . . .                   | 30 |
| 4.3.3.19 | LeGapReadTxPower() . . . . .                | 31 |
| 4.3.3.20 | LeGapReadWhiteListSize() . . . . .          | 31 |
| 4.3.3.21 | LeGapRemoveFromWhiteList() . . . . .        | 31 |
| 4.3.3.22 | LeGapScanningReq() . . . . .                | 32 |

|          |                                |    |
|----------|--------------------------------|----|
| 4.3.3.23 | LeGapSetAdvData()              | 32 |
| 4.3.3.24 | LeGapSetAdvParameter()         | 33 |
| 4.3.3.25 | LeGapSetConnParameter()        | 33 |
| 4.3.3.26 | LeGapSetDataChannelPduLen()    | 33 |
| 4.3.3.27 | LeGapSetRandAddr()             | 34 |
| 4.3.3.28 | LeGapSetRpaTimeout()           | 34 |
| 4.3.3.29 | LeGapSetStaticAddr()           | 35 |
| 4.3.3.30 | LeSetScanParameter()           | 35 |
| 4.3.3.31 | LeSetScanRspData()             | 35 |
| 4.4      | BLE GATT APIs                  | 37 |
| 4.4.1    | Detailed Description           | 41 |
| 4.4.2    | Macro Definition Documentation | 41 |
| 4.4.2.1  | CHAR_AGGREGATE_DESCRIPTOR      | 41 |
| 4.4.2.2  | CHAR_CLIENT_CONFIG_DESCRIPTOR  | 42 |
| 4.4.2.3  | CHAR_DECL_UUID16_ATTR_VAL      | 42 |
| 4.4.2.4  | CHAR_EXT_PROP_DESCRIPTOR       | 42 |
| 4.4.2.5  | CHAR_PRESENT_FORMAT_DESCRIPTOR | 42 |
| 4.4.2.6  | CHAR_SERVER_CONFIG_DESCRIPTOR  | 42 |
| 4.4.2.7  | CHAR_USER_DESC_DESCRIPTOR      | 42 |
| 4.4.2.8  | CHARACTERISTIC_DECL_UUID128    | 43 |
| 4.4.2.9  | CHARACTERISTIC_DECL_UUID16     | 43 |
| 4.4.2.10 | CHARACTERISTIC_UUID128         | 43 |
| 4.4.2.11 | CHARACTERISTIC_UUID16          | 43 |
| 4.4.2.12 | GATT_CHAR_AGG_FORMAT_UUID      | 43 |
| 4.4.2.13 | GATT_CHAR_EXT_PROPS_UUID       | 43 |
| 4.4.2.14 | GATT_CHAR_FORMAT_UUID          | 44 |
| 4.4.2.15 | GATT_CHAR_USER_DESC_UUID       | 44 |
| 4.4.2.16 | GATT_CHARACTERISTIC_UUID       | 44 |
| 4.4.2.17 | GATT_CLIENT_CHAR_CFG_UUID      | 44 |
| 4.4.2.18 | GATT_EXT_REPORT_REF_UUID       | 44 |

|          |   |    |
|----------|---|----|
| 4.4.2.19 | GATT_INCLUDE_UUID . . . . .               | 44 |
| 4.4.2.20 | GATT_PRIMARY_SERVICE_UUID . . . . .       | 44 |
| 4.4.2.21 | GATT_REPORT_REF_UUID . . . . .            | 44 |
| 4.4.2.22 | GATT_SECONDARY_SERVICE_UUID . . . . .     | 45 |
| 4.4.2.23 | GATT_SERV_CHAR_CFG_UUID . . . . .         | 45 |
| 4.4.2.24 | GATT_VALID_RANGE_UUID . . . . .           | 45 |
| 4.4.2.25 | INCLUDE_DECL_UUID128 . . . . .            | 45 |
| 4.4.2.26 | INCLUDE_DECL_UUID128_ATTR_VAL . . . . .   | 45 |
| 4.4.2.27 | INCLUDE_DECL_UUID16_ATTR_VAL . . . . .    | 45 |
| 4.4.2.28 | INCLUDE_DECL_UUINT16 . . . . .            | 45 |
| 4.4.2.29 | LE_ATT_UUID_SIZE . . . . .                | 46 |
| 4.4.2.30 | LE_GATT_CHAR_PROP_AUTH . . . . .          | 46 |
| 4.4.2.31 | LE_GATT_CHAR_PROP_BCAST . . . . .         | 46 |
| 4.4.2.32 | LE_GATT_CHAR_PROP_EXT_PROP . . . . .      | 46 |
| 4.4.2.33 | LE_GATT_CHAR_PROP_IND . . . . .           | 46 |
| 4.4.2.34 | LE_GATT_CHAR_PROP_NTF . . . . .           | 46 |
| 4.4.2.35 | LE_GATT_CHAR_PROP_RD . . . . .            | 46 |
| 4.4.2.36 | LE_GATT_CHAR_PROP_WR . . . . .            | 47 |
| 4.4.2.37 | LE_GATT_CHAR_PROP_WR_NO_RESP . . . . .    | 47 |
| 4.4.2.38 | LE_GATT_CLIENT_CFG_INDICATION . . . . .   | 47 |
| 4.4.2.39 | LE_GATT_CLIENT_CFG_NOTIFICATION . . . . . | 47 |
| 4.4.2.40 | LE_GATT_EXT_PROP_RELIABLE_WR . . . . .    | 47 |
| 4.4.2.41 | LE_GATT_EXT_PROP_WR_AUX . . . . .         | 47 |
| 4.4.2.42 | LE_GATT_FLAG_PREPARE_WRITE . . . . .      | 47 |
| 4.4.2.43 | LE_GATT_FLAG_WRITE_CMD . . . . .          | 47 |
| 4.4.2.44 | LE_GATT_FLAG_WRITE_REQ . . . . .          | 48 |
| 4.4.2.45 | LE_GATT_PERM_AUTH_READABLE . . . . .      | 48 |
| 4.4.2.46 | LE_GATT_PERM_AUTH_WRITABLE . . . . .      | 48 |
| 4.4.2.47 | LE_GATT_PERM_NONE . . . . .               | 48 |
| 4.4.2.48 | LE_GATT_PERM_READ . . . . .               | 48 |



|          |  |    |
|----------|--|----|
| 4.4.2.49 | LE_GATT_PERM_RELIABLE_WRITE . . . . .    | 48 |
| 4.4.2.50 | LE_GATT_PERM_WRITE_CMD . . . . .         | 48 |
| 4.4.2.51 | LE_GATT_PERM_WRITE_REQ . . . . .         | 48 |
| 4.4.2.52 | LE_GATT_PERMIT_AUTHEN_READ . . . . .     | 49 |
| 4.4.2.53 | LE_GATT_PERMIT_AUTHEN_WRITE . . . . .    | 49 |
| 4.4.2.54 | LE_GATT_PERMIT_AUTHOR_READ . . . . .     | 49 |
| 4.4.2.55 | LE_GATT_PERMIT_AUTHOR_WRITE . . . . .    | 49 |
| 4.4.2.56 | LE_GATT_PERMIT_ENCRYPT_READ . . . . .    | 49 |
| 4.4.2.57 | LE_GATT_PERMIT_ENCRYPT_WRITE . . . . .   | 49 |
| 4.4.2.58 | LE_GATT_PERMIT_READ . . . . .            | 49 |
| 4.4.2.59 | LE_GATT_PERMIT_READABLE . . . . .        | 49 |
| 4.4.2.60 | LE_GATT_PERMIT_SC_AUTHEN_READ . . . . .  | 50 |
| 4.4.2.61 | LE_GATT_PERMIT_SC_AUTHEN_WRITE . . . . . | 50 |
| 4.4.2.62 | LE_GATT_PERMIT_WRITABLE . . . . .        | 50 |
| 4.4.2.63 | LE_GATT_PERMIT_WRITE . . . . .           | 50 |
| 4.4.2.64 | PRIMARY_SERVICE_DECL_UUID128 . . . . .   | 50 |
| 4.4.2.65 | PRIMARY_SERVICE_DECL_UUID16 . . . . .    | 50 |
| 4.4.2.66 | SECONDARY_SERVICE_DECL_UUID128 . . . . . | 50 |
| 4.4.2.67 | SECONDARY_SERVICE_DECL_UUID16 . . . . .  | 51 |
| 4.4.3    | Enumeration Type Documentation . . . . . | 51 |
| 4.4.3.1  | anonymous enum . . . . .                 | 51 |
| 4.4.4    | Function Documentation . . . . .         | 52 |
| 4.4.4.1  | LeGattAccessReadRsp() . . . . .          | 52 |
| 4.4.4.2  | LeGattAccessWriteRsp() . . . . .         | 52 |
| 4.4.4.3  | LeGattChangeAttrVal() . . . . .          | 53 |
| 4.4.4.4  | LeGattCharValConfirmation() . . . . .    | 53 |
| 4.4.4.5  | LeGattCharValIndicate() . . . . .        | 54 |
| 4.4.4.6  | LeGattCharValNotify() . . . . .          | 54 |
| 4.4.4.7  | LeGattExchangeMtuReq() . . . . .         | 55 |
| 4.4.4.8  | LeGattExchangeMtuRsp() . . . . .         | 55 |

|          |   |    |
|----------|---|----|
| 4.4.4.9  | <a href="#">LeGattExecuteWriteCharValReliable()</a> | 56 |
| 4.4.4.10 | <a href="#">LeGattFindAllCharacteristic()</a>       | 56 |
| 4.4.4.11 | <a href="#">LeGattFindAllCharDescriptor()</a>       | 56 |
| 4.4.4.12 | <a href="#">LeGattFindAllPrimaryService()</a>       | 57 |
| 4.4.4.13 | <a href="#">LeGattFindCharacteristicByUuid()</a>    | 57 |
| 4.4.4.14 | <a href="#">LeGattFindIncludedService()</a>         | 58 |
| 4.4.4.15 | <a href="#">LeGattFindPrimaryServiceByUuid()</a>    | 58 |
| 4.4.4.16 | <a href="#">LeGattGetAttrHandle()</a>               | 59 |
| 4.4.4.17 | <a href="#">LeGattGetAttrVal()</a>                  | 59 |
| 4.4.4.18 | <a href="#">LeGattGetAttrValLen()</a>               | 59 |
| 4.4.4.19 | <a href="#">LeGattGetAttrValMaxLen()</a>            | 61 |
| 4.4.4.20 | <a href="#">LeGattInit()</a>                        | 61 |
| 4.4.4.21 | <a href="#">LeGattModifyAttrVal()</a>               | 62 |
| 4.4.4.22 | <a href="#">LeGattPrepareWriteCharValReliable()</a> | 62 |
| 4.4.4.23 | <a href="#">LeGattReadCharValByUuid()</a>           | 63 |
| 4.4.4.24 | <a href="#">LeGattReadCharValue()</a>               | 63 |
| 4.4.4.25 | <a href="#">LeGattReadLongCharVal()</a>             | 64 |
| 4.4.4.26 | <a href="#">LeGattReadMultipleCharVal()</a>         | 64 |
| 4.4.4.27 | <a href="#">LeGattRegisterIncludeService()</a>      | 64 |
| 4.4.4.28 | <a href="#">LeGattRegisterService()</a>             | 65 |
| 4.4.4.29 | <a href="#">LeGattSignedWriteNoRsp()</a>            | 65 |
| 4.4.4.30 | <a href="#">LeGattStopCurrentProcedure()</a>        | 66 |
| 4.4.4.31 | <a href="#">LeGattWriteCharVal()</a>                | 66 |
| 4.4.4.32 | <a href="#">LeGattWriteCharValReliable()</a>        | 67 |
| 4.4.4.33 | <a href="#">LeGattWriteLongCharVal()</a>            | 67 |
| 4.4.4.34 | <a href="#">LeGattWriteNoRsp()</a>                  | 68 |
| 4.4.5    | <a href="#">Variable Documentation</a>              | 68 |
| 4.4.5.1  | <a href="#">gcCharacteristicUuid</a>                | 68 |
| 4.4.5.2  | <a href="#">gcCharAggregateUuid</a>                 | 68 |
| 4.4.5.3  | <a href="#">gcCharExtPropUuid</a>                   | 69 |

|          |  |    |
|----------|--|----|
| 4.4.5.4  | <a href="#">gcCharFormatUuid</a>               | 69 |
| 4.4.5.5  | <a href="#">gcCharUserDescUuid</a>             | 69 |
| 4.4.5.6  | <a href="#">gcClientCharConfigUuid</a>         | 69 |
| 4.4.5.7  | <a href="#">gcExtReportRefUuid</a>             | 69 |
| 4.4.5.8  | <a href="#">gcIncludeUuid</a>                  | 69 |
| 4.4.5.9  | <a href="#">gcPrimaryServiceUuid</a>           | 69 |
| 4.4.5.10 | <a href="#">gcReportRefUuid</a>                | 69 |
| 4.4.5.11 | <a href="#">gcSecondaryServiceUuid</a>         | 70 |
| 4.4.5.12 | <a href="#">gcServerCharConfigUuid</a>         | 70 |
| 4.4.5.13 | <a href="#">gcValidRangeUuid</a>               | 70 |
| 4.5      | <a href="#">BLE MSG APIs</a>                   | 71 |
| 4.5.1    | <a href="#">Detailed Description</a>           | 72 |
| 4.5.2    | <a href="#">Macro Definition Documentation</a> | 72 |
| 4.5.2.1  | <a href="#">LE_ATT_MSG_BASE</a>                | 72 |
| 4.5.2.2  | <a href="#">LE_CM_MSG_BASE</a>                 | 72 |
| 4.5.2.3  | <a href="#">LE_GATT_MSG_BASE</a>               | 73 |
| 4.5.2.4  | <a href="#">LE_HCI_MSG_BASE</a>                | 73 |
| 4.5.2.5  | <a href="#">LE_L2CAP_MSG_BASE</a>              | 73 |
| 4.5.2.6  | <a href="#">LE_SMP_MSG_BASE</a>                | 73 |
| 4.5.2.7  | <a href="#">LE_SYS_MSG_BASE</a>                | 73 |
| 4.5.2.8  | <a href="#">MESSAGE_ALLOCATE</a>               | 73 |
| 4.5.2.9  | <a href="#">MESSAGE_BULID</a>                  | 73 |
| 4.5.2.10 | <a href="#">MESSAGE_DATA_BULID</a>             | 74 |
| 4.5.2.11 | <a href="#">MESSAGE_OFFSET</a>                 | 74 |
| 4.5.2.12 | <a href="#">T_HOUR</a>                         | 74 |
| 4.5.2.13 | <a href="#">T_MIN</a>                          | 74 |
| 4.5.2.14 | <a href="#">T_SEC</a>                          | 74 |
| 4.5.3    | <a href="#">Typedef Documentation</a>          | 74 |
| 4.5.3.1  | <a href="#">MESSAGE</a>                        | 74 |
| 4.5.3.2  | <a href="#">MESSAGEID</a>                      | 75 |

|          |                                |    |
|----------|--------------------------------|----|
| 4.5.3.3  | MsgData                        | 75 |
| 4.5.3.4  | MsgLock                        | 75 |
| 4.5.3.5  | MSGLOCK                        | 75 |
| 4.5.3.6  | MSGSUBID                       | 75 |
| 4.5.3.7  | MSGTIMER                       | 75 |
| 4.5.3.8  | Task                           | 75 |
| 4.5.3.9  | TASK                           | 75 |
| 4.5.3.10 | TASKHANDLER                    | 76 |
| 4.5.3.11 | TASKPACK                       | 76 |
| 4.5.4    | Enumeration Type Documentation | 76 |
| 4.5.4.1  | anonymous enum                 | 76 |
| 4.5.5    | Function Documentation         | 76 |
| 4.5.5.1  | LeCancelAllMessage()           | 76 |
| 4.5.5.2  | LeCancelAllSubMessage()        | 77 |
| 4.5.5.3  | LeCancelFirstMessage()         | 77 |
| 4.5.5.4  | LeCancelFirstSubMessage()      | 78 |
| 4.5.5.5  | LeGetSubMsgId()                | 78 |
| 4.5.5.6  | LeHostCreateTask()             | 78 |
| 4.5.5.7  | LeHostMessageLoop()            | 79 |
| 4.5.5.8  | LeSendMessage()                | 79 |
| 4.5.5.9  | LeSendMessageAfter()           | 79 |
| 4.5.5.10 | LeSendMessageUnlock()          | 80 |
| 4.5.5.11 | LeSendSubMessage()             | 80 |
| 4.5.5.12 | LeSendSubMessageAfter()        | 81 |
| 4.5.5.13 | LeSendSubMessageUnlock()       | 81 |
| 4.6      | BLE SMP APIs                   | 83 |
| 4.6.1    | Detailed Description           | 84 |
| 4.6.2    | Macro Definition Documentation | 84 |
| 4.6.2.1  | LE_MAX_BOND_COUNT              | 84 |
| 4.6.2.2  | LE_SM_IO_CAP_DISP_ONLY         | 84 |

|          |  |    |
|----------|--|----|
| 4.6.2.3  | LE_SM_IO_CAP_DISP_YES_NO . . . . .       | 84 |
| 4.6.2.4  | LE_SM_IO_CAP_KEYBOARD_DISP . . . . .     | 85 |
| 4.6.2.5  | LE_SM_IO_CAP_KEYBOARD_ONLY . . . . .     | 85 |
| 4.6.2.6  | LE_SM_IO_CAP_NO_IO . . . . .             | 85 |
| 4.6.2.7  | LE_SM_PAIR_MITM_NO . . . . .             | 85 |
| 4.6.2.8  | LE_SM_PAIR_MITM_YES . . . . .            | 85 |
| 4.6.2.9  | LE_SM_PAIR_OOB_NO . . . . .              | 85 |
| 4.6.2.10 | LE_SM_PAIR_OOB_YES . . . . .             | 85 |
| 4.6.2.11 | LE_SM_PAIR_SC_NO . . . . .               | 85 |
| 4.6.2.12 | LE_SM_PAIR_SC_YES . . . . .              | 86 |
| 4.6.3    | Enumeration Type Documentation . . . . . | 86 |
| 4.6.3.1  | anonymous enum . . . . .                 | 86 |
| 4.6.3.2  | anonymous enum . . . . .                 | 86 |
| 4.6.4    | Function Documentation . . . . .         | 87 |
| 4.6.4.1  | LeSmpInit() . . . . .                    | 87 |
| 4.6.4.2  | LeSmpOobAuthDataRsp() . . . . .          | 87 |
| 4.6.4.3  | LeSmpOobPresent() . . . . .              | 87 |
| 4.6.4.4  | LeSmpPasskeyInput() . . . . .            | 88 |
| 4.6.4.5  | LeSmpScOobComputeConfirmVal() . . . . .  | 88 |
| 4.6.4.6  | LeSmpScOobDataRsp() . . . . .            | 89 |
| 4.6.4.7  | LeSmpSecurityReq() . . . . .             | 89 |
| 4.6.4.8  | LeSmpSecurityRsp() . . . . .             | 89 |
| 4.6.4.9  | LeSmpSetDefaultConfig() . . . . .        | 90 |
| 4.6.4.10 | LeSmpUserConfirmRsp() . . . . .          | 90 |
| 4.7      | WIFI APIs . . . . .                      | 91 |
| 4.7.1    | Detailed Description . . . . .           | 92 |
| 4.7.2    | Macro Definition Documentation . . . . . | 92 |
| 4.7.2.1  | WIFI_BEACON_INTERVAL_LENGTH . . . . .    | 92 |
| 4.7.2.2  | WIFI_CAPABILITY_INFO_LENGTH . . . . .    | 92 |
| 4.7.2.3  | WIFI_LENGTH_802_11 . . . . .             | 92 |

|         |   |     |
|---------|---|-----|
| 4.7.2.4 | WIFI_LENGTH_PASSPHRASE . . . . .                | 93  |
| 4.7.2.5 | WIFI_MAC_ADDRESS_LENGTH . . . . .               | 93  |
| 4.7.2.6 | WIFI_MAX_LENGTH_OF_SSID . . . . .               | 93  |
| 4.7.2.7 | WIFI_MAX_SCAN_AP_NUM . . . . .                  | 93  |
| 4.7.2.8 | WIFI_MAX_SUPPORTED_RATES . . . . .              | 93  |
| 4.7.3   | Typedef Documentation . . . . .                 | 93  |
| 4.7.3.1 | wifi_event_notify_cb_t . . . . .                | 93  |
| 4.7.4   | Function Documentation . . . . .                | 94  |
| 4.7.4.1 | wifi_event_process_handler() . . . . .          | 94  |
| 4.7.4.2 | wifi_install_default_event_handlers() . . . . . | 94  |
| 4.7.4.3 | wifi_register_event_handler() . . . . .         | 95  |
| 4.8     | WIFI Common APIs . . . . .                      | 96  |
| 4.8.1   | Detailed Description . . . . .                  | 96  |
| 4.8.2   | Typedef Documentation . . . . .                 | 96  |
| 4.8.2.1 | wifi_event_cb_t . . . . .                       | 96  |
| 4.8.3   | Function Documentation . . . . .                | 97  |
| 4.8.3.1 | wifi_event_loop_init() . . . . .                | 97  |
| 4.8.3.2 | wifi_event_loop_send() . . . . .                | 98  |
| 4.8.3.3 | wifi_event_loop_set_cb() . . . . .              | 98  |
| 4.8.3.4 | wifi_event_process_handler() . . . . .          | 99  |
| 4.9     | WIFI STA APIs . . . . .                         | 100 |
| 4.9.1   | Detailed Description . . . . .                  | 101 |
| 4.9.2   | Typedef Documentation . . . . .                 | 101 |
| 4.9.2.1 | wifi_event_handler_t . . . . .                  | 102 |
| 4.9.2.2 | wifi_init_complete_cb_t . . . . .               | 102 |
| 4.9.2.3 | wifi_result_t . . . . .                         | 102 |
| 4.9.3   | Function Documentation . . . . .                | 102 |
| 4.9.3.1 | wifi_auto_connect_del_ap_info() . . . . .       | 102 |
| 4.9.3.2 | wifi_auto_connect_get_ap_info() . . . . .       | 103 |
| 4.9.3.3 | wifi_auto_connect_get_ap_num() . . . . .        | 103 |

|          |  |     |
|----------|--|-----|
| 4.9.3.4  | wifi_auto_connect_get_mode()               | 104 |
| 4.9.3.5  | wifi_auto_connect_init()                   | 104 |
| 4.9.3.6  | wifi_auto_connect_set_ap_num()             | 104 |
| 4.9.3.7  | wifi_auto_connect_set_mode()               | 104 |
| 4.9.3.8  | wifi_auto_connect_start()                  | 106 |
| 4.9.3.9  | wifi_config_get_bandwidth()                | 106 |
| 4.9.3.10 | wifi_config_get_bssid()                    | 107 |
| 4.9.3.11 | wifi_config_get_channel()                  | 107 |
| 4.9.3.12 | wifi_config_get_mac_address()              | 108 |
| 4.9.3.13 | wifi_config_get_ssid()                     | 108 |
| 4.9.3.14 | wifi_config_set_bandwidth()                | 109 |
| 4.9.3.15 | wifi_config_set_bssid()                    | 109 |
| 4.9.3.16 | wifi_config_set_channel()                  | 109 |
| 4.9.3.17 | wifi_config_set_mac_address()              | 111 |
| 4.9.3.18 | wifi_config_set_ssid()                     | 111 |
| 4.9.3.19 | wifi_connection_connect()                  | 112 |
| 4.9.3.20 | wifi_connection_disconnect_ap()            | 112 |
| 4.9.3.21 | wifi_connection_disconnect_sta()           | 113 |
| 4.9.3.22 | wifi_connection_get_rssi()                 | 113 |
| 4.9.3.23 | wifi_connection_register_event_handler()   | 113 |
| 4.9.3.24 | wifi_connection_unregister_event_handler() | 114 |
| 4.9.3.25 | wifi_deinit()                              | 114 |
| 4.9.3.26 | wifi_fast_connect_get_mode()               | 115 |
| 4.9.3.27 | wifi_fast_connect_set_mode()               | 115 |
| 4.9.3.28 | wifi_fast_connect_start()                  | 116 |
| 4.9.3.29 | wifi_get_config()                          | 116 |
| 4.9.3.30 | wifi_get_fast_conn_mode()                  | 116 |
| 4.9.3.31 | wifi_init()                                | 117 |
| 4.9.3.32 | wifi_scan_get_ap_list()                    | 117 |
| 4.9.3.33 | wifi_scan_get_ap_num()                     | 117 |

|          |                                     |            |
|----------|-------------------------------------|------------|
| 4.9.3.34 | wifi_scan_get_ap_records()          | 118        |
| 4.9.3.35 | wifi_scan_scan_stop()               | 118        |
| 4.9.3.36 | wifi_scan_start()                   | 119        |
| 4.9.3.37 | wifi_set_config()                   | 119        |
| 4.9.3.38 | wifi_sta_get_ap_info()              | 120        |
| 4.9.3.39 | wifi_start()                        | 120        |
| 4.9.3.40 | wifi_stop()                         | 120        |
| 4.10     | Enumeration                         | 121        |
| 4.10.1   | Detailed Description                | 121        |
| 4.10.2   | Enumeration Type Documentation      | 121        |
| 4.10.2.1 | wifi_auth_mode_t                    | 121        |
| 4.10.2.2 | wifi_bandwidth_t                    | 122        |
| 4.10.2.3 | wifi_cipher_type_t                  | 122        |
| 4.10.2.4 | wifi_event_t                        | 122        |
| 4.10.2.5 | wifi_mode_t                         | 123        |
| 4.10.2.6 | wifi_reason_code_t                  | 123        |
| 4.10.2.7 | wifi_scan_method_t                  | 124        |
| 4.10.2.8 | wifi_scan_type_t                    | 124        |
| 4.10.2.9 | wifi_sort_method_t                  | 126        |
| <b>5</b> | <b>Data Structure Documentation</b> | <b>127</b> |
| 5.1      | auto_conn_info_t Struct Reference   | 127        |
| 5.1.1    | Field Documentation                 | 127        |
| 5.1.1.1  | ap_channel                          | 127        |
| 5.1.1.2  | beacon_interval                     | 128        |
| 5.1.1.3  | bssid                               | 128        |
| 5.1.1.4  | capabilities                        | 128        |
| 5.1.1.5  | dtim_prod                           | 128        |
| 5.1.1.6  | fast_connect                        | 128        |
| 5.1.1.7  | free_ocpy                           | 128        |
| 5.1.1.8  | hid_ssid                            | 128        |



|          |  |     |
|----------|--|-----|
| 5.1.1.9  | <a href="#">latest_beacon_rx_time</a>                            | 128 |
| 5.1.1.10 | <a href="#">passphrase</a>                                       | 129 |
| 5.1.1.11 | <a href="#">psk</a>  | 129 |
| 5.1.1.12 | <a href="#">rsn_ie</a>   | 129 |
| 5.1.1.13 | <a href="#">rssi</a>   | 129 |
| 5.1.1.14 | <a href="#">ssid</a>   | 129 |
| 5.1.1.15 | <a href="#">supported_rates</a>                                  | 129 |
| 5.1.1.16 | <a href="#">wpa_data</a>   | 129 |
| 5.1.1.17 | <a href="#">wpa_ie</a>   | 129 |
| 5.2      | <a href="#">auto_connect_cfg_t Struct Reference</a>              | 130 |
| 5.2.1    | <a href="#">Field Documentation</a>                              | 130 |
| 5.2.1.1  | <a href="#">flag</a>   | 130 |
| 5.2.1.2  | <a href="#">front</a>  | 130 |
| 5.2.1.3  | <a href="#">max_save_num</a>                                     | 130 |
| 5.2.1.4  | <a href="#">pFCInfo</a>  | 130 |
| 5.2.1.5  | <a href="#">rear</a>   | 131 |
| 5.2.1.6  | <a href="#">retryCount</a>                                       | 131 |
| 5.2.1.7  | <a href="#">targetIdx</a>  | 131 |
| 5.2.1.8  | <a href="#">uFCapNum</a>   | 131 |
| 5.3      | <a href="#">event_msg_t Struct Reference</a>                     | 131 |
| 5.3.1    | <a href="#">Detailed Description</a>                             | 131 |
| 5.3.2    | <a href="#">Field Documentation</a>                              | 131 |
| 5.3.2.1  | <a href="#">event</a>  | 132 |
| 5.3.2.2  | <a href="#">length</a>   | 132 |
| 5.3.2.3  | <a href="#">param</a>  | 132 |
| 5.4      | <a href="#">LE_BT_ADDR_T Struct Reference</a>                    | 132 |
| 5.4.1    | <a href="#">Field Documentation</a>                              | 132 |
| 5.4.1.1  | <a href="#">addr</a>   | 132 |
| 5.4.1.2  | <a href="#">type</a>   | 132 |
| 5.5      | <a href="#">LE_CM_CONNECTION_COMPLETE_IND_T Struct Reference</a> | 133 |

|         |   |     |
|---------|---|-----|
| 5.5.1   | Field Documentation                                   | 133 |
| 5.5.1.1 | conn_hdl  | 133 |
| 5.5.1.2 | conn_interval   | 133 |
| 5.5.1.3 | conn_latency  | 133 |
| 5.5.1.4 | dev_id  | 133 |
| 5.5.1.5 | peer_addr   | 134 |
| 5.5.1.6 | peer_addr_type  | 134 |
| 5.5.1.7 | role  | 134 |
| 5.5.1.8 | status  | 134 |
| 5.5.1.9 | supervision_timeout                                   | 134 |
| 5.6     | LE_CM_MSG_ADVERTISE_REPORT_IND_T Struct Reference     | 134 |
| 5.6.1   | Field Documentation                                   | 135 |
| 5.6.1.1 | addr  | 135 |
| 5.6.1.2 | addr_type   | 135 |
| 5.6.1.3 | data  | 135 |
| 5.6.1.4 | event_type  | 135 |
| 5.6.1.5 | len   | 135 |
| 5.6.1.6 | rss   | 135 |
| 5.7     | LE_CM_MSG_CONN_PARA_REQ_T Struct Reference            | 135 |
| 5.7.1   | Field Documentation                                   | 136 |
| 5.7.1.1 | conn_hdl  | 136 |
| 5.7.1.2 | itv_max   | 136 |
| 5.7.1.3 | itv_min   | 136 |
| 5.7.1.4 | latency   | 136 |
| 5.7.1.5 | sv_tmo  | 136 |
| 5.8     | LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T Struct Reference | 136 |
| 5.8.1   | Field Documentation                                   | 137 |
| 5.8.1.1 | conn_hdl  | 137 |
| 5.8.1.2 | interval  | 137 |
| 5.8.1.3 | latency   | 137 |

|          |  |     |
|----------|--|-----|
| 5.8.1.4  | status   | 137 |
| 5.8.1.5  | supervision_timeout                                  | 137 |
| 5.9      | LE_CM_MSG_DATA_LEN_CHANGE_IND_T Struct Reference     | 137 |
| 5.9.1    | Field Documentation                                  | 138 |
| 5.9.1.1  | conn_hdl   | 138 |
| 5.9.1.2  | max_rx_octets  | 138 |
| 5.9.1.3  | max_rx_time  | 138 |
| 5.9.1.4  | max_tx_octets  | 138 |
| 5.9.1.5  | max_tx_time  | 138 |
| 5.10     | LE_CM_MSG_DIRECT_ADV_REPORT_IND_T Struct Reference   | 138 |
| 5.10.1   | Field Documentation                                  | 139 |
| 5.10.1.1 | direct_addr  | 139 |
| 5.10.1.2 | direct_addr_type                                     | 139 |
| 5.10.1.3 | peer_addr  | 139 |
| 5.10.1.4 | peer_addr_type                                       | 139 |
| 5.10.1.5 | rsi  | 139 |
| 5.11     | LE_CM_MSG_DISCONNECT_COMPLETE_IND_T Struct Reference | 139 |
| 5.11.1   | Field Documentation                                  | 140 |
| 5.11.1.1 | conn_hdl   | 140 |
| 5.11.1.2 | reason   | 140 |
| 5.11.1.3 | status   | 140 |
| 5.12     | LE_CM_MSG_ENCRYPTION_CHANGE_IND_T Struct Reference   | 140 |
| 5.12.1   | Field Documentation                                  | 140 |
| 5.12.1.1 | conn_hdl   | 141 |
| 5.12.1.2 | devid  | 141 |
| 5.12.1.3 | enabled  | 141 |
| 5.12.1.4 | status   | 141 |
| 5.13     | LE_CM_MSG_ENCRYPTION_REFRESH_IND_T Struct Reference  | 141 |
| 5.13.1   | Field Documentation                                  | 141 |
| 5.13.1.1 | conn_hdl   | 141 |

|          |   |     |
|----------|---|-----|
| 5.13.1.2 | devid   | 142 |
| 5.13.1.3 | enabled   | 142 |
| 5.13.1.4 | status  | 142 |
| 5.14     | LE_CM_MSG_INIT_COMPLETE_CFM_T Struct Reference            | 142 |
| 5.14.1   | Field Documentation                                       | 142 |
| 5.14.1.1 | status  | 142 |
| 5.15     | LE_CM_MSG_LTK_REQ_IND_T Struct Reference                  | 142 |
| 5.15.1   | Field Documentation                                       | 143 |
| 5.15.1.1 | conn_hdl  | 143 |
| 5.15.1.2 | devid   | 143 |
| 5.15.1.3 | ediv  | 143 |
| 5.15.1.4 | rand  | 143 |
| 5.16     | LE_CM_MSG_READ_ADV_TX_POWER_CFM_T Struct Reference        | 143 |
| 5.16.1   | Field Documentation                                       | 144 |
| 5.16.1.1 | pwr_level   | 144 |
| 5.16.1.2 | status  | 144 |
| 5.17     | LE_CM_MSG_READ_BD_ADDR_CFM_T Struct Reference             | 144 |
| 5.17.1   | Field Documentation                                       | 144 |
| 5.17.1.1 | bd_addr   | 144 |
| 5.17.1.2 | status  | 144 |
| 5.18     | LE_CM_MSG_READ_CHANNEL_MAP_CFM_T Struct Reference         | 145 |
| 5.18.1   | Field Documentation                                       | 145 |
| 5.18.1.1 | ch_map  | 145 |
| 5.18.1.2 | conn_hdl  | 145 |
| 5.18.1.3 | status  | 145 |
| 5.19     | LE_CM_MSG_READ_RESOLVING_LIST_SIZE_CFM_T Struct Reference | 145 |
| 5.19.1   | Field Documentation                                       | 145 |
| 5.19.1.1 | size  | 146 |
| 5.19.1.2 | status  | 146 |
| 5.20     | LE_CM_MSG_READ_RSSI_CFM_T Struct Reference                | 146 |

|          |   |     |
|----------|---|-----|
| 5.20.1   | Field Documentation                                   | 146 |
| 5.20.1.1 | conn_hdl  | 146 |
| 5.20.1.2 | rsi   | 146 |
| 5.20.1.3 | status  | 146 |
| 5.21     | LE_CM_MSG_READ_TX_POWER_CFM_T Struct Reference        | 147 |
| 5.21.1   | Field Documentation                                   | 147 |
| 5.21.1.1 | conn_hdl  | 147 |
| 5.21.1.2 | status  | 147 |
| 5.21.1.3 | tx_power  | 147 |
| 5.22     | LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM_T Struct Reference | 147 |
| 5.22.1   | Field Documentation                                   | 147 |
| 5.22.1.1 | size  | 148 |
| 5.22.1.2 | status  | 148 |
| 5.23     | LE_CM_MSG_SET_DATA_LENGTH_CFM_T Struct Reference      | 148 |
| 5.23.1   | Field Documentation                                   | 148 |
| 5.23.1.1 | conn_hdl  | 148 |
| 5.23.1.2 | status  | 148 |
| 5.24     | LE_CM_MSG_SET_DISCONNECT_CFM_T Struct Reference       | 148 |
| 5.24.1   | Field Documentation                                   | 149 |
| 5.24.1.1 | handle  | 149 |
| 5.24.1.2 | status  | 149 |
| 5.25     | LE_CM_MSG_SIGNAL_UPDATE_REQ_T Struct Reference        | 149 |
| 5.25.1   | Field Documentation                                   | 149 |
| 5.25.1.1 | conn_hdl  | 149 |
| 5.25.1.2 | identifier  | 150 |
| 5.25.1.3 | interval_max  | 150 |
| 5.25.1.4 | interval_min  | 150 |
| 5.25.1.5 | slave_latency   | 150 |
| 5.25.1.6 | timeout_multiplier                                    | 150 |
| 5.26     | LE_CM_REQ_STATUS_T Struct Reference                   | 150 |

|          |   |     |
|----------|---|-----|
| 5.26.1   | Field Documentation                         | 150 |
| 5.26.1.1 | status                                      | 151 |
| 5.27     | LE_CONN_PARAM_T Struct Reference            | 151 |
| 5.27.1   | Field Documentation                         | 151 |
| 5.27.1.1 | itv_max                                     | 151 |
| 5.27.1.2 | itv_min                                     | 151 |
| 5.27.1.3 | latency                                     | 151 |
| 5.27.1.4 | sv_timeout                                  | 151 |
| 5.28     | LE_GAP_ADVERTISING_PARAM_T Struct Reference | 152 |
| 5.28.1   | Field Documentation                         | 152 |
| 5.28.1.1 | channel_map                                 | 152 |
| 5.28.1.2 | filter_policy                               | 152 |
| 5.28.1.3 | interval_max                                | 152 |
| 5.28.1.4 | interval_min                                | 152 |
| 5.28.1.5 | own_addr_type                               | 153 |
| 5.28.1.6 | peer_addr                                   | 153 |
| 5.28.1.7 | peer_addr_type                              | 153 |
| 5.28.1.8 | type  | 153 |
| 5.29     | LE_GAP_CONN_PARAM_T Struct Reference        | 153 |
| 5.29.1   | Field Documentation                         | 153 |
| 5.29.1.1 | interval_max                                | 153 |
| 5.29.1.2 | interval_min                                | 154 |
| 5.29.1.3 | latency                                     | 154 |
| 5.29.1.4 | supervision_timeout                         | 154 |
| 5.30     | LE_GAP_SCAN_PARAM_T Struct Reference        | 154 |
| 5.30.1   | Field Documentation                         | 154 |
| 5.30.1.1 | filter_policy                               | 154 |
| 5.30.1.2 | interval                                    | 154 |
| 5.30.1.3 | own_addr_type                               | 155 |
| 5.30.1.4 | type  | 155 |

|          |   |     |
|----------|---|-----|
| 5.30.1.5 | window  | 155 |
| 5.31     | LE_GATT_ATTR_T Struct Reference                         | 155 |
| 5.31.1   | Field Documentation                                     | 155 |
| 5.31.1.1 | format  | 155 |
| 5.31.1.2 | handle  | 156 |
| 5.31.1.3 | len   | 156 |
| 5.31.1.4 | maxLen  | 156 |
| 5.31.1.5 | permit  | 156 |
| 5.31.1.6 | pUuid   | 156 |
| 5.31.1.7 | pVal  | 156 |
| 5.32     | LE_GATT_MSG_ACCESS_READ_IND_T Struct Reference          | 156 |
| 5.32.1   | Field Documentation                                     | 157 |
| 5.32.1.1 | conn_hdl  | 157 |
| 5.32.1.2 | devid   | 157 |
| 5.32.1.3 | handle  | 157 |
| 5.32.1.4 | offset  | 157 |
| 5.33     | LE_GATT_MSG_ACCESS_WRITE_IND_T Struct Reference         | 157 |
| 5.33.1   | Field Documentation                                     | 157 |
| 5.33.1.1 | conn_hdl  | 158 |
| 5.33.1.2 | devid   | 158 |
| 5.33.1.3 | flag  | 158 |
| 5.33.1.4 | handle  | 158 |
| 5.33.1.5 | len   | 158 |
| 5.33.1.6 | offset  | 158 |
| 5.33.1.7 | pVal  | 158 |
| 5.34     | LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND_T Struct Reference | 158 |
| 5.34.1   | Field Documentation                                     | 159 |
| 5.34.1.1 | conn_hdl  | 159 |
| 5.34.1.2 | devid   | 159 |
| 5.34.1.3 | format  | 159 |

|          |   |     |
|----------|---|-----|
| 5.34.1.4 | handle  | 159 |
| 5.34.1.5 | uuid  | 159 |
| 5.35     | LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND_T Struct Reference | 159 |
| 5.35.1   | Field Documentation   | 160 |
| 5.35.1.1 | conn_hdl  | 160 |
| 5.35.1.2 | devid   | 160 |
| 5.35.1.3 | format  | 160 |
| 5.35.1.4 | handle  | 160 |
| 5.35.1.5 | property  | 160 |
| 5.35.1.6 | uuid  | 161 |
| 5.35.1.7 | val_hdl   | 161 |
| 5.36     | LE_GATT_MSG_CHARACTERISTIC_VAL_IND_T Struct Reference       | 161 |
| 5.36.1   | Field Documentation   | 161 |
| 5.36.1.1 | att_err   | 161 |
| 5.36.1.2 | conn_hdl  | 161 |
| 5.36.1.3 | devid   | 162 |
| 5.36.1.4 | handle  | 162 |
| 5.36.1.5 | len   | 162 |
| 5.36.1.6 | offset  | 162 |
| 5.36.1.7 | val   | 162 |
| 5.37     | LE_GATT_MSG_CONFIRMATION_CFM_T Struct Reference             | 162 |
| 5.37.1   | Field Documentation   | 162 |
| 5.37.1.1 | conn_hdl  | 163 |
| 5.37.1.2 | devid   | 163 |
| 5.37.1.3 | handle  | 163 |
| 5.38     | LE_GATT_MSG_EXCHANGE_MTU_CFM_T Struct Reference             | 163 |
| 5.38.1   | Field Documentation   | 163 |
| 5.38.1.1 | conn_hdl  | 163 |
| 5.38.1.2 | current_rx_mtu  | 163 |
| 5.38.1.3 | devid   | 164 |



|  |     |
|--|-----|
| 5.39 LE_GATT_MSG_EXCHANGE_MTU_IND_T Struct Reference . . . . .             | 164 |
| 5.39.1 Field Documentation . . . . .                                       | 164 |
| 5.39.1.1 client_rx_mtu . . . . .   | 164 |
| 5.39.1.2 conn_hdl . . . . .  | 164 |
| 5.39.1.3 devid . . . . .   | 164 |
| 5.40 LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM_T Struct Reference . . . . .   | 164 |
| 5.40.1 Field Documentation . . . . .                                       | 165 |
| 5.40.1.1 att_err . . . . .   | 165 |
| 5.40.1.2 conn_hdl . . . . .  | 165 |
| 5.40.1.3 devid . . . . .   | 165 |
| 5.40.1.4 err_hdl . . . . .   | 165 |
| 5.40.1.5 status . . . . .  | 165 |
| 5.41 LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM_T Struct Reference . . . . .       | 165 |
| 5.41.1 Field Documentation . . . . .                                       | 166 |
| 5.41.1.1 att_err . . . . .   | 166 |
| 5.41.1.2 conn_hdl . . . . .  | 166 |
| 5.41.1.3 devid . . . . .   | 166 |
| 5.41.1.4 handle . . . . .  | 166 |
| 5.41.1.5 status . . . . .  | 166 |
| 5.42 LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_CFM_T Struct Reference . . . . . | 166 |
| 5.42.1 Field Documentation . . . . .                                       | 167 |
| 5.42.1.1 att_err . . . . .   | 167 |
| 5.42.1.2 conn_hdl . . . . .  | 167 |
| 5.42.1.3 devid . . . . .   | 167 |
| 5.42.1.4 handle . . . . .  | 167 |
| 5.42.1.5 status . . . . .  | 167 |
| 5.43 LE_GATT_MSG_FIND_CHARACTERISTIC_CFM_T Struct Reference . . . . .      | 167 |
| 5.43.1 Field Documentation . . . . .                                       | 168 |
| 5.43.1.1 att_err . . . . .   | 168 |
| 5.43.1.2 conn_hdl . . . . .  | 168 |

|          |   |     |
|----------|---|-----|
| 5.43.1.3 | devid   | 168 |
| 5.43.1.4 | handle  | 168 |
| 5.43.1.5 | status  | 168 |
| 5.44     | LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM_T Struct Reference        | 168 |
| 5.44.1   | Field Documentation   | 169 |
| 5.44.1.1 | att_err   | 169 |
| 5.44.1.2 | conn_hdl  | 169 |
| 5.44.1.3 | devid   | 169 |
| 5.44.1.4 | handle  | 169 |
| 5.44.1.5 | status  | 169 |
| 5.45     | LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM_T Struct Reference | 169 |
| 5.45.1   | Field Documentation   | 170 |
| 5.45.1.1 | att_err   | 170 |
| 5.45.1.2 | conn_hdl  | 170 |
| 5.45.1.3 | devid   | 170 |
| 5.45.1.4 | handle  | 170 |
| 5.45.1.5 | status  | 170 |
| 5.46     | LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T Struct Reference         | 170 |
| 5.46.1   | Field Documentation   | 171 |
| 5.46.1.1 | conn_hdl  | 171 |
| 5.46.1.2 | devid   | 171 |
| 5.46.1.3 | end_hdl   | 171 |
| 5.46.1.4 | format  | 171 |
| 5.46.1.5 | handle  | 171 |
| 5.46.1.6 | start_hdl   | 172 |
| 5.46.1.7 | uuid  | 172 |
| 5.47     | LE_GATT_MSG_INDICATE_IND_T Struct Reference                     | 172 |
| 5.47.1   | Field Documentation   | 172 |
| 5.47.1.1 | conn_hdl  | 172 |
| 5.47.1.2 | devid   | 172 |

|          |   |     |
|----------|---|-----|
| 5.47.1.3 | handle  | 172 |
| 5.47.1.4 | len   | 173 |
| 5.47.1.5 | val   | 173 |
| 5.48     | LE_GATT_MSG_NOTIFY_CFM_T Struct Reference                 | 173 |
| 5.48.1   | Field Documentation                                       | 173 |
| 5.48.1.1 | conn_hdl  | 173 |
| 5.48.1.2 | devid   | 173 |
| 5.48.1.3 | handle  | 173 |
| 5.48.1.4 | status  | 174 |
| 5.49     | LE_GATT_MSG_NOTIFY_IND_T Struct Reference                 | 174 |
| 5.49.1   | Field Documentation                                       | 174 |
| 5.49.1.1 | conn_hdl  | 174 |
| 5.49.1.2 | devid   | 174 |
| 5.49.1.3 | handle  | 174 |
| 5.49.1.4 | len   | 174 |
| 5.49.1.5 | val   | 175 |
| 5.50     | LE_GATT_MSG_OPERATION_TIMEOUT_T Struct Reference          | 175 |
| 5.50.1   | Field Documentation                                       | 175 |
| 5.50.1.1 | att_op  | 175 |
| 5.50.1.2 | conn_hdl  | 175 |
| 5.50.1.3 | devid   | 175 |
| 5.51     | LE_GATT_MSG_PREPARE_WRITE_RELIABLE_CFM_T Struct Reference | 175 |
| 5.51.1   | Field Documentation                                       | 176 |
| 5.51.1.1 | att_err   | 176 |
| 5.51.1.2 | conn_hdl  | 176 |
| 5.51.1.3 | devid   | 176 |
| 5.51.1.4 | handle  | 176 |
| 5.51.1.5 | status  | 176 |
| 5.52     | LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_CFM_T Struct Reference  | 176 |
| 5.52.1   | Field Documentation                                       | 177 |

|          |  |     |
|----------|--|-----|
| 5.52.1.1 | att_err  | 177 |
| 5.52.1.2 | conn_hdl   | 177 |
| 5.52.1.3 | devid  | 177 |
| 5.52.1.4 | handle   | 177 |
| 5.52.1.5 | status   | 177 |
| 5.53     | LE_GATT_MSG_READ_CHARACTERISTIC_VALUE_CFM_T Struct Reference | 177 |
| 5.53.1   | Field Documentation  | 178 |
| 5.53.1.1 | att_err  | 178 |
| 5.53.1.2 | conn_hdl   | 178 |
| 5.53.1.3 | devid  | 178 |
| 5.53.1.4 | handle   | 178 |
| 5.53.1.5 | status   | 178 |
| 5.54     | LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM_T Struct Reference        | 178 |
| 5.54.1   | Field Documentation  | 179 |
| 5.54.1.1 | att_err  | 179 |
| 5.54.1.2 | conn_hdl   | 179 |
| 5.54.1.3 | devid  | 179 |
| 5.54.1.4 | handle   | 179 |
| 5.54.1.5 | status   | 179 |
| 5.55     | LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL_CFM_T Struct Reference    | 179 |
| 5.55.1   | Field Documentation  | 180 |
| 5.55.1.1 | att_err  | 180 |
| 5.55.1.2 | conn_hdl   | 180 |
| 5.55.1.3 | devid  | 180 |
| 5.55.1.4 | err_hdl  | 180 |
| 5.55.1.5 | len  | 180 |
| 5.55.1.6 | status   | 181 |
| 5.55.1.7 | val  | 181 |
| 5.56     | LE_GATT_MSG_SERVICE_INFO_IND_T Struct Reference              | 181 |
| 5.56.1   | Field Documentation  | 181 |

|          |  |     |
|----------|--|-----|
| 5.56.1.1 | conn_hdl   | 181 |
| 5.56.1.2 | devid  | 181 |
| 5.56.1.3 | end_hdl  | 181 |
| 5.56.1.4 | format   | 182 |
| 5.56.1.5 | start_hdl  | 182 |
| 5.56.1.6 | uuid   | 182 |
| 5.57     | LE_GATT_MSG_SIGNED_WRITE_CFM_T Struct Reference            | 182 |
| 5.57.1   | Field Documentation  | 182 |
| 5.57.1.1 | conn_hdl   | 182 |
| 5.57.1.2 | devid  | 182 |
| 5.57.1.3 | handle   | 183 |
| 5.57.1.4 | status   | 183 |
| 5.58     | LE_GATT_MSG_WRITE_CHAR_VAL_RELIABLE_CFM_T Struct Reference | 183 |
| 5.58.1   | Field Documentation  | 183 |
| 5.58.1.1 | att_err  | 183 |
| 5.58.1.2 | conn_hdl   | 183 |
| 5.58.1.3 | devid  | 183 |
| 5.58.1.4 | handle   | 184 |
| 5.58.1.5 | status   | 184 |
| 5.59     | LE_GATT_MSG_WRITE_CHAR_VALUE_CFM_T Struct Reference        | 184 |
| 5.59.1   | Field Documentation  | 184 |
| 5.59.1.1 | att_err  | 184 |
| 5.59.1.2 | conn_hdl   | 184 |
| 5.59.1.3 | devid  | 184 |
| 5.59.1.4 | handle   | 185 |
| 5.59.1.5 | status   | 185 |
| 5.60     | LE_GATT_MSG_WRITE_LONG_CHAR_VALUE_CFM_T Struct Reference   | 185 |
| 5.60.1   | Field Documentation  | 185 |
| 5.60.1.1 | att_err  | 185 |
| 5.60.1.2 | conn_hdl   | 185 |

|          |  |     |
|----------|--|-----|
| 5.60.1.3 | devid  | 185 |
| 5.60.1.4 | handle   | 186 |
| 5.60.1.5 | status   | 186 |
| 5.61     | LE_GATT_MSG_WRITE_NO_RSP_CFM_T Struct Reference      | 186 |
| 5.61.1   | Field Documentation                                  | 186 |
| 5.61.1.1 | conn_hdl   | 186 |
| 5.61.1.2 | devid  | 186 |
| 5.61.1.3 | handle   | 186 |
| 5.61.1.4 | status   | 187 |
| 5.62     | LE_GATT_SERVICE_T Struct Reference                   | 187 |
| 5.62.1   | Field Documentation                                  | 187 |
| 5.62.1.1 | endHdl   | 187 |
| 5.62.1.2 | pAttr  | 187 |
| 5.62.1.3 | startHdl   | 187 |
| 5.62.1.4 | svc_id   | 187 |
| 5.63     | LE_SMP_MSG_ENCRYPTION_CHANGE_IND_T Struct Reference  | 188 |
| 5.63.1   | Field Documentation                                  | 188 |
| 5.63.1.1 | conn_hdl   | 188 |
| 5.63.1.2 | enable   | 188 |
| 5.64     | LE_SMP_MSG_ENCRYPTION_REFRESH_IND_T Struct Reference | 188 |
| 5.64.1   | Field Documentation                                  | 188 |
| 5.64.1.1 | conn_hdl   | 188 |
| 5.64.1.2 | status   | 189 |
| 5.65     | LE_SMP_MSG_OOB_DATA_REQUEST_IND_T Struct Reference   | 189 |
| 5.65.1   | Field Documentation                                  | 189 |
| 5.65.1.1 | conn_hdl   | 189 |
| 5.66     | LE_SMP_MSG_PAIRING_ACTION_IND_T Struct Reference     | 189 |
| 5.66.1   | Field Documentation                                  | 189 |
| 5.66.1.1 | action   | 189 |
| 5.66.1.2 | conn_hdl   | 190 |

|          |  |     |
|----------|--|-----|
| 5.66.1.3 | lost_bond  | 190 |
| 5.66.1.4 | sc   | 190 |
| 5.67     | LE_SMP_MSG_PAIRING_COMPLETE_IND_T Struct Reference       | 190 |
| 5.67.1   | Field Documentation                                      | 190 |
| 5.67.1.1 | authenticated  | 190 |
| 5.67.1.2 | bonded   | 190 |
| 5.67.1.3 | conn_hdl   | 191 |
| 5.67.1.4 | peer_id_addr   | 191 |
| 5.67.1.5 | sc   | 191 |
| 5.67.1.6 | status   | 191 |
| 5.68     | LE_SMP_MSG_PASSKEY_DISPLAY_IND_T Struct Reference        | 191 |
| 5.68.1   | Field Documentation                                      | 191 |
| 5.68.1.1 | conn_hdl   | 191 |
| 5.68.1.2 | passkey  | 192 |
| 5.69     | LE_SMP_MSG_PASSKEY_INPUT_IND_T Struct Reference          | 192 |
| 5.69.1   | Field Documentation                                      | 192 |
| 5.69.1.1 | conn_hdl   | 192 |
| 5.70     | LE_SMP_MSG_SC_OOB_DATA_REQUEST_IND_T Struct Reference    | 192 |
| 5.70.1   | Field Documentation                                      | 192 |
| 5.70.1.1 | conn_hdl   | 192 |
| 5.71     | LE_SMP_MSG_SLAVE_SECURITY_REQUEST_IND_T Struct Reference | 193 |
| 5.71.1   | Field Documentation                                      | 193 |
| 5.71.1.1 | bondable   | 193 |
| 5.71.1.2 | conn_hdl   | 193 |
| 5.71.1.3 | keypress   | 193 |
| 5.71.1.4 | mitm   | 193 |
| 5.71.1.5 | sc   | 193 |
| 5.72     | LE_SMP_MSG_USER_CONFIRM_IND_T Struct Reference           | 194 |
| 5.72.1   | Field Documentation                                      | 194 |
| 5.72.1.1 | confirm_num  | 194 |

|           |   |     |
|-----------|---|-----|
| 5.72.1.2  | conn_hdl  | 194 |
| 5.73      | LE_SMP_SC_OOB_DATA_T Struct Reference           | 194 |
| 5.73.1    | Field Documentation                             | 194 |
| 5.73.1.1  | confirm   | 194 |
| 5.73.1.2  | rand  | 195 |
| 5.74      | LE_SYS_MSG_BUF_OVERFLOW_T Struct Reference      | 195 |
| 5.74.1    | Field Documentation                             | 195 |
| 5.74.1.1  | conn_hdl  | 195 |
| 5.75      | mw_wifi_auto_connect_ap_info_t Struct Reference | 195 |
| 5.75.1    | Field Documentation                             | 196 |
| 5.75.1.1  | ap_channel                                      | 196 |
| 5.75.1.2  | beacon_interval                                 | 196 |
| 5.75.1.3  | bssid   | 196 |
| 5.75.1.4  | capabilities                                    | 196 |
| 5.75.1.5  | dtim_prod                                       | 196 |
| 5.75.1.6  | fast_connect                                    | 196 |
| 5.75.1.7  | free_ocpy                                       | 196 |
| 5.75.1.8  | hid_ssid  | 197 |
| 5.75.1.9  | latest_beacon_rx_time                           | 197 |
| 5.75.1.10 | passphrase                                      | 197 |
| 5.75.1.11 | psk   | 197 |
| 5.75.1.12 | rsn_ie  | 197 |
| 5.75.1.13 | rsi   | 197 |
| 5.75.1.14 | ssid  | 197 |
| 5.75.1.15 | supported_rates                                 | 197 |
| 5.75.1.16 | wpa_data  | 198 |
| 5.75.1.17 | wpa_ie  | 198 |
| 5.76      | MwFimAutoConnectCFG_t Struct Reference          | 198 |
| 5.76.1    | Field Documentation                             | 198 |
| 5.76.1.1  | flag  | 198 |



|           |  |     |
|-----------|--|-----|
| 5.76.1.2  | front                                    | 198 |
| 5.76.1.3  | max_save_num                             | 198 |
| 5.76.1.4  | rear                                     | 199 |
| 5.76.1.5  | targetIdx                                | 199 |
| 5.77      | T_RfCmd Struct Reference                 | 199 |
| 5.77.1    | Field Documentation                      | 199 |
| 5.77.1.1  | iArgc                                    | 199 |
| 5.77.1.2  | saArgv                                   | 199 |
| 5.77.1.3  | u32Type                                  | 199 |
| 5.78      | T_RfEvt Struct Reference                 | 199 |
| 5.78.1    | Field Documentation                      | 200 |
| 5.78.1.1  | pParam                                   | 200 |
| 5.78.1.2  | u16RfMode                                | 200 |
| 5.78.1.3  | u16RxCnt                                 | 200 |
| 5.78.1.4  | u16RxCrcOkCnt                            | 200 |
| 5.78.1.5  | u32Freq                                  | 201 |
| 5.78.1.6  | u32Mode                                  | 201 |
| 5.78.1.7  | u32RfChannel                             | 201 |
| 5.78.1.8  | u32Type                                  | 201 |
| 5.78.1.9  | u8Freq                                   | 201 |
| 5.78.1.10 | u8IpcEnable                              | 201 |
| 5.78.1.11 | u8Len                                    | 201 |
| 5.78.1.12 | u8Pkt                                    | 201 |
| 5.78.1.13 | u8Reserved                               | 202 |
| 5.78.1.14 | u8Status                                 | 202 |
| 5.78.1.15 | u8Unicast                                | 202 |
| 5.79      | wifi_active_scan_time_t Struct Reference | 202 |
| 5.79.1    | Detailed Description                     | 202 |
| 5.79.2    | Field Documentation                      | 202 |
| 5.79.2.1  | max                                      | 202 |

|           |   |     |
|-----------|---|-----|
| 5.79.2.2  | min                                       | 203 |
| 5.80      | wifi_ap_config_t Struct Reference         | 203 |
| 5.80.1    | Detailed Description                      | 203 |
| 5.80.2    | Field Documentation                       | 203 |
| 5.80.2.1  | auth_mode                                 | 203 |
| 5.80.2.2  | beacon_interval                           | 203 |
| 5.80.2.3  | channel                                   | 204 |
| 5.80.2.4  | encrypt_type                              | 204 |
| 5.80.2.5  | max_connection                            | 204 |
| 5.80.2.6  | password                                  | 204 |
| 5.80.2.7  | password_length                           | 204 |
| 5.80.2.8  | ssid                                      | 204 |
| 5.80.2.9  | ssid_hidden                               | 204 |
| 5.80.2.10 | ssid_length                               | 204 |
| 5.81      | wifi_auto_connect_info_f Struct Reference | 205 |
| 5.81.1    | Detailed Description                      | 205 |
| 5.81.2    | Field Documentation                       | 205 |
| 5.81.2.1  | ap_channel                                | 205 |
| 5.81.2.2  | beacon_interval                           | 205 |
| 5.81.2.3  | bssid                                     | 206 |
| 5.81.2.4  | capabilities                              | 206 |
| 5.81.2.5  | dtim_prod                                 | 206 |
| 5.81.2.6  | fast_connect                              | 206 |
| 5.81.2.7  | free_ocpy                                 | 206 |
| 5.81.2.8  | hid_ssid                                  | 206 |
| 5.81.2.9  | latest_beacon_rx_time                     | 206 |
| 5.81.2.10 | passphrase                                | 206 |
| 5.81.2.11 | psk                                       | 207 |
| 5.81.2.12 | rsn_ie                                    | 207 |
| 5.81.2.13 | rsni                                      | 207 |

|           |  |     |
|-----------|--|-----|
| 5.81.2.14 | ssid   | 207 |
| 5.81.2.15 | supported_rates                                | 207 |
| 5.81.2.16 | wpa_data                                       | 207 |
| 5.81.2.17 | wpa_ie   | 207 |
| 5.82      | wifi_config_t Union Reference                  | 207 |
| 5.82.1    | Detailed Description                           | 208 |
| 5.82.2    | Field Documentation                            | 208 |
| 5.82.2.1  | ap_config                                      | 208 |
| 5.82.2.2  | sta_config                                     | 208 |
| 5.83      | wifi_event_info_t Union Reference              | 208 |
| 5.83.1    | Detailed Description                           | 208 |
| 5.83.2    | Field Documentation                            | 209 |
| 5.83.2.1  | connected                                      | 209 |
| 5.83.2.2  | disconnected                                   | 209 |
| 5.83.2.3  | got_ip   | 209 |
| 5.83.2.4  | scan_done                                      | 209 |
| 5.84      | wifi_event_sta_connected_t Struct Reference    | 209 |
| 5.84.1    | Detailed Description                           | 209 |
| 5.84.2    | Field Documentation                            | 210 |
| 5.84.2.1  | authmode                                       | 210 |
| 5.84.2.2  | bssid  | 210 |
| 5.84.2.3  | channel  | 210 |
| 5.84.2.4  | ssid   | 210 |
| 5.84.2.5  | ssid_len                                       | 210 |
| 5.85      | wifi_event_sta_disconnected_t Struct Reference | 210 |
| 5.85.1    | Detailed Description                           | 211 |
| 5.85.2    | Field Documentation                            | 211 |
| 5.85.2.1  | bssid  | 211 |
| 5.85.2.2  | reason   | 211 |
| 5.85.2.3  | ssid   | 211 |

|          |   |     |
|----------|---|-----|
| 5.85.2.4 | ssid_len                                    | 211 |
| 5.86     | wifi_event_sta_got_ip_t Struct Reference    | 211 |
| 5.86.1   | Detailed Description                        | 212 |
| 5.86.2   | Field Documentation                         | 212 |
| 5.86.2.1 | ip_changed                                  | 212 |
| 5.87     | wifi_event_sta_scan_done_t Struct Reference | 212 |
| 5.87.1   | Detailed Description                        | 212 |
| 5.87.2   | Field Documentation                         | 212 |
| 5.87.2.1 | number                                      | 212 |
| 5.87.2.2 | scan_id                                     | 212 |
| 5.87.2.3 | status                                      | 213 |
| 5.88     | wifi_fast_scan_threshold_t Struct Reference | 213 |
| 5.88.1   | Detailed Description                        | 213 |
| 5.88.2   | Field Documentation                         | 213 |
| 5.88.2.1 | authmode                                    | 213 |
| 5.88.2.2 | rsi   | 213 |
| 5.89     | wifi_init_config_t Struct Reference         | 213 |
| 5.89.1   | Detailed Description                        | 214 |
| 5.89.2   | Field Documentation                         | 214 |
| 5.89.2.1 | event_handler                               | 214 |
| 5.89.2.2 | magic                                       | 214 |
| 5.90     | wifi_scan_config_t Struct Reference         | 214 |
| 5.90.1   | Detailed Description                        | 214 |
| 5.90.2   | Field Documentation                         | 215 |
| 5.90.2.1 | bssid                                       | 215 |
| 5.90.2.2 | channel                                     | 215 |
| 5.90.2.3 | scan_time                                   | 215 |
| 5.90.2.4 | scan_type                                   | 215 |
| 5.90.2.5 | show_hidden                                 | 215 |
| 5.90.2.6 | ssid  | 215 |

|           |                                    |     |
|-----------|------------------------------------|-----|
| 5.91      | wifi_scan_info_t Struct Reference  | 215 |
| 5.91.1    | Detailed Description               | 216 |
| 5.91.2    | Field Documentation                | 216 |
| 5.91.2.1  | auth_mode                          | 216 |
| 5.91.2.2  | beacon_interval                    | 216 |
| 5.91.2.3  | bssid                              | 216 |
| 5.91.2.4  | capability_info                    | 216 |
| 5.91.2.5  | channel                            | 217 |
| 5.91.2.6  | group_cipher                       | 217 |
| 5.91.2.7  | pairwise_cipher                    | 217 |
| 5.91.2.8  | rsi                                | 217 |
| 5.91.2.9  | ssid                               | 217 |
| 5.91.2.10 | ssid_length                        | 217 |
| 5.92      | wifi_scan_list_t Struct Reference  | 217 |
| 5.92.1    | Detailed Description               | 218 |
| 5.92.2    | Field Documentation                | 218 |
| 5.92.2.1  | ap_record                          | 218 |
| 5.92.2.2  | num                                | 218 |
| 5.93      | wifi_scan_time_t Union Reference   | 218 |
| 5.93.1    | Detailed Description               | 218 |
| 5.93.2    | Field Documentation                | 218 |
| 5.93.2.1  | active                             | 218 |
| 5.93.2.2  | passive                            | 219 |
| 5.94      | wifi_sta_config_t Struct Reference | 219 |
| 5.94.1    | Detailed Description               | 219 |
| 5.94.2    | Field Documentation                | 219 |
| 5.94.2.1  | bssid                              | 219 |
| 5.94.2.2  | bssid_present                      | 219 |
| 5.94.2.3  | password                           | 220 |
| 5.94.2.4  | password_length                    | 220 |
| 5.94.2.5  | scan_method                        | 220 |
| 5.94.2.6  | sort_method                        | 220 |
| 5.94.2.7  | ssid                               | 220 |
| 5.94.2.8  | ssid_length                        | 220 |
| 5.94.2.9  | threshold                          | 220 |



# 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:

|                            |     |
|----------------------------|-----|
| BLE ALL APIs . . . . .     | 9   |
| BLE CM APIs . . . . .      | 10  |
| BLE GAP APIs . . . . .     | 16  |
| BLE GATT APIs . . . . .    | 37  |
| BLE MSG APIs . . . . .     | 71  |
| BLE SMP APIs . . . . .     | 83  |
| WIFI APIs . . . . .        | 91  |
| WIFI Common APIs . . . . . | 96  |
| WIFI STA APIs . . . . .    | 100 |
| Enumeration . . . . .      | 121 |



## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

|  |     |
|--|-----|
| <a href="#">auto_conn_info_t</a>                         | 127 |
| <a href="#">auto_connect_cfg_t</a>                       | 130 |
| <a href="#">event_msg_t</a>                              |     |
| Send information to event by <a href="#">event_msg_t</a> | 131 |
| <a href="#">LE_BT_ADDR_T</a>                             | 132 |
| <a href="#">LE_CM_CONNECTION_COMPLETE_IND_T</a>          | 133 |
| <a href="#">LE_CM_MSG_ADVERTISE_REPORT_IND_T</a>         | 134 |
| <a href="#">LE_CM_MSG_CONN_PARA_REQ_T</a>                | 135 |
| <a href="#">LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T</a>     | 136 |
| <a href="#">LE_CM_MSG_DATA_LEN_CHANGE_IND_T</a>          | 137 |
| <a href="#">LE_CM_MSG_DIRECT_ADV_REPORT_IND_T</a>        | 138 |
| <a href="#">LE_CM_MSG_DISCONNECT_COMPLETE_IND_T</a>      | 139 |
| <a href="#">LE_CM_MSG_ENCRYPTION_CHANGE_IND_T</a>        | 140 |
| <a href="#">LE_CM_MSG_ENCRYPTION_REFRESH_IND_T</a>       | 141 |
| <a href="#">LE_CM_MSG_INIT_COMPLETE_CFM_T</a>            | 142 |
| <a href="#">LE_CM_MSG_LTK_REQ_IND_T</a>                  | 142 |
| <a href="#">LE_CM_MSG_READ_ADV_TX_POWER_CFM_T</a>        | 143 |
| <a href="#">LE_CM_MSG_READ_BD_ADDR_CFM_T</a>             | 144 |
| <a href="#">LE_CM_MSG_READ_CHANNEL_MAP_CFM_T</a>         | 145 |
| <a href="#">LE_CM_MSG_READ_RESOLVING_LIST_SIZE_CFM_T</a> | 145 |
| <a href="#">LE_CM_MSG_READ_RSSI_CFM_T</a>                | 146 |
| <a href="#">LE_CM_MSG_READ_TX_POWER_CFM_T</a>            | 147 |
| <a href="#">LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM_T</a>     | 147 |
| <a href="#">LE_CM_MSG_SET_DATA_LENGTH_CFM_T</a>          | 148 |
| <a href="#">LE_CM_MSG_SET_DISCONNECT_CFM_T</a>           | 148 |
| <a href="#">LE_CM_MSG_SIGNAL_UPDATE_REQ_T</a>            | 149 |
| <a href="#">LE_CM_REQ_STATUS_T</a>                       | 150 |
| <a href="#">LE_CONN_PARA_T</a>                           | 151 |
| <a href="#">LE_GAP_ADVERTISING_PARAM_T</a>               | 152 |
| <a href="#">LE_GAP_CONN_PARAM_T</a>                      | 153 |
| <a href="#">LE_GAP_SCAN_PARAM_T</a>                      | 154 |
| <a href="#">LE_GATT_ATTR_T</a>                           | 155 |
| <a href="#">LE_GATT_MSG_ACCESS_READ_IND_T</a>            | 156 |
| <a href="#">LE_GATT_MSG_ACCESS_WRITE_IND_T</a>           | 157 |
| <a href="#">LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND_T</a>   | 158 |

|   |     |
|---|-----|
| LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND_T                                    | 159 |
| LE_GATT_MSG_CHARACTERISTIC_VAL_IND_T  | 161 |
| LE_GATT_MSG_CONFIRMATION_CFM_T  | 162 |
| LE_GATT_MSG_EXCHANGE_MTU_CFM_T  | 163 |
| LE_GATT_MSG_EXCHANGE_MTU_IND_T  | 164 |
| LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM_T                                      | 164 |
| LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM_T  | 165 |
| LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_CFM_T                                    | 166 |
| LE_GATT_MSG_FIND_CHARACTERISTIC_CFM_T   | 167 |
| LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM_T                                       | 168 |
| LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM_T                                | 169 |
| LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T  | 170 |
| LE_GATT_MSG_INDICATE_IND_T  | 172 |
| LE_GATT_MSG_NOTIFY_CFM_T  | 173 |
| LE_GATT_MSG_NOTIFY_IND_T  | 174 |
| LE_GATT_MSG_OPERATION_TIMEOUT_T   | 175 |
| LE_GATT_MSG_PREPARE_WRITE_RELIABLE_CFM_T                                      | 175 |
| LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_CFM_T                                       | 176 |
| LE_GATT_MSG_READ_CHARACTERISTIC_VALUE_CFM_T                                   | 177 |
| LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM_T  | 178 |
| LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL_CFM_T                                      | 179 |
| LE_GATT_MSG_SERVICE_INFO_IND_T  | 181 |
| LE_GATT_MSG_SIGNED_WRITE_CFM_T  | 182 |
| LE_GATT_MSG_WRITE_CHAR_VAL_RELIABLE_CFM_T                                     | 183 |
| LE_GATT_MSG_WRITE_CHAR_VALUE_CFM_T  | 184 |
| LE_GATT_MSG_WRITE_LONG_CHAR_VALUE_CFM_T                                       | 185 |
| LE_GATT_MSG_WRITE_NO_RSP_CFM_T  | 186 |
| LE_GATT_SERVICE_T   | 187 |
| LE_SMP_MSG_ENCRYPTION_CHANGE_IND_T  | 188 |
| LE_SMP_MSG_ENCRYPTION_REFRESH_IND_T   | 188 |
| LE_SMP_MSG_OOB_DATA_REQUEST_IND_T   | 189 |
| LE_SMP_MSG_PAIRING_ACTION_IND_T   | 189 |
| LE_SMP_MSG_PAIRING_COMPLETE_IND_T   | 190 |
| LE_SMP_MSG_PASSKEY_DISPLAY_IND_T  | 191 |
| LE_SMP_MSG_PASSKEY_INPUT_IND_T  | 192 |
| LE_SMP_MSG_SC_OOB_DATA_REQUEST_IND_T  | 192 |
| LE_SMP_MSG_SLAVE_SECURITY_REQUEST_IND_T                                       | 193 |
| LE_SMP_MSG_USER_CONFIRM_IND_T   | 194 |
| LE_SMP_SC_OOB_DATA_T  | 194 |
| LE_SYS_MSG_BUF_OVERFLOW_T   | 195 |
| mw_wifi_auto_connect_ap_info_t  | 195 |
| MwFimAutoConnectCFG_t   | 198 |
| T_RfCmd   | 199 |
| T_RfEvt   | 199 |
| wifi_active_scan_time_t   |     |
| Range of active scan times per channel  | 202 |
| wifi_ap_config_t  |     |
| This structure is the Wi-Fi configuration for initialization for Soft-AP mode | 203 |
| wifi_auto_connect_info_f  |     |
| WiFi auto connect info parameters   | 205 |
| wifi_config_t   |     |
| Wi-Fi configuration for initialization  | 207 |
| wifi_event_info_t   |     |
| Wifi_event_info_t   | 208 |
| wifi_event_sta_connected_t  |     |
| Wifi_event_sta_connected_t  | 209 |
| wifi_event_sta_disconnected_t   |     |
| Wifi_event_sta_disconnected_t   | 210 |

|   |     |
|---|-----|
| <a href="#">wifi_event_sta_got_ip_t</a>   |     |
| Wifi_event_sta_got_ip_t . . . . .   | 211 |
| <a href="#">wifi_event_sta_scan_done_t</a>  |     |
| Wifi_event_sta_scan_done_t . . . . .  | 212 |
| <a href="#">wifi_fast_scan_threshold_t</a>  |     |
| Structure describing parameters for a Wi-Fi fast scan . . . . .                               | 213 |
| <a href="#">wifi_init_config_t</a>  |     |
| WiFi stack configuration parameters . . . . .   | 213 |
| <a href="#">wifi_scan_config_t</a>  |     |
| Parameters for an SSID scan . . . . .   | 214 |
| <a href="#">wifi_scan_info_t</a>  |     |
| This structure defines the inforamtion of scanned APs . . . . .                               | 215 |
| <a href="#">wifi_scan_list_t</a>  |     |
| This structure defines the list of scanned APs with their corresponding information . . . . . | 217 |
| <a href="#">wifi_scan_time_t</a>  |     |
| Aggregate of active & passive scan time per channel . . . . .                                 | 218 |
| <a href="#">wifi_sta_config_t</a>   |     |
| This structure is the Wi-Fi configuration for initialization for STA mode . . . . .           | 219 |



## 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

|            |                        |
|------------|------------------------|
| <i>the</i> | 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_RANDOM 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_RANDOM_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

|                |                             |
|----------------|-----------------------------|
| <i>bt_addr</i> | BT device address.          |
| <i>irk</i>     | 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

|                |                    |
|----------------|--------------------|
| <i>bt_addr</i> | 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

|              |                                   |
|--------------|-----------------------------------|
| <i>start</i> | 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

|                      |                             |
|----------------------|-----------------------------|
| <i>taddr</i>         | advertisers device address. |
| <i>own_addr_type</i> | 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

|           |               |
|-----------|---------------|
| <i>ch</i> | 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**

|                 |                               |
|-----------------|-------------------------------|
| <i>conn_hdl</i> | connection handle.            |
| <i>accept</i>   | 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**

|                 |                               |
|-----------------|-------------------------------|
| <i>conn_hdl</i> | connection handle.            |
| <i>para</i>     | update connection parameters. |

**Returns**

None.

**4.3.3.11 LeGapConnUpdateResponse()**

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

Connection parameters update response.

**Parameters**

|                   |                         |
|-------------------|-------------------------|
| <i>conn_hdl</i>   | connection handle.      |
| <i>identifier</i> | TBD                     |
| <i>accept</i>     | accept request, or not. |

**Returns**

None.

#### 4.3.3.12 LeGapDisconnectReq()

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

Disconnect the physical connection.

##### Parameters

|                 |                    |
|-----------------|--------------------|
| <i>conn_hdl</i> | 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

|             |               |
|-------------|---------------|
| <i>type</i> | address type. |
| <i>addr</i> | 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

|                 |                    |
|-----------------|--------------------|
| <i>conn_hdl</i> | 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

|                 |                    |
|-----------------|--------------------|
| <i>conn_hdl</i> | 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

|                 |   |
|-----------------|---|
| <i>conn_hdl</i> | connection handle.                                    |
| <i>type</i>     | current tx power, or maximum 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**

|                |                    |
|----------------|--------------------|
| <i>bt_addr</i> | 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**

|               |   |
|---------------|---|
| <i>start</i>  | TRUE is start, FALSE is not.  |
| <i>filter</i> | scan policy, refer to <code>LE_HCI_SCAN_FILT_*</code> in <a href="#">ble_hci_if.h</a> |

**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**

|             |                  |
|-------------|------------------|
| <i>len</i>  | ADV data length. |
| <i>data</i> | 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

|               |                     |
|---------------|---------------------|
| <i>params</i> | 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

|                            |                              |
|----------------------------|------------------------------|
| <i>interval_min</i>        | mininum connection interval. |
| <i>interval_max</i>        | maxinum connection interval. |
| <i>slave_latency</i>       | slave letency.               |
| <i>supervision_timeout</i> | 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

|                  |  |
|------------------|--|
| <i>tx_octets</i> | the maximum number of octets in the Payload field that the local device will send to the remote device.    |
| <i>tx_time</i>   | 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

|             |   |
|-------------|---|
| <i>addr</i> | 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

|                |                                   |
|----------------|-----------------------------------|
| <i>timeout</i> | 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**

|             |   |
|-------------|---|
| <i>addr</i> | 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**

|               |                  |
|---------------|------------------|
| <i>params</i> | 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**

|             |                            |
|-------------|----------------------------|
| <i>len</i>  | scan response data length. |
| <i>data</i> | 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 *)&gcCharacteristicUuid, LE_GATT_PERMIT_READ, 0, 5, (UINT8 *)&gcCharacteristicUuid, LE_GATT_PERMIT_READ, 0, 5, (UINT8 *)&gcCharacteristicUuid}`
- `#define CHARACTERISTIC_DECL_UUID16(pVal) {0, LE_GATT_UUID16, (UINT16 *)&gcCharacteristicUuid, LE_GATT_PERMIT_READ, 0, 5, (UINT8 *)&gcCharacteristicUuid, LE_GATT_PERMIT_READ, 0, 5, (UINT8 *)&gcCharacteristicUuid}`
- `#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_UUID16(pVal) {0, LE_GATT_UUID16, (UINT16 *)&pVal, 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, UI↔NT16 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 CHARAggregateDescriptor

```
#define CHARAggregateDescriptor(  
    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 *) (pVal)}
```

#### 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\_UUIN16

```
#define INCLUDE_DECL_UUIN16(  
    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 information indication                    |
| LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_↔<br>_CFM     | find all primary service confirm                  |
| LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_↔<br>_UUID_CFM | find primary service by UUID confirm              |
| LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM              | find include service confirm                      |
| LE_GATT_MSG_CHARACTERISTIC_DECL_INF_↔<br>O_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_VAL_↔<br>UE_CFM    | read characteristic value, confirm message        |
| LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_↔<br>CFM         | read characteristic value by UUID confirm message |
| LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM                 | read long characteristic value confirm message    |
| LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL_↔<br>CFM        | read multiple characteristic value confirm        |
| LE_GATT_MSG_WRITE_CHAR_VALUE_CFM                   | write characteristic value confirm                |
| LE_GATT_MSG_WRITE_LONG_CHAR_VALUE_↔<br>CFM         | write long characteristic value confirm           |
| LE_GATT_MSG_WRITE_CHAR_VAL_RELIABLE_↔<br>_CFM      | write characteristic value reliable confirm       |
| LE_GATT_MSG_PREPARE_WRITE_RELIABLE_↔<br>CFM        | prepare write reliable confirm                    |
| LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_↔<br>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 information |
| 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

|                 |   |
|-----------------|---|
| <i>conn_hdl</i> | connection handle.  |
| <i>handle</i>   | attribute handle.   |
| <i>att_err</i>  | 0 is OK, others refer to LE_ATT_ERR_* in <a href="#">ble_att_if.h</a> . |

## 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

|                 |   |
|-----------------|---|
| <i>conn_hdl</i> | connection handle.  |
| <i>method</i>   | refer to LE_GATT_FLAG_* in <a href="#">ble_gatt_if.h</a>                |
| <i>handle</i>   | attribute handle.   |
| <i>att_err</i>  | 0 is OK, others refer to LE_ATT_ERR_* in <a href="#">ble_att_if.h</a> . |

## 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

|    |               |                             |
|----|---------------|-----------------------------|
|    | <i>svc</i>    | service.                    |
|    | <i>attrId</i> | attribute index of service. |
| in | <i>len</i>    | attribute value length.     |
| in | <i>val</i>    | 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

|                 |                    |
|-----------------|--------------------|
| <i>conn_hdl</i> | 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**

|                 |                              |
|-----------------|------------------------------|
| <i>conn_hdl</i> | connection handle.           |
| <i>hdl</i>      | characteristic value handle. |
| <i>len</i>      | value length.                |
| <i>pval</i>     | 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**

|                 |                              |
|-----------------|------------------------------|
| <i>conn_hdl</i> | connection handle.           |
| <i>hdl</i>      | characteristic value handle. |
| <i>len</i>      | value length.                |
| <i>pval</i>     | 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

|                 |                    |
|-----------------|--------------------|
| <i>conn_hdl</i> | connection handle. |
| <i>mtu</i>      | 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

|                 |                    |
|-----------------|--------------------|
| <i>conn_hdl</i> | connection handle. |
| <i>mtu</i>      | 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

|                 |                       |
|-----------------|-----------------------|
| <i>conn_hdl</i> | connection handle.    |
| <i>yesno</i>    | 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

|                  |                    |
|------------------|--------------------|
| <i>conn_hdl</i>  | connection handle. |
| <i>start_hdl</i> | start handle.      |
| <i>end_hdl</i>   | 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

|                  |                    |
|------------------|--------------------|
| <i>conn_hdl</i>  | connection handle. |
| <i>start_hdl</i> | start handle.      |
| <i>end_hdl</i>   | 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

|                 |                    |
|-----------------|--------------------|
| <i>conn_hdl</i> | 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

|                  |                    |
|------------------|--------------------|
| <i>conn_hdl</i>  | connection handle. |
| <i>start_hdl</i> | start handle.      |
| <i>end_hdl</i>   | end handle.        |
| <i>format</i>    | UUID type.         |
| <i>uuid</i>      | 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**

|                  |                    |
|------------------|--------------------|
| <i>conn_hdl</i>  | connection handle. |
| <i>start_hdl</i> | start handle.      |
| <i>end_hdl</i>   | 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**

|                 |                    |
|-----------------|--------------------|
| <i>conn_hdl</i> | connection handle. |
| <i>format</i>   | UUID type.         |
| <i>uuid</i>     | 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

|               |                             |
|---------------|-----------------------------|
| <i>svc</i>    | service.                    |
| <i>attrId</i> | 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

|     |               |                             |
|-----|---------------|-----------------------------|
|     | <i>svc</i>    | service.                    |
|     | <i>attrId</i> | attribute index of service. |
| out | <i>len</i>    | attribute value length.     |
| out | <i>val</i>    | 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

|                     |                             |
|---------------------|-----------------------------|
| <i>svc</i>          | service.                    |
| <i>attr↔<br/>Id</i> | 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

|                     |                             |
|---------------------|-----------------------------|
| <i>svc</i>          | service.                    |
| <i>attr↔<br/>Id</i> | 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

|                |                            |
|----------------|----------------------------|
| <i>appTask</i> | 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

|               |                             |
|---------------|-----------------------------|
| <i>svc</i>    | servie.                     |
| <i>attrId</i> | attribute index of service. |
| <i>offset</i> | modify offset.              |
| <i>len</i>    | modify length.              |
| <i>val</i>    | 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

|                 |                              |
|-----------------|------------------------------|
| <i>conn_hdl</i> | connection handle.           |
| <i>handle</i>   | characteristic value handle. |
| <i>offset</i>   | offset written.              |
| <i>len</i>      | length written.              |
| <i>val</i>      | 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

|                  |                    |
|------------------|--------------------|
| <i>conn_hdl</i>  | connection handle. |
| <i>start_hdl</i> | start handle.      |
| <i>end_hdl</i>   | end handle.        |
| <i>format</i>    | UUID type.         |
| <i>uuid</i>      | 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

|                 |                              |
|-----------------|------------------------------|
| <i>conn_hdl</i> | connection handle.           |
| <i>handle</i>   | 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

|                 |                              |
|-----------------|------------------------------|
| <i>conn_hdl</i> | connection handle.           |
| <i>handle</i>   | characteristic value handle. |
| <i>offset</i>   | 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

|                 |                    |
|-----------------|--------------------|
| <i>conn_hdl</i> | connection handle. |
| <i>count</i>    | handle count.      |
| <i>handle</i>   | 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

|                  |                         |
|------------------|-------------------------|
| <i>inc_hdl</i>   | include service handle. |
| <i>start_hdl</i> | start handle.           |
| <i>end_hdl</i>   | end handle.             |
| <i>uuid</i>      | 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

|                  |                                  |
|------------------|----------------------------------|
| <i>attrTable</i> | service attribute table.         |
| <i>numAttr</i>   | 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**

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

|                 |                    |
|-----------------|--------------------|
| <i>conn_hdl</i> | 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**

|                 |                                   |
|-----------------|-----------------------------------|
| <i>conn_hdl</i> | connection handle.                |
| <i>handle</i>   | characteristic value handle.      |
| <i>len</i>      | length of the data to be written. |
| <i>val</i>      | 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

|                 |                              |
|-----------------|------------------------------|
| <i>conn_hdl</i> | connection handle.           |
| <i>handle</i>   | characteristic value handle. |
| <i>offset</i>   | offset written.              |
| <i>len</i>      | length written.              |
| <i>val</i>      | 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

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

|                 |                                   |
|-----------------|-----------------------------------|
| <i>conn_hdl</i> | connection handle.                |
| <i>handle</i>   | characteristic value handle.      |
| <i>len</i>      | length of the data to be written. |
| <i>val</i>      | 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, UINT32 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

|             |             |
|-------------|-------------|
| <i>task</i> | task.       |
| <i>id</i>   | 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**

|                    |                              |
|--------------------|------------------------------|
| <i>task</i>        | the task of recvice message. |
| <i>id</i>          | message id.                  |
| <i>sub↔<br/>id</i> | 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**

|             |             |
|-------------|-------------|
| <i>task</i> | task.       |
| <i>id</i>   | 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

|              |                              |
|--------------|------------------------------|
| <i>task</i>  | the task of recvice message. |
| <i>id</i>    | message id.                  |
| <i>subId</i> | sub message id.              |

##### Returns

True is ok, false is error.

#### 4.5.5.5 LeGetSubMsgId()

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

Get sub message id.

##### Parameters

|            |             |
|------------|-------------|
| <i>sub</i> | message id. |
|------------|-------------|

##### Returns

0 is ok, others is error.

#### 4.5.5.6 LeHostCreateTask()

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

Create BLE task.

**Parameters**

|             |                              |
|-------------|------------------------------|
| <i>task</i> | the reference of BLE task.   |
| <i>hdl</i>  | 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**

|              |                        |
|--------------|------------------------|
| <i>task</i>  | reference of BLE task. |
| <i>msgId</i> | message ID.            |
| <i>msg</i>   | 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

|                           |                        |
|---------------------------|------------------------|
| <i>task</i>               | reference of BLE task. |
| <i>msg</i> ↔<br><i>Id</i> | message ID.            |
| <i>msg</i>                | message.               |
| <i>delay</i>              | 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

|             |                              |
|-------------|------------------------------|
| <i>task</i> | the task of recvice message. |
| <i>id</i>   | message id.                  |
| <i>msg</i>  | message.                     |
| <i>lock</i> | lock number.                 |

#### Returns

None.

#### 4.5.5.11 LeSendSubMessage()

```

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

```

Send sub message.

## Parameters

|                    |                              |
|--------------------|------------------------------|
| <i>task</i>        | the task of recvice message. |
| <i>msg↔<br/>Id</i> | message id.                  |
| <i>subId</i>       | sub message id.              |
| <i>msg</i>         | 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

|                    |                              |
|--------------------|------------------------------|
| <i>task</i>        | the task of recvice message. |
| <i>msg↔<br/>Id</i> | message id.                  |
| <i>subId</i>       | sub message id.              |
| <i>msg</i>         | message.                     |
| <i>delay</i>       | 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**

|                    |                              |
|--------------------|------------------------------|
| <i>task</i>        | the task of recvice message. |
| <i>id</i>          | message id.                  |
| <i>sub↔<br/>id</i> | sub message id.              |
| <i>msg</i>         | message.                     |
| <i>lock</i>        | 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 data 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

|                |                            |
|----------------|----------------------------|
| <i>appTask</i> | 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

|                 |                    |
|-----------------|--------------------|
| <i>conn_hdl</i> | connection handle. |
| <i>data</i>     | response data.     |
| <i>len</i>      | data length.       |

#### Returns

None.

### 4.6.4.3 LeSmpOobPresent()

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

SMP OOB present.

**Parameters**

|                    |                    |
|--------------------|--------------------|
| <i>conn_hdl</i>    | connection handle. |
| <i>oob_present</i> | 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**

|                 |                    |
|-----------------|--------------------|
| <i>conn_hdl</i> | connection handle. |
| <i>passkey</i>  | passkey.           |

**Returns**

None.

**4.6.4.5 LeSmpScOobComputeConfirmVal()**

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

SMP secure connection OOB compute confirm value.

**Parameters**

|                |               |
|----------------|---------------|
| <i>rand</i>    | random data.  |
| <i>confirm</i> | 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

|                 |                    |
|-----------------|--------------------|
| <i>conn_hdl</i> | connection handld. |
| <i>our_rand</i> | our random data.   |
| <i>peer</i>     | peer OOB data.     |

##### Returns

None.

#### 4.6.4.7 LeSmpSecurityReq()

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

BLE SMP security request.

##### Parameters

|                 |                    |
|-----------------|--------------------|
| <i>conn_hdl</i> | 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

|                 |                               |
|-----------------|-------------------------------|
| <i>conn_hdl</i> | connection handle.            |
| <i>accept</i>   | 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**

|              |  |
|--------------|--|
| <i>iocap</i> | IO capability.   |
| <i>mitm</i>  | TRUE is MITM protected, FALSE is not.                        |
| <i>sc</i>    | TRUE is request BLE secure connection pairing, FALSE is not. |
| <i>bond</i>  | 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**

|                 |                    |
|-----------------|--------------------|
| <i>conn_hdl</i> | connection handle. |
| <i>accept</i>   | 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 information 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 | <i>event</i>   | event type Set the event type,Options are <ul style="list-style-type: none"> <li>• WIFI_EVENT_INIT_COMPLETE</li> <li>• WIFI_EVENT_SCAN_COMPLETE</li> <li>• WIFI_EVENT_STA_START</li> <li>• WIFI_EVENT_STA_STOP</li> <li>• WIFI_EVENT_STA_CONNECTED</li> <li>• WIFI_EVENT_STA_DISCONNECTED</li> <li>• WIFI_EVENT_STA_CONNECTION_FAILED</li> <li>• WIFI_EVENT_STA_GOT_IP</li> </ul> |
| in | <i>payload</i> | Data block that transmitted to event  |
| in | <i>length</i>  | 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 | <i>idx</i>     | one of the enums of<br>bt_scan_mode_t |
| in | <i>handler</i> | 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**

|                 |  |
|-----------------|--|
| <code>cb</code> | : 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 TCP/IP module, can call this API to send an event to event task

**Parameters**

|                          |   |
|--------------------------|---|
| <code>event_msg_t</code> | * 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

|                        |                          |
|------------------------|--------------------------|
| <i>wifi_event_cb_t</i> | cb : callback            |
| <i>void</i>            | *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 | <i>event</i>   | event type Set the event type,Options are <ul style="list-style-type: none"> <li>• WIFI_EVENT_INIT_COMPLETE</li> <li>• WIFI_EVENT_SCAN_COMPLETE</li> <li>• WIFI_EVENT_STA_START</li> <li>• WIFI_EVENT_STA_STOP</li> <li>• WIFI_EVENT_STA_CONNECTED</li> <li>• WIFI_EVENT_STA_DISCONNECTED</li> <li>• WIFI_EVENT_STA_CONNECTION_FAILED</li> <li>• WIFI_EVENT_STA_GOT_IP</li> </ul> |
| in | <i>payload</i> | Data block transmitted to event   |
| in | <i>length</i>  | 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 | <i>event</i>   | is an optional event to register. For more details, please refer to <a href="#">wifi_event_t</a> .  |
| in | <i>payload</i> | 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 | <i>length</i>  | 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 <a href="#">wifi_init()</a> . 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 | <i>index</i> | : 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 | <i>index</i> | : Index of ap information, The range is 0 to 3                         |
| in | <i>info</i>  | : <a href="#">wifi_auto_connect_info_f</a> 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 | <i>mode</i> | 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  | <i>interface</i> | Configure the current wifi working mode,The options are <ul style="list-style-type: none"> <li>• WIFI_MODE_STA</li> <li>• WIFI_MODE_AP (currently not support)</li> </ul> |
| out | <i>bandwidth</i> | 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 | <i>bssid</i> | 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  | <i>interface</i> | Configure the current wifi working mode,The options are <ul style="list-style-type: none"><li>• WIFI_MODE_STA</li><li>• WIFI_MODE_AP (currently not support)</li></ul> |
| out | <i>channel</i>   | 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  | <i>interface</i> | Configure the current wifi working mode, The options are <ul style="list-style-type: none"><li>WIFI_MODE_STA</li><li>WIFI_MODE_AP (currently not support)</li></ul> |
| out | <i>address</i>   | 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 | <i>ssid</i>        | Get ssid by pointer                  |
| out | <i>ssid_length</i> | 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 | <i>interface</i> | Configure the current wifi working mode, The options are <ul style="list-style-type: none"><li>• WIFI_MODE_STA</li><li>• WIFI_MODE_AP (currently not support)</li></ul> |
| in | <i>bandwidth</i> | 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 | <i>bssid</i> | 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 | <i>interface</i> | Configure the current wifi working mode,The options are <ul style="list-style-type: none"> <li>• WIFI_MODE_STA</li> <li>• WIFI_MODE_AP (currently not support)</li> </ul> |
| in | <i>channel</i>   | 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 | <i>interface</i> | Configure the current wifi working mode,The options are <ul style="list-style-type: none"> <li>• WIFI_MODE_STA</li> <li>• WIFI_MODE_AP (currently not support)</li> </ul> |
| in | <i>address</i>   | 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 | <i>interface</i>   | Configure the current wifi working mode, The options are <ul style="list-style-type: none"><li>WIFI_MODE_STA</li><li>WIFI_MODE_AP (currently not support)</li></ul> |
| in | <i>ssid</i>        | Set the value of ssid   |
| in | <i>ssid_length</i> | 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 | <i>config</i> | 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 | <i>address</i> | 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 | <i>rssi</i> | 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 | <i>event</i>   | The type of the registered event. Options are <ul style="list-style-type: none"> <li>• WIFI_EVENT_INIT_COMPLETE</li> <li>• WIFI_EVENT_SCAN_COMPLETE</li> <li>• WIFI_EVENT_STA_START</li> <li>• WIFI_EVENT_STA_STOP</li> <li>• WIFI_EVENT_STA_CONNECTED</li> <li>• WIFI_EVENT_STA_DISCONNECTED</li> <li>• WIFI_EVENT_STA_CONNECTION_FAILED</li> <li>• WIFI_EVENT_STA_GOT_IP</li> </ul> |
| in | <i>handler</i> | 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 | <i>event</i>   | The type of the unregistered event. Options please refer to <a href="#">wifi_connection_register_event_handler()</a> |
| in | <i>handler</i> | 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 | <i>ap_index</i> | : 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 | <i>mode</i>     | : Configure the fast connect mode ,0 means disable fast connection, and 1 enable the fast connection mode |
| in | <i>ap_index</i> | : 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  | <i>interface</i> | Configure wifi working mode,The options are <ul style="list-style-type: none"><li>• WIFI_MODE_STA</li><li>• WIFI_MODE_AP (currently not support)</li></ul> |
| out | <i>conf</i>      | 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 | <i>config</i>  | pointer to Wi-Fi init configuration structure; can point to a temporary variable.          |
| in | <i>init_cb</i> | 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 | <i>scan_list</i> | 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 | <i>number</i> | 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 | <i>number</i>     | 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 | <i>ap_records</i> | <a href="#">wifi_scan_info_t</a> 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 | <code>config</code> | Configure parameters for scan operation   |
| in | <code>block</code>  | 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, `ssid_set` shall be set to 0; set to 1 means user want to check MAC address of certain AP.
3. OPL1000 is limited to working on one channel.

## Parameters

|    |                        |  |
|----|------------------------|--|
| in | <code>interface</code> | Configure wifi working mode, The options are <ul style="list-style-type: none"> <li>• <code>WIFI_MODE_STA</code></li> <li>• <code>WIFI_MODE_AP</code> (currently not support)</li> </ul> |
| in | <code>conf</code>      | 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 | <i>ap_info</i> | 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_STA_AUTO_CONNECT_FAILED_IND`, `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_UNACCEPTABLE`, `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` }  
*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 <a href="#">wifi_reason_code_t</a> . |
| WIFI_EVENT_STA_GOT_IP                  | station got IP from connected AP   |
| WIFI_EVENT_STA_AUTO_CONNECT_FAILED_IND | station auto connect failed indication   |
| 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_OVERLOAD | 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.                           |

## Enumerator

|  |   |
|--|---|
| WIFI_REASON_CODE_CLASS_3_ERR                   | 7 Class 3 frame received from nonauthenticated STA.   |
| 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_UNACCEPTABLE | 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_TIMEOUT       | 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_INVALID          | 18 Invalid group cipher.  |
| WIFI_REASON_CODE_PAIRWISE_CIPHER_INVALID       | 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_CAPABILITIES     | 22 Invalid RSN information element capabilities.  |
| WIFI_REASON_CODE_IEEE_802_1X_AUTH_FAILED       | 23 IEEE 802.1X authentication failed.   |
| WIFI_REASON_CODE_CIPHER_REJECTED               | 24 Cipher suite rejected because of the security policy.  |

## 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 [rsni](#)
- 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

peer 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

maximum connection interval

#### 5.7.1.3 itv\_min

UINT16 itv\_min

minimum 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

maximum connection interval

#### 5.25.1.4 interval\_min

UINT16 interval\_min

minimum 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

maximum 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`

maximum advertising interval

#### 5.28.1.4 `interval_min`

`UINT16 interval_min`

minimum 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

maximum connection interval

#### 5.29.1.2 interval\_min

UINT16 interval\_min

minimum 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

maximum 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 [rsni](#)
- 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 information 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](#) `rss`
- [uint8\\_t](#) `ssid` [[WIFI\\_MAX\\_LENGTH\\_OF\\_SSID](#)]
- [uint8\\_t](#) `ssid_length`

### 5.91.1 Detailed Description

This structure defines the information 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

- action
  - LE\_SMP\_MSG\_PAIRING\_ACTION\_IND\_T, [189](#)
- active
  - wifi\_scan\_time\_t, [218](#)
- addr
  - LE\_BT\_ADDR\_T, [132](#)
  - LE\_CM\_MSG\_ADVERTISE\_REPORT\_IND\_T, [135](#)
- addr\_type
  - LE\_CM\_MSG\_ADVERTISE\_REPORT\_IND\_T, [135](#)
- ap\_channel
  - auto\_conn\_info\_t, [127](#)
  - mw\_wifi\_auto\_connect\_ap\_info\_t, [196](#)
  - wifi\_auto\_connect\_info\_f, [205](#)
- ap\_config
  - wifi\_config\_t, [208](#)
- ap\_record
  - wifi\_scan\_list\_t, [218](#)
- att\_err
  - LE\_GATT\_MSG\_CHARACTERISTIC\_VAL\_IND\_T, [161](#)
  - LE\_GATT\_MSG\_EXECUTE\_WRITE\_RELIABLE\_CFM\_T, [165](#)
  - LE\_GATT\_MSG\_FIND\_ALL\_CHAR\_DESC\_CFM\_T, [166](#)
  - LE\_GATT\_MSG\_FIND\_ALL\_PRIMARY\_SERVICE\_CFM\_T, [167](#)
  - LE\_GATT\_MSG\_FIND\_CHARACTERISTIC\_CFM\_T, [168](#)
  - LE\_GATT\_MSG\_FIND\_INCLUDED\_SERVICE\_CFM\_T, [169](#)
  - LE\_GATT\_MSG\_FIND\_PRIMARY\_SERVICE\_BY\_UUID\_CFM\_T, [170](#)
  - LE\_GATT\_MSG\_PREPARE\_WRITE\_RELIABLE\_CFM\_T, [176](#)
  - LE\_GATT\_MSG\_READ\_CHAR\_VAL\_BY\_UUID\_CFM\_T, [177](#)
  - LE\_GATT\_MSG\_READ\_CHARACTERISTIC\_VALUE\_CFM\_T, [178](#)
  - LE\_GATT\_MSG\_READ\_LONG\_CHAR\_VAL\_CFM\_T, [179](#)
  - LE\_GATT\_MSG\_READ\_MULTIPLE\_CHAR\_VAL\_CFM\_T, [180](#)
  - LE\_GATT\_MSG\_WRITE\_CHAR\_VAL\_RELIABLE\_CFM\_T, [183](#)
  - LE\_GATT\_MSG\_WRITE\_CHAR\_VALUE\_CFM\_T, [184](#)
  - LE\_GATT\_MSG\_WRITE\_LONG\_CHAR\_VALUE\_CFM\_T, [185](#)
- att\_op
  - LE\_GATT\_MSG\_OPERATION\_TIMEOUT\_T, [175](#)
- auth\_mode
  - wifi\_ap\_config\_t, [203](#)
  - wifi\_scan\_info\_t, [216](#)
- authenticated
  - LE\_SMP\_MSG\_PAIRING\_COMPLETE\_IND\_T, [190](#)
- authmode
  - wifi\_event\_sta\_connected\_t, [210](#)
  - wifi\_fast\_scan\_threshold\_t, [213](#)
- auto\_conn\_info\_t, [127](#)
  - ap\_channel, [127](#)
  - beacon\_interval, [127](#)
  - bssid, [128](#)
  - capabilities, [128](#)
  - dtim\_prod, [128](#)
  - fast\_connect, [128](#)
  - free\_ocpy, [128](#)
  - hid\_ssid, [128](#)
  - latest\_beacon\_rx\_time, [128](#)
  - passphrase, [128](#)
  - psk, [129](#)
  - rsn\_ie, [129](#)
  - rss, [129](#)
  - ssid, [129](#)
  - supported\_rates, [129](#)
  - wpa\_data, [129](#)
  - wpa\_ie, [129](#)
- auto\_connect\_cfg\_t, [130](#)
  - flag, [130](#)
  - front, [130](#)
  - max\_save\_num, [130](#)
  - pFCInfo, [130](#)
  - rear, [130](#)
  - retryCount, [131](#)
  - targetIdx, [131](#)
  - uFCAPNum, [131](#)
- BLE ALL APIs, [9](#)
- BLE CM APIs, [10](#)
  - LE\_CM\_MSG\_ADD\_TO\_RESOLVING\_LIST\_CFM\_T, [11](#)
  - LE\_CM\_MSG\_ADD\_TO\_WHITE\_LIST\_CFM\_T, [11](#)
  - LE\_CM\_MSG\_CANCEL\_CONNECTION\_CFM\_T, [11](#)
  - LE\_CM\_MSG\_CLEAR\_RESOLVING\_LIST\_CFM\_T, [12](#)

- LE\_CM\_MSG\_CLEAR\_WHITE\_LIST\_CFM\_T, [12](#)
- LE\_CM\_MSG\_CREATE\_CONNECTION\_CFM\_T, [12](#)
- LE\_CM\_MSG\_ENTER\_ADVERTISING\_CFM\_T, [12](#)
- LE\_CM\_MSG\_ENTER\_SCANNING\_CFM\_T, [12](#)
- LE\_CM\_MSG\_EXIT\_ADVERTISING\_CFM\_T, [12](#)
- LE\_CM\_MSG\_EXIT\_SCANNING\_CFM\_T, [12](#)
- LE\_CM\_MSG\_REMOVE\_FROM\_RESOLVING\_LIST\_CFM\_T, [12](#)
- LE\_CM\_MSG\_REMOVE\_FROM\_WHITE\_LIST\_CFM\_T, [13](#)
- LE\_CM\_MSG\_SET\_ADVERTISING\_DATA\_CFM\_T, [13](#)
- LE\_CM\_MSG\_SET\_ADVERTISING\_PARAMS\_CFM\_T, [13](#)
- LE\_CM\_MSG\_SET\_CHANNEL\_MAP\_CFM\_T, [13](#)
- LE\_CM\_MSG\_SET\_RANDOM\_ADDRESS\_CFM\_T, [13](#)
- LE\_CM\_MSG\_SET\_RPA\_TIMEOUT\_CFM\_T, [13](#)
- LE\_CM\_MSG\_SET\_SCAN\_PARAMS\_CFM\_T, [13](#)
- LE\_CM\_MSG\_SET\_SCAN\_RSP\_DATA\_CFM\_T, [13](#)
- LeCmlInit, [15](#)
- BLE GAP APIs, [16](#)
  - GAP\_ADTYPE\_128BIT\_COMPLETE, [18](#)
  - GAP\_ADTYPE\_128BIT\_MORE, [18](#)
  - GAP\_ADTYPE\_16BIT\_COMPLETE, [18](#)
  - GAP\_ADTYPE\_16BIT\_MORE, [18](#)
  - GAP\_ADTYPE\_32BIT\_COMPLETE, [19](#)
  - GAP\_ADTYPE\_32BIT\_MORE, [19](#)
  - GAP\_ADTYPE\_3D\_INFO\_DATA, [19](#)
  - GAP\_ADTYPE\_ADV\_INTERVAL, [19](#)
  - GAP\_ADTYPE\_APPEARANCE, [19](#)
  - GAP\_ADTYPE\_FLAGS\_BREDR\_NOT\_SUPPORTED, [19](#)
  - GAP\_ADTYPE\_FLAGS\_GENERAL, [19](#)
  - GAP\_ADTYPE\_FLAGS\_LIMITED, [20](#)
  - GAP\_ADTYPE\_FLAGS, [19](#)
  - GAP\_ADTYPE\_LE\_BD\_ADDR, [20](#)
  - GAP\_ADTYPE\_LE\_ROLE, [20](#)
  - GAP\_ADTYPE\_LOCAL\_NAME\_COMPLETE, [20](#)
  - GAP\_ADTYPE\_LOCAL\_NAME\_SHORT, [20](#)
  - GAP\_ADTYPE\_MANUFACTURER\_SPECIFIC, [20](#)
  - GAP\_ADTYPE\_OOB\_CLASS\_OF\_DEVICE, [20](#)
  - GAP\_ADTYPE\_OOB\_SIMPLE\_PAIRING\_HASHC, [20](#)
  - GAP\_ADTYPE\_OOB\_SIMPLE\_PAIRING\_RANDOM, [21](#)
  - GAP\_ADTYPE\_POWER\_LEVEL, [21](#)
  - GAP\_ADTYPE\_PUBLIC\_TARGET\_ADDR, [21](#)
  - GAP\_ADTYPE\_RANDOM\_TARGET\_ADDR, [21](#)
  - GAP\_ADTYPE\_SERVICE\_DATA\_128BIT, [21](#)
  - GAP\_ADTYPE\_SERVICE\_DATA\_32BIT, [21](#)
  - GAP\_ADTYPE\_SERVICE\_DATA, [21](#)
  - GAP\_ADTYPE\_SERVICES\_LIST\_128BIT, [21](#)
  - GAP\_ADTYPE\_SERVICES\_LIST\_16BIT, [22](#)
  - GAP\_ADTYPE\_SIGNED\_DATA, [22](#)
  - GAP\_ADTYPE\_SIMPLE\_PAIRING\_HASHC, [256](#)
  - GAP\_ADTYPE\_SIMPLE\_PAIRING\_RANDOM, [256](#)
  - GAP\_ADTYPE\_SLAVE\_CONN\_INTERVAL\_RANDOM, [22](#)
  - GAP\_ADTYPE\_SM\_OOB\_FLAG, [22](#)
  - GAP\_ADTYPE\_SM\_TK, [22](#)
  - GAP\_PUBLIC\_ADDR, [22](#)
  - GAP\_RANDOM\_ADDR\_NRPA, [23](#)
  - GAP\_RANDOM\_ADDR\_RPA, [23](#)
  - GAP\_RANDOM\_ADDR\_STATIC, [23](#)
  - GAP\_SCAN\_TYPE\_ACTIVE, [23](#)
  - GAP\_SCAN\_TYPE\_PASSIVE, [23](#)
  - GAP\_TX\_PWR\_CURR\_VAL, [23](#)
  - GAP\_TX\_PWR\_MAX\_VAL, [23](#)
  - GAPBOND\_IO\_CAP\_DISPLAY\_ONLY, [23](#)
  - GAPBOND\_IO\_CAP\_DISPLAY\_YES\_NO, [24](#)
  - GAPBOND\_IO\_CAP\_KEYBOARD\_DISPLAY, [24](#)
  - GAPBOND\_IO\_CAP\_KEYBOARD\_ONLY, [24](#)
  - GAPBOND\_IO\_CAP\_NO\_INPUT\_NO\_OUTPUT, [24](#)
  - GAPBOND\_PAIRING\_MODE\_INITIATE, [24](#)
  - GAPBOND\_PAIRING\_MODE\_NO\_PAIRING, [24](#)
  - GAPBOND\_PAIRING\_MODE\_WAIT\_FOR\_REQ, [24](#)
  - LE\_GAP\_ADV\_MAX\_SIZE, [24](#)
  - LeGapAddToResolvingList, [25](#)
  - LeGapAddToWhiteList, [25](#)
  - LeGapAdvertisingEnable, [25](#)
  - LeGapCentralConnectReq, [26](#)
  - LeGapCentralSetDataChannel, [26](#)
  - LeGapClearResolvingList, [27](#)
  - LeGapClearWhiteList, [27](#)
  - LeGapConnParaRequestRsp, [27](#)
  - LeGapConnUpdateRequest, [28](#)
  - LeGapConnUpdateResponse, [28](#)
  - LeGapConnectCancelReq, [27](#)
  - LeGapDisconnectReq, [29](#)
  - LeGapGenRandAddr, [29](#)
  - LeGapGetBtAddr, [29](#)
  - LeGapReadAdvChannelTxPower, [29](#)
  - LeGapReadChannelMap, [30](#)
  - LeGapReadResolvingListSize, [30](#)
  - LeGapReadRssi, [30](#)
  - LeGapReadTxPower, [31](#)
  - LeGapReadWhiteListSize, [31](#)
  - LeGapRemoveFromWhiteList, [31](#)
  - LeGapScanningReq, [32](#)
  - LeGapSetAdvData, [32](#)
  - LeGapSetAdvParameter, [33](#)
  - LeGapSetConnParameter, [33](#)
  - LeGapSetDataChannelPduLen, [33](#)
  - LeGapSetRandAddr, [34](#)
  - LeGapSetRpaTimeout, [34](#)
  - LeGapSetStaticAddr, [35](#)
  - LeSetScanParameter, [35](#)
  - LeSetScanRspData, [35](#)

- BLE GATT APIs, 37
  - CHAR\_AGGREGATE\_DESCRIPTOR, 41
  - CHAR\_CLIENT\_CONFIG\_DESCRIPTOR, 41
  - CHAR\_DECL\_UUID16\_ATTR\_VAL, 42
  - CHAR\_EXT\_PROP\_DESCRIPTOR, 42
  - CHAR\_PRESENT\_FORMAT\_DESCRIPTOR, 42
  - CHAR\_SERVER\_CONFIG\_DESCRIPTOR, 42
  - CHAR\_USER\_DESC\_DESCRIPTOR, 42
  - CHARACTERISTIC\_DECL\_UUID128, 42
  - CHARACTERISTIC\_DECL\_UUID16, 43
  - CHARACTERISTIC\_UUID128, 43
  - CHARACTERISTIC\_UUID16, 43
  - GATT\_CHAR\_AGG\_FORMAT\_UUID, 43
  - GATT\_CHAR\_EXT\_PROPS\_UUID, 43
  - GATT\_CHAR\_FORMAT\_UUID, 43
  - GATT\_CHAR\_USER\_DESC\_UUID, 44
  - GATT\_CHARACTERISTIC\_UUID, 44
  - GATT\_CLIENT\_CHAR\_CFG\_UUID, 44
  - GATT\_EXT\_REPORT\_REF\_UUID, 44
  - GATT\_INCLUDE\_UUID, 44
  - GATT\_PRIMARY\_SERVICE\_UUID, 44
  - GATT\_REPORT\_REF\_UUID, 44
  - GATT\_SECONDARY\_SERVICE\_UUID, 44
  - GATT\_SERV\_CHAR\_CFG\_UUID, 45
  - GATT\_VALID\_RANGE\_UUID, 45
  - gcCharAggregateUuid, 68
  - gcCharExtPropUuid, 68
  - gcCharFormatUuid, 69
  - gcCharUserDescUuid, 69
  - gcCharacteristicUuid, 68
  - gcClientCharConfigUuid, 69
  - gcExtReportRefUuid, 69
  - gcIncludeUuid, 69
  - gcPrimaryServiceUuid, 69
  - gcReportRefUuid, 69
  - gcSecondaryServiceUuid, 69
  - gcServerCharConfigUuid, 70
  - gcValidRangeUuid, 70
  - INCLUDE\_DECL\_UUID128, 45
  - INCLUDE\_DECL\_UUID128\_ATTR\_VAL, 45
  - INCLUDE\_DECL\_UUID16\_ATTR\_VAL, 45
  - INCLUDE\_DECL\_UUID16, 45
  - LE\_ATT\_UUID\_SIZE, 45
  - LE\_GATT\_CHAR\_PROP\_AUTH, 46
  - LE\_GATT\_CHAR\_PROP\_BCAST, 46
  - LE\_GATT\_CHAR\_PROP\_EXT\_PROP, 46
  - LE\_GATT\_CHAR\_PROP\_IND, 46
  - LE\_GATT\_CHAR\_PROP\_NTF, 46
  - LE\_GATT\_CHAR\_PROP\_RD, 46
  - LE\_GATT\_CHAR\_PROP\_WR\_NO\_RESP, 47
  - LE\_GATT\_CHAR\_PROP\_WR, 46
  - LE\_GATT\_CLIENT\_CFG\_INDICATION, 47
  - LE\_GATT\_CLIENT\_CFG\_NOTIFICATION, 47
  - LE\_GATT\_EXT\_PROP\_RELIABLE\_WR, 47
  - LE\_GATT\_EXT\_PROP\_WR\_AUX, 47
  - LE\_GATT\_FLAG\_PREPARE\_WRITE, 47
  - LE\_GATT\_FLAG\_WRITE\_CMD, 47
  - LE\_GATT\_FLAG\_WRITE\_REQ, 47
  - LE\_GATT\_PERM\_AUTH\_READABLE, 48
  - LE\_GATT\_PERM\_AUTH\_WRITABLE, 48
  - LE\_GATT\_PERM\_NONE, 48
  - LE\_GATT\_PERM\_READ, 48
  - LE\_GATT\_PERM\_RELIABLE\_WRITE, 48
  - LE\_GATT\_PERM\_WRITE\_CMD, 48
  - LE\_GATT\_PERM\_WRITE\_REQ, 48
  - LE\_GATT\_PERMIT\_AUTHEN\_READ, 48
  - LE\_GATT\_PERMIT\_AUTHEN\_WRITE, 49
  - LE\_GATT\_PERMIT\_AUTHOR\_READ, 49
  - LE\_GATT\_PERMIT\_AUTHOR\_WRITE, 49
  - LE\_GATT\_PERMIT\_ENCRYPT\_READ, 49
  - LE\_GATT\_PERMIT\_ENCRYPT\_WRITE, 49
  - LE\_GATT\_PERMIT\_READABLE, 49
  - LE\_GATT\_PERMIT\_READ, 49
  - LE\_GATT\_PERMIT\_SC\_AUTHEN\_READ, 49
  - LE\_GATT\_PERMIT\_SC\_AUTHEN\_WRITE, 50
  - LE\_GATT\_PERMIT\_WRITABLE, 50
  - LE\_GATT\_PERMIT\_WRITE, 50
  - LeGattAccessReadRsp, 52
  - LeGattAccessWriteRsp, 52
  - LeGattChangeAttrVal, 53
  - LeGattCharValConfirmation, 53
  - LeGattCharValIndicate, 54
  - LeGattCharValNotify, 54
  - LeGattExchangeMtuReq, 55
  - LeGattExchangeMtuRsp, 55
  - LeGattExecuteWriteCharValReliable, 55
  - LeGattFindAllCharDescriptor, 56
  - LeGattFindAllCharacteristic, 56
  - LeGattFindAllPrimaryService, 57
  - LeGattFindCharacteristicByUuid, 57
  - LeGattFindIncludedService, 58
  - LeGattFindPrimaryServiceByUuid, 58
  - LeGattGetAttrHandle, 58
  - LeGattGetAttrVal, 59
  - LeGattGetAttrValLen, 59
  - LeGattGetAttrValMaxLen, 61
  - LeGattInit, 61
  - LeGattModifyAttrVal, 62
  - LeGattPrepareWriteCharValReliable, 62
  - LeGattReadCharValByUuid, 63
  - LeGattReadCharValue, 63
  - LeGattReadLongCharVal, 64
  - LeGattReadMultipleCharVal, 64
  - LeGattRegisterIncludeService, 64
  - LeGattRegisterService, 65
  - LeGattSignedWriteNoRsp, 65
  - LeGattStopCurrentProcedure, 66
  - LeGattWriteCharVal, 66
  - LeGattWriteCharValReliable, 67
  - LeGattWriteLongCharVal, 67
  - LeGattWriteNoRsp, 68
  - PRIMARY\_SERVICE\_DECL\_UUID128, 50
  - PRIMARY\_SERVICE\_DECL\_UUID16, 50
  - SECONDARY\_SERVICE\_DECL\_UUID128, 50
  - SECONDARY\_SERVICE\_DECL\_UUID16, 50
- BLE MSG APIs, 71

- LE\_ATT\_MSG\_BASE, [72](#)
- LE\_CM\_MSG\_BASE, [72](#)
- LE\_GATT\_MSG\_BASE, [72](#)
- LE\_HCI\_MSG\_BASE, [73](#)
- LE\_L2CAP\_MSG\_BASE, [73](#)
- LE\_SMP\_MSG\_BASE, [73](#)
- LE\_SYS\_MSG\_BASE, [73](#)
- LeCancelAllMessage, [76](#)
- LeCancelAllSubMessage, [77](#)
- LeCancelFirstMessage, [77](#)
- LeCancelFirstSubMessage, [77](#)
- LeGetSubMsgId, [78](#)
- LeHostCreateTask, [78](#)
- LeHostMessageLoop, [79](#)
- LeSendMessage, [79](#)
- LeSendMessageAfter, [79](#)
- LeSendMessageUnlock, [80](#)
- LeSendSubMessage, [80](#)
- LeSendSubMessageAfter, [81](#)
- LeSendSubMessageUnlock, [81](#)
- MESSAGE\_ALLOCATE, [73](#)
- MESSAGE\_BULID, [73](#)
- MESSAGE\_DATA\_BULID, [73](#)
- MESSAGE\_OFFSET, [74](#)
- MESSAGEID, [74](#)
- MESSAGE, [74](#)
- MSGLOCK, [75](#)
- MSGSUBID, [75](#)
- MSGTIMER, [75](#)
- MsgData, [75](#)
- MsgLock, [75](#)
- T\_HOUR, [74](#)
- T\_MIN, [74](#)
- T\_SEC, [74](#)
- TASKHANDLER, [75](#)
- TASKPACK, [76](#)
- TASK, [75](#)
- Task, [75](#)
- BLE SMP APIs, [83](#)
  - LE\_MAX\_BOND\_COUNT, [84](#)
  - LE\_SM\_IO\_CAP\_DISP\_ONLY, [84](#)
  - LE\_SM\_IO\_CAP\_DISP\_YES\_NO, [84](#)
  - LE\_SM\_IO\_CAP\_KEYBOARD\_DISP, [84](#)
  - LE\_SM\_IO\_CAP\_KEYBOARD\_ONLY, [85](#)
  - LE\_SM\_IO\_CAP\_NO\_IO, [85](#)
  - LE\_SM\_PAIR\_MITM\_NO, [85](#)
  - LE\_SM\_PAIR\_MITM\_YES, [85](#)
  - LE\_SM\_PAIR\_OOB\_NO, [85](#)
  - LE\_SM\_PAIR\_OOB\_YES, [85](#)
  - LE\_SM\_PAIR\_SC\_NO, [85](#)
  - LE\_SM\_PAIR\_SC\_YES, [85](#)
  - LeSmpInit, [87](#)
  - LeSmpOobAuthDataRsp, [87](#)
  - LeSmpOobPresent, [87](#)
  - LeSmpPasskeyInput, [88](#)
  - LeSmpScOobComputeConfirmVal, [88](#)
  - LeSmpScOobDataRsp, [88](#)
  - LeSmpSecurityReq, [89](#)
  - LeSmpSecurityRsp, [89](#)
  - LeSmpSetDefaultConfig, [90](#)
  - LeSmpUserConfirmRsp, [90](#)
- bd\_addr
  - LE\_CM\_MSG\_READ\_BD\_ADDR\_CFM\_T, [144](#)
- beacon\_interval
  - auto\_conn\_info\_t, [127](#)
  - mw\_wifi\_auto\_connect\_ap\_info\_t, [196](#)
  - wifi\_ap\_config\_t, [203](#)
  - wifi\_auto\_connect\_info\_f, [205](#)
  - wifi\_scan\_info\_t, [216](#)
- bondable
  - LE\_SMP\_MSG\_SLAVE\_SECURITY\_REQUEST\_IND\_T, [193](#)
- bonded
  - LE\_SMP\_MSG\_PAIRING\_COMPLETE\_IND\_T, [190](#)
- bssid
  - auto\_conn\_info\_t, [128](#)
  - mw\_wifi\_auto\_connect\_ap\_info\_t, [196](#)
  - wifi\_auto\_connect\_info\_f, [205](#)
  - wifi\_event\_sta\_connected\_t, [210](#)
  - wifi\_event\_sta\_disconnected\_t, [211](#)
  - wifi\_scan\_config\_t, [215](#)
  - wifi\_scan\_info\_t, [216](#)
  - wifi\_sta\_config\_t, [219](#)
- bssid\_present
  - wifi\_sta\_config\_t, [219](#)
- CHARAggregateDescriptor
  - BLE GATT APIs, [41](#)
- CHARClientConfigDescriptor
  - BLE GATT APIs, [41](#)
- CHAR\_DECL\_UUID16\_ATTR\_VAL
  - BLE GATT APIs, [42](#)
- CHAR\_EXT\_PROP\_DESCRIPTOR
  - BLE GATT APIs, [42](#)
- CHAR\_PRESENT\_FORMAT\_DESCRIPTOR
  - BLE GATT APIs, [42](#)
- CHAR\_SERVER\_CONFIG\_DESCRIPTOR
  - BLE GATT APIs, [42](#)
- CHAR\_USER\_DESC\_DESCRIPTOR
  - BLE GATT APIs, [42](#)
- CHARACTERISTIC\_DECL\_UUID128
  - BLE GATT APIs, [42](#)
- CHARACTERISTIC\_DECL\_UUID16
  - BLE GATT APIs, [43](#)
- CHARACTERISTIC\_UUID128
  - BLE GATT APIs, [43](#)
- CHARACTERISTIC\_UUID16
  - BLE GATT APIs, [43](#)
- capabilities
  - auto\_conn\_info\_t, [128](#)
  - mw\_wifi\_auto\_connect\_ap\_info\_t, [196](#)
  - wifi\_auto\_connect\_info\_f, [206](#)
- capability\_info
  - wifi\_scan\_info\_t, [216](#)
- ch\_map

- LE\_CM\_MSG\_READ\_CHANNEL\_MAP\_CFM\_T, 145
- channel
  - wifi\_ap\_config\_t, 203
  - wifi\_event\_sta\_connected\_t, 210
  - wifi\_scan\_config\_t, 215
  - wifi\_scan\_info\_t, 216
- channel\_map
  - LE\_GAP\_ADVERTISING\_PARAM\_T, 152
- client\_rx\_mtu
  - LE\_GATT\_MSG\_EXCHANGE\_MTU\_IND\_T, 164
- confirm
  - LE\_SMP\_SC\_OOB\_DATA\_T, 194
- confirm\_num
  - LE\_SMP\_MSG\_USER\_CONFIRM\_IND\_T, 194
- conn\_hdl
  - LE\_CM\_CONNECTION\_COMPLETE\_IND\_T, 133
  - LE\_CM\_MSG\_CONN\_PARA\_REQ\_T, 136
  - LE\_CM\_MSG\_CONN\_UPDATE\_COMPLETE\_I↔ND\_T, 137
  - LE\_CM\_MSG\_DATA\_LEN\_CHANGE\_IND\_T, 138
  - LE\_CM\_MSG\_DISCONNECT\_COMPLETE\_IN↔D\_T, 140
  - LE\_CM\_MSG\_ENCRYPTION\_CHANGE\_IND\_T, 140
  - LE\_CM\_MSG\_ENCRYPTION\_REFRESH\_IND\_T, 141
  - LE\_CM\_MSG\_LTK\_REQ\_IND\_T, 143
  - LE\_CM\_MSG\_READ\_CHANNEL\_MAP\_CFM\_T, 145
  - LE\_CM\_MSG\_READ\_RSSI\_CFM\_T, 146
  - LE\_CM\_MSG\_READ\_TX\_POWER\_CFM\_T, 147
  - LE\_CM\_MSG\_SET\_DATA\_LENGTH\_CFM\_T, 148
  - LE\_CM\_MSG\_SIGNAL\_UPDATE\_REQ\_T, 149
  - LE\_GATT\_MSG\_ACCESS\_READ\_IND\_T, 157
  - LE\_GATT\_MSG\_ACCESS\_WRITE\_IND\_T, 157
  - LE\_GATT\_MSG\_CHAR\_DESCRIPTOR\_INFO↔IND\_T, 159
  - LE\_GATT\_MSG\_CHARACTERISTIC\_DECL\_IN↔FO\_IND\_T, 160
  - LE\_GATT\_MSG\_CHARACTERISTIC\_VAL\_IN↔T, 161
  - LE\_GATT\_MSG\_CONFIRMATION\_CFM\_T, 162
  - LE\_GATT\_MSG\_EXCHANGE\_MTU\_CFM\_T, 163
  - LE\_GATT\_MSG\_EXCHANGE\_MTU\_IND\_T, 164
  - LE\_GATT\_MSG\_EXECUTE\_WRITE\_RELIABL↔E\_CFM\_T, 165
  - LE\_GATT\_MSG\_FIND\_ALL\_CHAR\_DESC\_CF↔M\_T, 166
  - LE\_GATT\_MSG\_FIND\_ALL\_PRIMARY\_SERVI↔CE\_CFM\_T, 167
  - LE\_GATT\_MSG\_FIND\_CHARACTERISTIC\_CF↔M\_T, 168
  - LE\_GATT\_MSG\_FIND\_INCLUDED\_SERVICE↔CFM\_T, 169
  - LE\_GATT\_MSG\_FIND\_PRIMARY\_SERVICE\_B↔Y\_UUID\_CFM\_T, 170
  - LE\_GATT\_MSG\_INCLUDE\_SERVICE\_INFO\_I↔ND\_T, 171
  - LE\_GATT\_MSG\_INDICATE\_IND\_T, 172
  - LE\_GATT\_MSG\_NOTIFY\_CFM\_T, 173
  - LE\_GATT\_MSG\_NOTIFY\_IND\_T, 174
  - LE\_GATT\_MSG\_OPERATION\_TIMEOUT\_T, 175
  - LE\_GATT\_MSG\_PREPARE\_WRITE\_RELIABL↔E\_CFM\_T, 176
  - LE\_GATT\_MSG\_READ\_CHAR\_VAL\_BY\_UUID↔CFM\_T, 177
  - LE\_GATT\_MSG\_READ\_CHARACTERISTIC\_V↔ALUE\_CFM\_T, 178
  - LE\_GATT\_MSG\_READ\_LONG\_CHAR\_VAL\_C↔FM\_T, 179
  - LE\_GATT\_MSG\_READ\_MULTIPLE\_CHAR\_VA↔L\_CFM\_T, 180
  - LE\_GATT\_MSG\_SERVICE\_INFO\_IND\_T, 181
  - LE\_GATT\_MSG\_SIGNED\_WRITE\_CFM\_T, 182
  - LE\_GATT\_MSG\_WRITE\_CHAR\_VAL\_RELIAB↔LE\_CFM\_T, 183
  - LE\_GATT\_MSG\_WRITE\_CHAR\_VALUE\_CFM↔T, 184
  - LE\_GATT\_MSG\_WRITE\_LONG\_CHAR\_VALU↔E\_CFM\_T, 185
  - LE\_GATT\_MSG\_WRITE\_NO\_RSP\_CFM\_T, 186
  - LE\_SMP\_MSG\_ENCRYPTION\_CHANGE\_IN↔D\_T, 188
  - LE\_SMP\_MSG\_ENCRYPTION\_REFRESH\_IN↔D\_T, 188
  - LE\_SMP\_MSG\_OOB\_DATA\_REQUEST\_IND\_T, 189
  - LE\_SMP\_MSG\_PAIRING\_ACTION\_IND\_T, 189
  - LE\_SMP\_MSG\_PAIRING\_COMPLETE\_IND\_T, 190
  - LE\_SMP\_MSG\_PASSKEY\_DISPLAY\_IND\_T, 191
  - LE\_SMP\_MSG\_PASSKEY\_INPUT\_IND\_T, 192
  - LE\_SMP\_MSG\_SC\_OOB\_DATA\_REQUEST\_I↔ND\_T, 192
  - LE\_SMP\_MSG\_SLAVE\_SECURITY\_REQUES↔T\_IND\_T, 193
  - LE\_SMP\_MSG\_USER\_CONFIRM\_IND\_T, 194
  - LE\_SYS\_MSG\_BUF\_OVERFLOW\_T, 195
- conn\_interval
  - LE\_CM\_CONNECTION\_COMPLETE\_IND\_T, 133
- conn\_latency
  - LE\_CM\_CONNECTION\_COMPLETE\_IND\_T, 133
- connected
  - wifi\_event\_info\_t, 209
- current\_rx\_mtu
  - LE\_GATT\_MSG\_EXCHANGE\_MTU\_CFM\_T, 163
- data
  - LE\_CM\_MSG\_ADVERTISE\_REPORT\_IND↔T, 135
- dev\_id
  - LE\_CM\_CONNECTION\_COMPLETE\_IND\_T, 133
- devid
  - LE\_CM\_MSG\_ENCRYPTION\_CHANGE\_IND\_T, 141

- LE\_CM\_MSG\_ENCRYPTION\_REFRESH\_IND\_T, 141
- LE\_CM\_MSG\_LTK\_REQ\_IND\_T, 143
- LE\_GATT\_MSG\_ACCESS\_READ\_IND\_T, 157
- LE\_GATT\_MSG\_ACCESS\_WRITE\_IND\_T, 158
- LE\_GATT\_MSG\_CHAR\_DESCRIPTOR\_INFO\_IND\_T, 159
- LE\_GATT\_MSG\_CHARACTERISTIC\_DECL\_IND\_T, 160
- LE\_GATT\_MSG\_CHARACTERISTIC\_VAL\_IND\_T, 161
- LE\_GATT\_MSG\_CONFIRMATION\_CFM\_T, 163
- LE\_GATT\_MSG\_EXCHANGE\_MTU\_CFM\_T, 163
- LE\_GATT\_MSG\_EXCHANGE\_MTU\_IND\_T, 164
- LE\_GATT\_MSG\_EXECUTE\_WRITE\_RELIABLE\_CFM\_T, 165
- LE\_GATT\_MSG\_FIND\_ALL\_CHAR\_DESC\_CFM\_T, 166
- LE\_GATT\_MSG\_FIND\_ALL\_PRIMARY\_SERVICE\_CFM\_T, 167
- LE\_GATT\_MSG\_FIND\_CHARACTERISTIC\_CFM\_T, 168
- LE\_GATT\_MSG\_FIND\_INCLUDED\_SERVICE\_CFM\_T, 169
- LE\_GATT\_MSG\_FIND\_PRIMARY\_SERVICE\_BY\_UUID\_CFM\_T, 170
- LE\_GATT\_MSG\_INCLUDE\_SERVICE\_INFO\_IND\_T, 171
- LE\_GATT\_MSG\_INDICATE\_IND\_T, 172
- LE\_GATT\_MSG\_NOTIFY\_CFM\_T, 173
- LE\_GATT\_MSG\_NOTIFY\_IND\_T, 174
- LE\_GATT\_MSG\_OPERATION\_TIMEOUT\_T, 175
- LE\_GATT\_MSG\_PREPARE\_WRITE\_RELIABLE\_CFM\_T, 176
- LE\_GATT\_MSG\_READ\_CHAR\_VAL\_BY\_UUID\_CFM\_T, 177
- LE\_GATT\_MSG\_READ\_CHARACTERISTIC\_VALUE\_CFM\_T, 178
- LE\_GATT\_MSG\_READ\_LONG\_CHAR\_VAL\_CFM\_T, 179
- LE\_GATT\_MSG\_READ\_MULTIPLE\_CHAR\_VAL\_CFM\_T, 180
- LE\_GATT\_MSG\_SERVICE\_INFO\_IND\_T, 181
- LE\_GATT\_MSG\_SIGNED\_WRITE\_CFM\_T, 182
- LE\_GATT\_MSG\_WRITE\_CHAR\_VAL\_RELIABLE\_CFM\_T, 183
- LE\_GATT\_MSG\_WRITE\_CHAR\_VALUE\_CFM\_T, 184
- LE\_GATT\_MSG\_WRITE\_LONG\_CHAR\_VALUE\_CFM\_T, 185
- LE\_GATT\_MSG\_WRITE\_NO\_RSP\_CFM\_T, 186
- direct\_addr
  - LE\_CM\_MSG\_DIRECT\_ADV\_REPORT\_IND\_T, 139
- direct\_addr\_type
  - LE\_CM\_MSG\_DIRECT\_ADV\_REPORT\_IND\_T, 139
- disconnected
  - wifi\_event\_info\_t, 209
- dtim\_prod
  - auto\_conn\_info\_t, 128
  - mw\_wifi\_auto\_connect\_ap\_info\_t, 196
  - wifi\_auto\_connect\_info\_f, 206
- ediv
  - LE\_CM\_MSG\_LTK\_REQ\_IND\_T, 143
- enable
  - LE\_SMP\_MSG\_ENCRYPTION\_CHANGE\_IND\_T, 188
- enabled
  - LE\_CM\_MSG\_ENCRYPTION\_CHANGE\_IND\_T, 141
  - LE\_CM\_MSG\_ENCRYPTION\_REFRESH\_IND\_T, 142
- encrypt\_type
  - wifi\_ap\_config\_t, 204
- end\_hdl
  - LE\_GATT\_MSG\_INCLUDE\_SERVICE\_INFO\_IND\_T, 171
  - LE\_GATT\_MSG\_SERVICE\_INFO\_IND\_T, 181
- endHdl
  - LE\_GATT\_SERVICE\_T, 187
- Enumeration, 121
  - wifi\_auth\_mode\_t, 121
  - wifi\_bandwidth\_t, 122
  - wifi\_cipher\_type\_t, 122
  - wifi\_event\_t, 122
  - wifi\_mode\_t, 123
  - wifi\_reason\_code\_t, 123
  - wifi\_scan\_method\_t, 124
  - wifi\_scan\_type\_t, 124
  - wifi\_sort\_method\_t, 126
- err\_hdl
  - LE\_GATT\_MSG\_EXECUTE\_WRITE\_RELIABLE\_CFM\_T, 165
  - LE\_GATT\_MSG\_READ\_MULTIPLE\_CHAR\_VAL\_CFM\_T, 180
- event
  - event\_msg\_t, 131
- event\_handler
  - wifi\_init\_config\_t, 214
- event\_msg\_t, 131
  - event, 131
  - length, 132
  - param, 132
- event\_type
  - LE\_CM\_MSG\_ADVERTISE\_REPORT\_IND\_T, 135
- fast\_connect
  - auto\_conn\_info\_t, 128
  - mw\_wifi\_auto\_connect\_ap\_info\_t, 196
  - wifi\_auto\_connect\_info\_f, 206
- filter\_policy
  - LE\_GAP\_ADVERTISING\_PARAM\_T, 152
  - LE\_GAP\_SCAN\_PARAM\_T, 154
- flag



- auto\_connect\_cfg\_t, 130
- LE\_GATT\_MSG\_ACCESS\_WRITE\_IND\_T, 158
- MwFimAutoConnectCFG\_t, 198
- format
  - LE\_GATT\_ATTR\_T, 155
  - LE\_GATT\_MSG\_CHAR\_DESCRIPTOR\_INFO\_IND\_T, 159
  - LE\_GATT\_MSG\_CHARACTERISTIC\_DECL\_INFO\_IND\_T, 160
  - LE\_GATT\_MSG\_INCLUDE\_SERVICE\_INFO\_IND\_T, 171
  - LE\_GATT\_MSG\_SERVICE\_INFO\_IND\_T, 181
- free\_ocpy
  - auto\_conn\_info\_t, 128
  - mw\_wifi\_auto\_connect\_ap\_info\_t, 196
  - wifi\_auto\_connect\_info\_f, 206
- front
  - auto\_connect\_cfg\_t, 130
  - MwFimAutoConnectCFG\_t, 198
- GAP\_ADTYPE\_128BIT\_COMPLETE
  - BLE GAP APIs, 18
- GAP\_ADTYPE\_128BIT\_MORE
  - BLE GAP APIs, 18
- GAP\_ADTYPE\_16BIT\_COMPLETE
  - BLE GAP APIs, 18
- GAP\_ADTYPE\_16BIT\_MORE
  - BLE GAP APIs, 18
- GAP\_ADTYPE\_32BIT\_COMPLETE
  - BLE GAP APIs, 19
- GAP\_ADTYPE\_32BIT\_MORE
  - BLE GAP APIs, 19
- GAP\_ADTYPE\_3D\_INFO\_DATA
  - BLE GAP APIs, 19
- GAP\_ADTYPE\_ADV\_INTERVAL
  - BLE GAP APIs, 19
- GAP\_ADTYPE\_APPEARANCE
  - BLE GAP APIs, 19
- GAP\_ADTYPE\_FLAGS\_BREDR\_NOT\_SUPPORTED
  - BLE GAP APIs, 19
- GAP\_ADTYPE\_FLAGS\_GENERAL
  - BLE GAP APIs, 19
- GAP\_ADTYPE\_FLAGS\_LIMITED
  - BLE GAP APIs, 20
- GAP\_ADTYPE\_FLAGS
  - BLE GAP APIs, 19
- GAP\_ADTYPE\_LE\_BD\_ADDR
  - BLE GAP APIs, 20
- GAP\_ADTYPE\_LE\_ROLE
  - BLE GAP APIs, 20
- GAP\_ADTYPE\_LOCAL\_NAME\_COMPLETE
  - BLE GAP APIs, 20
- GAP\_ADTYPE\_LOCAL\_NAME\_SHORT
  - BLE GAP APIs, 20
- GAP\_ADTYPE\_MANUFACTURER\_SPECIFIC
  - BLE GAP APIs, 20
- GAP\_ADTYPE\_OOB\_CLASS\_OF\_DEVICE
  - BLE GAP APIs, 20
- GAP\_ADTYPE\_OOB\_SIMPLE\_PAIRING\_HASHC
  - BLE GAP APIs, 20
- GAP\_ADTYPE\_OOB\_SIMPLE\_PAIRING\_RANDR
  - BLE GAP APIs, 21
- GAP\_ADTYPE\_POWER\_LEVEL
  - BLE GAP APIs, 21
- GAP\_ADTYPE\_PUBLIC\_TARGET\_ADDR
  - BLE GAP APIs, 21
- GAP\_ADTYPE\_RANDOM\_TARGET\_ADDR
  - BLE GAP APIs, 21
- GAP\_ADTYPE\_SERVICE\_DATA\_128BIT
  - BLE GAP APIs, 21
- GAP\_ADTYPE\_SERVICE\_DATA\_32BIT
  - BLE GAP APIs, 21
- GAP\_ADTYPE\_SERVICE\_DATA
  - BLE GAP APIs, 21
- GAP\_ADTYPE\_SERVICES\_LIST\_128BIT
  - BLE GAP APIs, 21
- GAP\_ADTYPE\_SERVICES\_LIST\_16BIT
  - BLE GAP APIs, 22
- GAP\_ADTYPE\_SIGNED\_DATA
  - BLE GAP APIs, 22
- GAP\_ADTYPE\_SIMPLE\_PAIRING\_HASHC\_256
  - BLE GAP APIs, 22
- GAP\_ADTYPE\_SIMPLE\_PAIRING\_RANDR\_256
  - BLE GAP APIs, 22
- GAP\_ADTYPE\_SLAVE\_CONN\_INTERVAL\_RANGE
  - BLE GAP APIs, 22
- GAP\_ADTYPE\_SM\_OOB\_FLAG
  - BLE GAP APIs, 22
- GAP\_ADTYPE\_SM\_TK
  - BLE GAP APIs, 22
- GAP\_PUBLIC\_ADDR
  - BLE GAP APIs, 22
- GAP\_RAND\_ADDR\_NRPA
  - BLE GAP APIs, 23
- GAP\_RAND\_ADDR\_RPA
  - BLE GAP APIs, 23
- GAP\_RAND\_ADDR\_STATIC
  - BLE GAP APIs, 23
- GAP\_SCAN\_TYPE\_ACTIVE
  - BLE GAP APIs, 23
- GAP\_SCAN\_TYPE\_PASSIVE
  - BLE GAP APIs, 23
- GAP\_TX\_PWR\_CURR\_VAL
  - BLE GAP APIs, 23
- GAP\_TX\_PWR\_MAX\_VAL
  - BLE GAP APIs, 23
- GAPBOND\_IO\_CAP\_DISPLAY\_ONLY
  - BLE GAP APIs, 23
- GAPBOND\_IO\_CAP\_DISPLAY\_YES\_NO
  - BLE GAP APIs, 24
- GAPBOND\_IO\_CAP\_KEYBOARD\_DISPLAY
  - BLE GAP APIs, 24
- GAPBOND\_IO\_CAP\_KEYBOARD\_ONLY
  - BLE GAP APIs, 24
- GAPBOND\_IO\_CAP\_NO\_INPUT\_NO\_OUTPUT
  - BLE GAP APIs, 24
- GAPBOND\_PAIRING\_MODE\_INITIATE

- BLE GAP APIs, [24](#)
- GAPBOND\_PAIRING\_MODE\_NO\_PAIRING
  - BLE GAP APIs, [24](#)
- GAPBOND\_PAIRING\_MODE\_WAIT\_FOR\_REQ
  - BLE GAP APIs, [24](#)
- GATT\_CHAR\_AGG\_FORMAT\_UUID
  - BLE GATT APIs, [43](#)
- GATT\_CHAR\_EXT\_PROPS\_UUID
  - BLE GATT APIs, [43](#)
- GATT\_CHAR\_FORMAT\_UUID
  - BLE GATT APIs, [43](#)
- GATT\_CHAR\_USER\_DESC\_UUID
  - BLE GATT APIs, [44](#)
- GATT\_CHARACTERISTIC\_UUID
  - BLE GATT APIs, [44](#)
- GATT\_CLIENT\_CHAR\_CFG\_UUID
  - BLE GATT APIs, [44](#)
- GATT\_EXT\_REPORT\_REF\_UUID
  - BLE GATT APIs, [44](#)
- GATT\_INCLUDE\_UUID
  - BLE GATT APIs, [44](#)
- GATT\_PRIMARY\_SERVICE\_UUID
  - BLE GATT APIs, [44](#)
- GATT\_REPORT\_REF\_UUID
  - BLE GATT APIs, [44](#)
- GATT\_SECONDARY\_SERVICE\_UUID
  - BLE GATT APIs, [44](#)
- GATT\_SERV\_CHAR\_CFG\_UUID
  - BLE GATT APIs, [45](#)
- GATT\_VALID\_RANGE\_UUID
  - BLE GATT APIs, [45](#)
- gcCharAggregateUuid
  - BLE GATT APIs, [68](#)
- gcCharExtPropUuid
  - BLE GATT APIs, [68](#)
- gcCharFormatUuid
  - BLE GATT APIs, [69](#)
- gcCharUserDescUuid
  - BLE GATT APIs, [69](#)
- gcCharacteristicUuid
  - BLE GATT APIs, [68](#)
- gcClientCharConfigUuid
  - BLE GATT APIs, [69](#)
- gcExtReportRefUuid
  - BLE GATT APIs, [69](#)
- gcIncludeUuid
  - BLE GATT APIs, [69](#)
- gcPrimaryServiceUuid
  - BLE GATT APIs, [69](#)
- gcReportRefUuid
  - BLE GATT APIs, [69](#)
- gcSecondaryServiceUuid
  - BLE GATT APIs, [69](#)
- gcServerCharConfigUuid
  - BLE GATT APIs, [70](#)
- gcValidRangeUuid
  - BLE GATT APIs, [70](#)
- got\_ip
  - wifi\_event\_info\_t, [209](#)
- group\_cipher
  - wifi\_scan\_info\_t, [217](#)
- handle
  - LE\_CM\_MSG\_SET\_DISCONNECT\_CFM\_T, [149](#)
  - LE\_GATT\_ATTR\_T, [155](#)
  - LE\_GATT\_MSG\_ACCESS\_READ\_IND\_T, [157](#)
  - LE\_GATT\_MSG\_ACCESS\_WRITE\_IND\_T, [158](#)
  - LE\_GATT\_MSG\_CHAR\_DESCRIPTOR\_INFO\_↵  
IND\_T, [159](#)
  - LE\_GATT\_MSG\_CHARACTERISTIC\_DECL\_IN↵  
FO\_IND\_T, [160](#)
  - LE\_GATT\_MSG\_CHARACTERISTIC\_VAL\_IND↵  
\_T, [162](#)
  - LE\_GATT\_MSG\_CONFIRMATION\_CFM\_T, [163](#)
  - LE\_GATT\_MSG\_FIND\_ALL\_CHAR\_DESC\_CF↵  
M\_T, [166](#)
  - LE\_GATT\_MSG\_FIND\_ALL\_PRIMARY\_SERVI↵  
CE\_CFM\_T, [167](#)
  - LE\_GATT\_MSG\_FIND\_CHARACTERISTIC\_CF↵  
M\_T, [168](#)
  - LE\_GATT\_MSG\_FIND\_INCLUDED\_SERVICE\_↵  
CFM\_T, [169](#)
  - LE\_GATT\_MSG\_FIND\_PRIMARY\_SERVICE\_B↵  
Y\_UUID\_CFM\_T, [170](#)
  - LE\_GATT\_MSG\_INCLUDE\_SERVICE\_INFO\_I↵  
ND\_T, [171](#)
  - LE\_GATT\_MSG\_INDICATE\_IND\_T, [172](#)
  - LE\_GATT\_MSG\_NOTIFY\_CFM\_T, [173](#)
  - LE\_GATT\_MSG\_NOTIFY\_IND\_T, [174](#)
  - LE\_GATT\_MSG\_PREPARE\_WRITE\_RELIABL↵  
E\_CFM\_T, [176](#)
  - LE\_GATT\_MSG\_READ\_CHAR\_VAL\_BY\_UUID↵  
\_CFM\_T, [177](#)
  - LE\_GATT\_MSG\_READ\_CHARACTERISTIC\_V↵  
ALUE\_CFM\_T, [178](#)
  - LE\_GATT\_MSG\_READ\_LONG\_CHAR\_VAL\_C↵  
FM\_T, [179](#)
  - LE\_GATT\_MSG\_SIGNED\_WRITE\_CFM\_T, [182](#)
  - LE\_GATT\_MSG\_WRITE\_CHAR\_VAL\_RELIAB↵  
LE\_CFM\_T, [183](#)
  - LE\_GATT\_MSG\_WRITE\_CHAR\_VALUE\_CFM↵  
\_T, [184](#)
  - LE\_GATT\_MSG\_WRITE\_LONG\_CHAR\_VALU↵  
E\_CFM\_T, [185](#)
  - LE\_GATT\_MSG\_WRITE\_NO\_RSP\_CFM\_T, [186](#)
- hid\_ssids
  - auto\_conn\_info\_t, [128](#)
  - mw\_wifi\_auto\_connect\_ap\_info\_t, [196](#)
  - wifi\_auto\_connect\_info\_f, [206](#)
- iArgc
  - T\_RfCmd, [199](#)
- INCLUDE\_DECL\_UUID128
  - BLE GATT APIs, [45](#)
- INCLUDE\_DECL\_UUID128\_ATTR\_VAL
  - BLE GATT APIs, [45](#)
- INCLUDE\_DECL\_UUID16\_ATTR\_VAL

- BLE GATT APIs, 45
- INCLUDE\_DECL\_UUINT16
  - BLE GATT APIs, 45
- identifier
  - LE\_CM\_MSG\_SIGNAL\_UPDATE\_REQ\_T, 149
- interval
  - LE\_CM\_MSG\_CONN\_UPDATE\_COMPLETE\_IND\_T, 137
  - LE\_GAP\_SCAN\_PARAM\_T, 154
- interval\_max
  - LE\_CM\_MSG\_SIGNAL\_UPDATE\_REQ\_T, 150
  - LE\_GAP\_ADVERTISING\_PARAM\_T, 152
  - LE\_GAP\_CONN\_PARAM\_T, 153
- interval\_min
  - LE\_CM\_MSG\_SIGNAL\_UPDATE\_REQ\_T, 150
  - LE\_GAP\_ADVERTISING\_PARAM\_T, 152
  - LE\_GAP\_CONN\_PARAM\_T, 153
- ip\_changed
  - wifi\_event\_sta\_got\_ip\_t, 212
- itv\_max
  - LE\_CM\_MSG\_CONN\_PARA\_REQ\_T, 136
  - LE\_CONN\_PARA\_T, 151
- itv\_min
  - LE\_CM\_MSG\_CONN\_PARA\_REQ\_T, 136
  - LE\_CONN\_PARA\_T, 151
- keypress
  - LE\_SMP\_MSG\_SLAVE\_SECURITY\_REQUEST\_IND\_T, 193
- LE\_ATT\_MSG\_BASE
  - BLE MSG APIs, 72
- LE\_ATT\_UUID\_SIZE
  - BLE GATT APIs, 45
- LE\_BT\_ADDR\_T, 132
  - addr, 132
  - type, 132
- LE\_CM\_CONNECTION\_COMPLETE\_IND\_T, 133
  - conn\_hdl, 133
  - conn\_interval, 133
  - conn\_latency, 133
  - dev\_id, 133
  - peer\_addr, 133
  - peer\_addr\_type, 134
  - role, 134
  - status, 134
  - supervision\_timeout, 134
- LE\_CM\_MSG\_ADD\_TO\_RESOLVING\_LIST\_CFM\_T
  - BLE CM APIs, 11
- LE\_CM\_MSG\_ADD\_TO\_WHITE\_LIST\_CFM\_T
  - BLE CM APIs, 11
- LE\_CM\_MSG\_ADVERTISE\_REPORT\_IND\_T, 134
  - addr, 135
  - addr\_type, 135
  - data, 135
  - event\_type, 135
  - len, 135
  - rss\_i, 135
- LE\_CM\_MSG\_BASE
  - BLE MSG APIs, 72
- LE\_CM\_MSG\_CANCEL\_CONNECTION\_CFM\_T
  - BLE CM APIs, 11
- LE\_CM\_MSG\_CLEAR\_RESOLVING\_LIST\_CFM\_T
  - BLE CM APIs, 12
- LE\_CM\_MSG\_CLEAR\_WHITE\_LIST\_CFM\_T
  - BLE CM APIs, 12
- LE\_CM\_MSG\_CONN\_PARA\_REQ\_T, 135
  - conn\_hdl, 136
  - itv\_max, 136
  - itv\_min, 136
  - latency, 136
  - sv\_tmo, 136
- LE\_CM\_MSG\_CONN\_UPDATE\_COMPLETE\_IND\_T, 136
  - conn\_hdl, 137
  - interval, 137
  - latency, 137
  - status, 137
  - supervision\_timeout, 137
- LE\_CM\_MSG\_CREATE\_CONNECTION\_CFM\_T
  - BLE CM APIs, 12
- LE\_CM\_MSG\_DATA\_LEN\_CHANGE\_IND\_T, 137
  - conn\_hdl, 138
  - max\_rx\_octets, 138
  - max\_rx\_time, 138
  - max\_tx\_octets, 138
  - max\_tx\_time, 138
- LE\_CM\_MSG\_DIRECT\_ADV\_REPORT\_IND\_T, 138
  - direct\_addr, 139
  - direct\_addr\_type, 139
  - peer\_addr, 139
  - peer\_addr\_type, 139
  - rss\_i, 139
- LE\_CM\_MSG\_DISCONNECT\_COMPLETE\_IND\_T, 139
  - conn\_hdl, 140
  - reason, 140
  - status, 140
- LE\_CM\_MSG\_ENCRYPTION\_CHANGE\_IND\_T, 140
  - conn\_hdl, 140
  - devid, 141
  - enabled, 141
  - status, 141
- LE\_CM\_MSG\_ENCRYPTION\_REFRESH\_IND\_T, 141
  - conn\_hdl, 141
  - devid, 141
  - enabled, 142
  - status, 142
- LE\_CM\_MSG\_ENTER\_ADVERTISING\_CFM\_T
  - BLE CM APIs, 12
- LE\_CM\_MSG\_ENTER\_SCANNING\_CFM\_T
  - BLE CM APIs, 12
- LE\_CM\_MSG\_EXIT\_ADVERTISING\_CFM\_T
  - BLE CM APIs, 12
- LE\_CM\_MSG\_EXIT\_SCANNING\_CFM\_T
  - BLE CM APIs, 12
- LE\_CM\_MSG\_INIT\_COMPLETE\_CFM\_T, 142

- status, [142](#)
- LE\_CM\_MSG\_LTK\_REQ\_IND\_T, [142](#)
  - conn\_hdl, [143](#)
  - devid, [143](#)
  - ediv, [143](#)
  - rand, [143](#)
- LE\_CM\_MSG\_READ\_ADV\_TX\_POWER\_CFM\_T, [143](#)
  - pwr\_level, [144](#)
  - status, [144](#)
- LE\_CM\_MSG\_READ\_BD\_ADDR\_CFM\_T, [144](#)
  - bd\_addr, [144](#)
  - status, [144](#)
- LE\_CM\_MSG\_READ\_CHANNEL\_MAP\_CFM\_T, [145](#)
  - ch\_map, [145](#)
  - conn\_hdl, [145](#)
  - status, [145](#)
- LE\_CM\_MSG\_READ\_RESOLVING\_LIST\_SIZE\_CFM\_T, [145](#)
  - size, [145](#)
  - status, [146](#)
- LE\_CM\_MSG\_READ\_RSSI\_CFM\_T, [146](#)
  - conn\_hdl, [146](#)
  - rssi, [146](#)
  - status, [146](#)
- LE\_CM\_MSG\_READ\_TX\_POWER\_CFM\_T, [147](#)
  - conn\_hdl, [147](#)
  - status, [147](#)
  - tx\_power, [147](#)
- LE\_CM\_MSG\_READ\_WHITE\_LIST\_SIZE\_CFM\_T, [147](#)
  - size, [147](#)
  - status, [148](#)
- LE\_CM\_MSG\_REMOVE\_FROM\_RESOLVING\_LIST\_CFM\_T, [148](#)
  - BLE CM APIs, [12](#)
- LE\_CM\_MSG\_REMOVE\_FROM\_WHITE\_LIST\_CFM\_T, [148](#)
  - BLE CM APIs, [13](#)
- LE\_CM\_MSG\_SET\_ADVERTISING\_DATA\_CFM\_T, [148](#)
  - BLE CM APIs, [13](#)
- LE\_CM\_MSG\_SET\_ADVERTISING\_PARAMS\_CFM\_T, [148](#)
  - BLE CM APIs, [13](#)
- LE\_CM\_MSG\_SET\_CHANNEL\_MAP\_CFM\_T, [148](#)
  - BLE CM APIs, [13](#)
- LE\_CM\_MSG\_SET\_DATA\_LENGTH\_CFM\_T, [148](#)
  - conn\_hdl, [148](#)
  - status, [148](#)
- LE\_CM\_MSG\_SET\_DISCONNECT\_CFM\_T, [148](#)
  - handle, [149](#)
  - status, [149](#)
- LE\_CM\_MSG\_SET\_RANDOM\_ADDRESS\_CFM\_T, [149](#)
  - BLE CM APIs, [13](#)
- LE\_CM\_MSG\_SET\_RPA\_TIMEOUT\_CFM\_T, [149](#)
  - BLE CM APIs, [13](#)
- LE\_CM\_MSG\_SET\_SCAN\_PARAMS\_CFM\_T, [149](#)
  - BLE CM APIs, [13](#)
- LE\_CM\_MSG\_SET\_SCAN\_RSP\_DATA\_CFM\_T, [149](#)
  - BLE CM APIs, [13](#)
- LE\_CM\_MSG\_SIGNAL\_UPDATE\_REQ\_T, [149](#)
  - conn\_hdl, [149](#)
  - identifier, [149](#)
  - interval\_max, [150](#)
  - interval\_min, [150](#)
  - slave\_latency, [150](#)
  - timeout\_multiplier, [150](#)
- LE\_CM\_REQ\_STATUS\_T, [150](#)
  - status, [150](#)
- LE\_CONN\_PARA\_T, [151](#)
  - itv\_max, [151](#)
  - itv\_min, [151](#)
  - latency, [151](#)
  - sv\_timeout, [151](#)
- LE\_GAP\_ADV\_MAX\_SIZE, [151](#)
  - BLE GAP APIs, [24](#)
- LE\_GAP\_ADVERTISING\_PARAM\_T, [152](#)
  - channel\_map, [152](#)
  - filter\_policy, [152](#)
  - interval\_max, [152](#)
  - interval\_min, [152](#)
  - own\_addr\_type, [152](#)
  - peer\_addr, [153](#)
  - peer\_addr\_type, [153](#)
  - type, [153](#)
- LE\_GAP\_CONN\_PARAM\_T, [153](#)
  - interval\_max, [153](#)
  - interval\_min, [153](#)
  - latency, [154](#)
  - supervision\_timeout, [154](#)
- LE\_GAP\_SCAN\_PARAM\_T, [154](#)
  - filter\_policy, [154](#)
  - interval, [154](#)
  - own\_addr\_type, [154](#)
  - type, [155](#)
  - window, [155](#)
- LE\_GATT\_ATTR\_T, [155](#)
  - format, [155](#)
  - handle, [155](#)
  - len, [156](#)
  - maxLen, [156](#)
  - pUuid, [156](#)
  - pVal, [156](#)
  - permit, [156](#)
- LE\_GATT\_CHAR\_PROP\_AUTH, [156](#)
  - BLE GATT APIs, [46](#)
- LE\_GATT\_CHAR\_PROP\_BCAST, [156](#)
  - BLE GATT APIs, [46](#)
- LE\_GATT\_CHAR\_PROP\_EXT\_PROP, [156](#)
  - BLE GATT APIs, [46](#)
- LE\_GATT\_CHAR\_PROP\_IND, [156](#)
  - BLE GATT APIs, [46](#)
- LE\_GATT\_CHAR\_PROP\_NTF, [156](#)
  - BLE GATT APIs, [46](#)
- LE\_GATT\_CHAR\_PROP\_RD, [156](#)
  - BLE GATT APIs, [46](#)
- LE\_GATT\_CHAR\_PROP\_WR\_NO\_RESP, [156](#)
  - BLE GATT APIs, [46](#)

- BLE GATT APIs, 47
- LE\_GATT\_CHAR\_PROP\_WR
  - BLE GATT APIs, 46
- LE\_GATT\_CLIENT\_CFG\_INDICATION
  - BLE GATT APIs, 47
- LE\_GATT\_CLIENT\_CFG\_NOTIFICATION
  - BLE GATT APIs, 47
- LE\_GATT\_EXT\_PROP\_RELIABLE\_WR
  - BLE GATT APIs, 47
- LE\_GATT\_EXT\_PROP\_WR\_AUX
  - BLE GATT APIs, 47
- LE\_GATT\_FLAG\_PREPARE\_WRITE
  - BLE GATT APIs, 47
- LE\_GATT\_FLAG\_WRITE\_CMD
  - BLE GATT APIs, 47
- LE\_GATT\_FLAG\_WRITE\_REQ
  - BLE GATT APIs, 47
- LE\_GATT\_MSG\_ACCESS\_READ\_IND\_T, 156
  - conn\_hdl, 157
  - devid, 157
  - handle, 157
  - offset, 157
- LE\_GATT\_MSG\_ACCESS\_WRITE\_IND\_T, 157
  - conn\_hdl, 157
  - devid, 158
  - flag, 158
  - handle, 158
  - len, 158
  - offset, 158
  - pVal, 158
- LE\_GATT\_MSG\_BASE
  - BLE MSG APIs, 72
- LE\_GATT\_MSG\_CHAR\_DESCRIPTOR\_INFO\_IND\_T, 158
  - conn\_hdl, 159
  - devid, 159
  - format, 159
  - handle, 159
  - uuid, 159
- LE\_GATT\_MSG\_CHARACTERISTIC\_DECL\_INFO\_IND\_T, 159
  - conn\_hdl, 160
  - devid, 160
  - format, 160
  - handle, 160
  - property, 160
  - uuid, 160
  - val\_hdl, 161
- LE\_GATT\_MSG\_CHARACTERISTIC\_VAL\_IND\_T, 161
  - att\_err, 161
  - conn\_hdl, 161
  - devid, 161
  - handle, 162
  - len, 162
  - offset, 162
  - val, 162
- LE\_GATT\_MSG\_CONFIRMATION\_CFM\_T, 162
  - conn\_hdl, 162
- devid, 163
  - handle, 163
- LE\_GATT\_MSG\_EXCHANGE\_MTU\_CFM\_T, 163
  - conn\_hdl, 163
  - current\_rx\_mtu, 163
  - devid, 163
- LE\_GATT\_MSG\_EXCHANGE\_MTU\_IND\_T, 164
  - client\_rx\_mtu, 164
  - conn\_hdl, 164
  - devid, 164
- LE\_GATT\_MSG\_EXECUTE\_WRITE\_RELIABLE\_CFM\_T, 164
  - att\_err, 165
  - conn\_hdl, 165
  - devid, 165
  - err\_hdl, 165
  - status, 165
- LE\_GATT\_MSG\_FIND\_ALL\_CHAR\_DESC\_CFM\_T, 165
  - att\_err, 166
  - conn\_hdl, 166
  - devid, 166
  - handle, 166
  - status, 166
- LE\_GATT\_MSG\_FIND\_ALL\_PRIMARY\_SERVICE\_CFM\_T, 166
  - att\_err, 167
  - conn\_hdl, 167
  - devid, 167
  - handle, 167
  - status, 167
- LE\_GATT\_MSG\_FIND\_CHARACTERISTIC\_CFM\_T, 167
  - att\_err, 168
  - conn\_hdl, 168
  - devid, 168
  - handle, 168
  - status, 168
- LE\_GATT\_MSG\_FIND\_INCLUDED\_SERVICE\_CFM\_T, 168
  - att\_err, 169
  - conn\_hdl, 169
  - devid, 169
  - handle, 169
  - status, 169
- LE\_GATT\_MSG\_FIND\_PRIMARY\_SERVICE\_BY\_UUID\_CFM\_T, 169
  - att\_err, 170
  - conn\_hdl, 170
  - devid, 170
  - handle, 170
  - status, 170
- LE\_GATT\_MSG\_INCLUDE\_SERVICE\_INFO\_IND\_T, 170
  - conn\_hdl, 171
  - devid, 171
  - end\_hdl, 171
  - format, 171

- handle, 171
  - start\_hdl, 171
  - uuid, 172
- LE\_GATT\_MSG\_INDICATE\_IND\_T, 172
  - conn\_hdl, 172
  - devid, 172
  - handle, 172
  - len, 172
  - val, 173
- LE\_GATT\_MSG\_NOTIFY\_CFM\_T, 173
  - conn\_hdl, 173
  - devid, 173
  - handle, 173
  - status, 173
- LE\_GATT\_MSG\_NOTIFY\_IND\_T, 174
  - conn\_hdl, 174
  - devid, 174
  - handle, 174
  - len, 174
  - val, 174
- LE\_GATT\_MSG\_OPERATION\_TIMEOUT\_T, 175
  - att\_op, 175
  - conn\_hdl, 175
  - devid, 175
- LE\_GATT\_MSG\_PREPARE\_WRITE\_RELIABLE\_CFM\_T, 176
  - att\_err, 176
  - conn\_hdl, 176
  - devid, 176
  - handle, 176
  - status, 176
- LE\_GATT\_MSG\_READ\_CHAR\_VAL\_BY\_UUID\_CFM\_T, 177
  - att\_err, 177
  - conn\_hdl, 177
  - devid, 177
  - handle, 177
  - status, 177
- LE\_GATT\_MSG\_READ\_CHARACTERISTIC\_VALUE\_CFM\_T, 178
  - att\_err, 178
  - conn\_hdl, 178
  - devid, 178
  - handle, 178
  - status, 178
- LE\_GATT\_MSG\_READ\_LONG\_CHAR\_VAL\_CFM\_T, 179
  - att\_err, 179
  - conn\_hdl, 179
  - devid, 179
  - handle, 179
  - status, 179
- LE\_GATT\_MSG\_READ\_MULTIPLE\_CHAR\_VAL\_CFM\_T, 180
  - att\_err, 180
  - conn\_hdl, 180
  - devid, 180
  - err\_hdl, 180
  - len, 180
  - status, 180
  - val, 181
- LE\_GATT\_MSG\_SERVICE\_INFO\_IND\_T, 181
  - conn\_hdl, 181
  - devid, 181
  - end\_hdl, 181
  - format, 181
  - start\_hdl, 182
  - uuid, 182
- LE\_GATT\_MSG\_SIGNED\_WRITE\_CFM\_T, 182
  - conn\_hdl, 182
  - devid, 182
  - handle, 182
  - status, 183
- LE\_GATT\_MSG\_WRITE\_CHAR\_VAL\_RELIABLE\_CFM\_T, 183
  - att\_err, 183
  - conn\_hdl, 183
  - devid, 183
  - handle, 183
  - status, 184
- LE\_GATT\_MSG\_WRITE\_CHAR\_VALUE\_CFM\_T, 184
  - att\_err, 184
  - conn\_hdl, 184
  - devid, 184
  - handle, 184
  - status, 185
- LE\_GATT\_MSG\_WRITE\_LONG\_CHAR\_VALUE\_CFM\_T, 185
  - att\_err, 185
  - conn\_hdl, 185
  - devid, 185
  - handle, 185
  - status, 186
- LE\_GATT\_MSG\_WRITE\_NO\_RSP\_CFM\_T, 186
  - conn\_hdl, 186
  - devid, 186
  - handle, 186
  - status, 186
- LE\_GATT\_PERM\_AUTH\_READABLE
  - BLE GATT APIs, 48
- LE\_GATT\_PERM\_AUTH\_WRITABLE
  - BLE GATT APIs, 48
- LE\_GATT\_PERM\_NONE
  - BLE GATT APIs, 48
- LE\_GATT\_PERM\_READ
  - BLE GATT APIs, 48
- LE\_GATT\_PERM\_RELIABLE\_WRITE
  - BLE GATT APIs, 48
- LE\_GATT\_PERM\_WRITE\_CMD
  - BLE GATT APIs, 48
- LE\_GATT\_PERM\_WRITE\_REQ
  - BLE GATT APIs, 48
- LE\_GATT\_PERMIT\_AUTHEN\_READ
  - BLE GATT APIs, 48
- LE\_GATT\_PERMIT\_AUTHEN\_WRITE
  - BLE GATT APIs, 49

- LE\_GATT\_PERMIT\_AUTHOR\_READ
  - BLE GATT APIs, [49](#)
- LE\_GATT\_PERMIT\_AUTHOR\_WRITE
  - BLE GATT APIs, [49](#)
- LE\_GATT\_PERMIT\_ENCRYPT\_READ
  - BLE GATT APIs, [49](#)
- LE\_GATT\_PERMIT\_ENCRYPT\_WRITE
  - BLE GATT APIs, [49](#)
- LE\_GATT\_PERMIT\_READABLE
  - BLE GATT APIs, [49](#)
- LE\_GATT\_PERMIT\_READ
  - BLE GATT APIs, [49](#)
- LE\_GATT\_PERMIT\_SC\_AUTHEN\_READ
  - BLE GATT APIs, [49](#)
- LE\_GATT\_PERMIT\_SC\_AUTHEN\_WRITE
  - BLE GATT APIs, [50](#)
- LE\_GATT\_PERMIT\_WRITABLE
  - BLE GATT APIs, [50](#)
- LE\_GATT\_PERMIT\_WRITE
  - BLE GATT APIs, [50](#)
- LE\_GATT\_SERVICE\_T, [187](#)
  - endHdl, [187](#)
  - pAttr, [187](#)
  - startHdl, [187](#)
  - svc\_id, [187](#)
- LE\_HCI\_MSG\_BASE
  - BLE MSG APIs, [73](#)
- LE\_L2CAP\_MSG\_BASE
  - BLE MSG APIs, [73](#)
- LE\_MAX\_BOND\_COUNT
  - BLE SMP APIs, [84](#)
- LE\_SM\_IO\_CAP\_DISP\_ONLY
  - BLE SMP APIs, [84](#)
- LE\_SM\_IO\_CAP\_DISP\_YES\_NO
  - BLE SMP APIs, [84](#)
- LE\_SM\_IO\_CAP\_KEYBOARD\_DISP
  - BLE SMP APIs, [84](#)
- LE\_SM\_IO\_CAP\_KEYBOARD\_ONLY
  - BLE SMP APIs, [85](#)
- LE\_SM\_IO\_CAP\_NO\_IO
  - BLE SMP APIs, [85](#)
- LE\_SM\_PAIR\_MITM\_NO
  - BLE SMP APIs, [85](#)
- LE\_SM\_PAIR\_MITM\_YES
  - BLE SMP APIs, [85](#)
- LE\_SM\_PAIR\_OOB\_NO
  - BLE SMP APIs, [85](#)
- LE\_SM\_PAIR\_OOB\_YES
  - BLE SMP APIs, [85](#)
- LE\_SM\_PAIR\_SC\_NO
  - BLE SMP APIs, [85](#)
- LE\_SM\_PAIR\_SC\_YES
  - BLE SMP APIs, [85](#)
- LE\_SMP\_MSG\_BASE
  - BLE MSG APIs, [73](#)
- LE\_SMP\_MSG\_ENCRYPTION\_CHANGE\_IND\_T, [188](#)
  - conn\_hdl, [188](#)
  - enable, [188](#)
- LE\_SMP\_MSG\_ENCRYPTION\_REFRESH\_IND\_T, [188](#)
  - conn\_hdl, [188](#)
  - status, [188](#)
- LE\_SMP\_MSG\_OOB\_DATA\_REQUEST\_IND\_T, [189](#)
  - conn\_hdl, [189](#)
- LE\_SMP\_MSG\_PAIRING\_ACTION\_IND\_T, [189](#)
  - action, [189](#)
  - conn\_hdl, [189](#)
  - lost\_bond, [190](#)
  - sc, [190](#)
- LE\_SMP\_MSG\_PAIRING\_COMPLETE\_IND\_T, [190](#)
  - authenticated, [190](#)
  - bonded, [190](#)
  - conn\_hdl, [190](#)
  - peer\_id\_addr, [191](#)
  - sc, [191](#)
  - status, [191](#)
- LE\_SMP\_MSG\_PASSKEY\_DISPLAY\_IND\_T, [191](#)
  - conn\_hdl, [191](#)
  - passkey, [191](#)
- LE\_SMP\_MSG\_PASSKEY\_INPUT\_IND\_T, [192](#)
  - conn\_hdl, [192](#)
- LE\_SMP\_MSG\_SC\_OOB\_DATA\_REQUEST\_IND\_T, [192](#)
  - conn\_hdl, [192](#)
- LE\_SMP\_MSG\_SLAVE\_SECURITY\_REQUEST\_IND\_T, [193](#)
  - bondable, [193](#)
  - conn\_hdl, [193](#)
  - keypress, [193](#)
  - mitm, [193](#)
  - sc, [193](#)
- LE\_SMP\_MSG\_USER\_CONFIRM\_IND\_T, [194](#)
  - confirm\_num, [194](#)
  - conn\_hdl, [194](#)
- LE\_SMP\_SC\_OOB\_DATA\_T, [194](#)
  - confirm, [194](#)
  - rand, [194](#)
- LE\_SYS\_MSG\_BASE
  - BLE MSG APIs, [73](#)
- LE\_SYS\_MSG\_BUF\_OVERFLOW\_T, [195](#)
  - conn\_hdl, [195](#)
- latency
  - LE\_CM\_MSG\_CONN\_PARA\_REQ\_T, [136](#)
  - LE\_CM\_MSG\_CONN\_UPDATE\_COMPLETE\_IND\_T, [137](#)
  - LE\_CONN\_PARA\_T, [151](#)
  - LE\_GAP\_CONN\_PARAM\_T, [154](#)
- latest\_beacon\_rx\_time
  - auto\_conn\_info\_t, [128](#)
  - mw\_wifi\_auto\_connect\_ap\_info\_t, [197](#)
  - wifi\_auto\_connect\_info\_f, [206](#)
- LeCancelAllMessage
  - BLE MSG APIs, [76](#)
- LeCancelAllSubMessage
  - BLE MSG APIs, [77](#)
- LeCancelFirstMessage



- BLE MSG APIs, [77](#)
- LeCancelFirstSubMessage
  - BLE MSG APIs, [77](#)
- LeCmInit
  - BLE CM APIs, [15](#)
- LeGapAddToResolvingList
  - BLE GAP APIs, [25](#)
- LeGapAddToWhiteList
  - BLE GAP APIs, [25](#)
- LeGapAdvertisingEnable
  - BLE GAP APIs, [25](#)
- LeGapCentralConnectReq
  - BLE GAP APIs, [26](#)
- LeGapCentralSetDataChannel
  - BLE GAP APIs, [26](#)
- LeGapClearResolvingList
  - BLE GAP APIs, [27](#)
- LeGapClearWhiteList
  - BLE GAP APIs, [27](#)
- LeGapConnParaRequestRsp
  - BLE GAP APIs, [27](#)
- LeGapConnUpdateRequest
  - BLE GAP APIs, [28](#)
- LeGapConnUpdateResponse
  - BLE GAP APIs, [28](#)
- LeGapConnectCancelReq
  - BLE GAP APIs, [27](#)
- LeGapDisconnectReq
  - BLE GAP APIs, [29](#)
- LeGapGenRandAddr
  - BLE GAP APIs, [29](#)
- LeGapGetBtAddr
  - BLE GAP APIs, [29](#)
- LeGapReadAdvChannelTxPower
  - BLE GAP APIs, [29](#)
- LeGapReadChannelMap
  - BLE GAP APIs, [30](#)
- LeGapReadResolvingListSize
  - BLE GAP APIs, [30](#)
- LeGapReadRssi
  - BLE GAP APIs, [30](#)
- LeGapReadTxPower
  - BLE GAP APIs, [31](#)
- LeGapReadWhiteListSize
  - BLE GAP APIs, [31](#)
- LeGapRemoveFromWhiteList
  - BLE GAP APIs, [31](#)
- LeGapScanningReq
  - BLE GAP APIs, [32](#)
- LeGapSetAdvData
  - BLE GAP APIs, [32](#)
- LeGapSetAdvParameter
  - BLE GAP APIs, [33](#)
- LeGapSetConnParameter
  - BLE GAP APIs, [33](#)
- LeGapSetDataChannelPduLen
  - BLE GAP APIs, [33](#)
- LeGapSetRandAddr
  - BLE GAP APIs, [34](#)
- LeGapSetRpaTimeout
  - BLE GAP APIs, [34](#)
- LeGapSetStaticAddr
  - BLE GAP APIs, [35](#)
- LeGattAccessReadRsp
  - BLE GATT APIs, [52](#)
- LeGattAccessWriteRsp
  - BLE GATT APIs, [52](#)
- LeGattChangeAttrVal
  - BLE GATT APIs, [53](#)
- LeGattCharValConfirmation
  - BLE GATT APIs, [53](#)
- LeGattCharValIndicate
  - BLE GATT APIs, [54](#)
- LeGattCharValNotify
  - BLE GATT APIs, [54](#)
- LeGattExchangeMtuReq
  - BLE GATT APIs, [55](#)
- LeGattExchangeMtuRsp
  - BLE GATT APIs, [55](#)
- LeGattExecuteWriteCharValReliable
  - BLE GATT APIs, [55](#)
- LeGattFindAllCharDescriptor
  - BLE GATT APIs, [56](#)
- LeGattFindAllCharacteristic
  - BLE GATT APIs, [56](#)
- LeGattFindAllPrimaryService
  - BLE GATT APIs, [57](#)
- LeGattFindCharacteristicByUuid
  - BLE GATT APIs, [57](#)
- LeGattFindIncludedService
  - BLE GATT APIs, [58](#)
- LeGattFindPrimaryServiceByUuid
  - BLE GATT APIs, [58](#)
- LeGattGetAttrHandle
  - BLE GATT APIs, [58](#)
- LeGattGetAttrVal
  - BLE GATT APIs, [59](#)
- LeGattGetAttrValLen
  - BLE GATT APIs, [59](#)
- LeGattGetAttrValMaxLen
  - BLE GATT APIs, [61](#)
- LeGattInit
  - BLE GATT APIs, [61](#)
- LeGattModifyAttrVal
  - BLE GATT APIs, [62](#)
- LeGattPrepareWriteCharValReliable
  - BLE GATT APIs, [62](#)
- LeGattReadCharValByUuid
  - BLE GATT APIs, [63](#)
- LeGattReadCharValue
  - BLE GATT APIs, [63](#)
- LeGattReadLongCharVal
  - BLE GATT APIs, [64](#)
- LeGattReadMultipleCharVal
  - BLE GATT APIs, [64](#)
- LeGattRegisterIncludeService



- BLE GATT APIs, [64](#)
- LeGattRegisterService
  - BLE GATT APIs, [65](#)
- LeGattSignedWriteNoRsp
  - BLE GATT APIs, [65](#)
- LeGattStopCurrentProcedure
  - BLE GATT APIs, [66](#)
- LeGattWriteCharVal
  - BLE GATT APIs, [66](#)
- LeGattWriteCharValReliable
  - BLE GATT APIs, [67](#)
- LeGattWriteLongCharVal
  - BLE GATT APIs, [67](#)
- LeGattWriteNoRsp
  - BLE GATT APIs, [68](#)
- LeGetSubMsgId
  - BLE MSG APIs, [78](#)
- LeHostCreateTask
  - BLE MSG APIs, [78](#)
- LeHostMessageLoop
  - BLE MSG APIs, [79](#)
- LeSendMessage
  - BLE MSG APIs, [79](#)
- LeSendMessageAfter
  - BLE MSG APIs, [79](#)
- LeSendMessageUnlock
  - BLE MSG APIs, [80](#)
- LeSendSubMessage
  - BLE MSG APIs, [80](#)
- LeSendSubMessageAfter
  - BLE MSG APIs, [81](#)
- LeSendSubMessageUnlock
  - BLE MSG APIs, [81](#)
- LeSetScanParameter
  - BLE GAP APIs, [35](#)
- LeSetScanRspData
  - BLE GAP APIs, [35](#)
- LeSmpInit
  - BLE SMP APIs, [87](#)
- LeSmpOobAuthDataRsp
  - BLE SMP APIs, [87](#)
- LeSmpOobPresent
  - BLE SMP APIs, [87](#)
- LeSmpPasskeyInput
  - BLE SMP APIs, [88](#)
- LeSmpScOobComputeConfirmVal
  - BLE SMP APIs, [88](#)
- LeSmpScOobDataRsp
  - BLE SMP APIs, [88](#)
- LeSmpSecurityReq
  - BLE SMP APIs, [89](#)
- LeSmpSecurityRsp
  - BLE SMP APIs, [89](#)
- LeSmpSetDefaultConfig
  - BLE SMP APIs, [90](#)
- LeSmpUserConfirmRsp
  - BLE SMP APIs, [90](#)
- len
  - LE\_CM\_MSG\_ADVERTISE\_REPORT\_IND\_↵  
T, [135](#)
  - LE\_GATT\_ATTR\_T, [156](#)
  - LE\_GATT\_MSG\_ACCESS\_WRITE\_IND\_T, [158](#)
  - LE\_GATT\_MSG\_CHARACTERISTIC\_VAL\_IND\_↵  
T, [162](#)
  - LE\_GATT\_MSG\_INDICATE\_IND\_T, [172](#)
  - LE\_GATT\_MSG\_NOTIFY\_IND\_T, [174](#)
  - LE\_GATT\_MSG\_READ\_MULTIPLE\_CHAR\_VAL\_↵  
L\_CFM\_T, [180](#)
- length
  - event\_msg\_t, [132](#)
- lost\_bond
  - LE\_SMP\_MSG\_PAIRING\_ACTION\_IND\_T, [190](#)
- MESSAGE\_ALLOCATE
  - BLE MSG APIs, [73](#)
- MESSAGE\_BULID
  - BLE MSG APIs, [73](#)
- MESSAGE\_DATA\_BULID
  - BLE MSG APIs, [73](#)
- MESSAGE\_OFFSET
  - BLE MSG APIs, [74](#)
- MESSAGEID
  - BLE MSG APIs, [74](#)
- MESSAGE
  - BLE MSG APIs, [74](#)
- MSGLOCK
  - BLE MSG APIs, [75](#)
- MSGSUBID
  - BLE MSG APIs, [75](#)
- MSGTIMER
  - BLE MSG APIs, [75](#)
- magic
  - wifi\_init\_config\_t, [214](#)
- max
  - wifi\_active\_scan\_time\_t, [202](#)
- max\_connection
  - wifi\_ap\_config\_t, [204](#)
- max\_rx\_octets
  - LE\_CM\_MSG\_DATA\_LEN\_CHANGE\_IND\_T, [138](#)
- max\_rx\_time
  - LE\_CM\_MSG\_DATA\_LEN\_CHANGE\_IND\_T, [138](#)
- max\_save\_num
  - auto\_connect\_cfg\_t, [130](#)
  - MwFimAutoConnectCFG\_t, [198](#)
- max\_tx\_octets
  - LE\_CM\_MSG\_DATA\_LEN\_CHANGE\_IND\_T, [138](#)
- max\_tx\_time
  - LE\_CM\_MSG\_DATA\_LEN\_CHANGE\_IND\_T, [138](#)
- maxLen
  - LE\_GATT\_ATTR\_T, [156](#)
- min
  - wifi\_active\_scan\_time\_t, [202](#)
- mitm
  - LE\_SMP\_MSG\_SLAVE\_SECURITY\_REQUEST\_↵  
T\_IND\_T, [193](#)
- MsgData
  - BLE MSG APIs, [75](#)

- MsgLock
  - BLE MSG APIs, 75
- mw\_wifi\_auto\_connect\_ap\_info\_t, 195
  - ap\_channel, 196
  - beacon\_interval, 196
  - bssid, 196
  - capabilities, 196
  - dtim\_prod, 196
  - fast\_connect, 196
  - free\_ocpy, 196
  - hid\_ssid, 196
  - latest\_beacon\_rx\_time, 197
  - passphrase, 197
  - psk, 197
  - rsn\_ie, 197
  - rsi, 197
  - ssid, 197
  - supported\_rates, 197
  - wpa\_data, 197
  - wpa\_ie, 198
- MwFimAutoConnectCFG\_t, 198
  - flag, 198
  - front, 198
  - max\_save\_num, 198
  - rear, 198
  - targetIdx, 199
- num
  - wifi\_scan\_list\_t, 218
- number
  - wifi\_event\_sta\_scan\_done\_t, 212
- offset
  - LE\_GATT\_MSG\_ACCESS\_READ\_IND\_T, 157
  - LE\_GATT\_MSG\_ACCESS\_WRITE\_IND\_T, 158
  - LE\_GATT\_MSG\_CHARACTERISTIC\_VAL\_IND\_T, 162
- own\_addr\_type
  - LE\_GAP\_ADVERTISING\_PARAM\_T, 152
  - LE\_GAP\_SCAN\_PARAM\_T, 154
- pAttr
  - LE\_GATT\_SERVICE\_T, 187
- pFCInfo
  - auto\_connect\_cfg\_t, 130
- pParam
  - T\_RfEvt, 200
- PRIMARY\_SERVICE\_DECL\_UUID128
  - BLE GATT APIs, 50
- PRIMARY\_SERVICE\_DECL\_UUID16
  - BLE GATT APIs, 50
- pUuid
  - LE\_GATT\_ATTR\_T, 156
- pVal
  - LE\_GATT\_ATTR\_T, 156
  - LE\_GATT\_MSG\_ACCESS\_WRITE\_IND\_T, 158
- pairwise\_cipher
  - wifi\_scan\_info\_t, 217
- param
  - event\_msg\_t, 132
- passive
  - wifi\_scan\_time\_t, 218
- passkey
  - LE\_SMP\_MSG\_PASSKEY\_DISPLAY\_IND\_T, 191
- passphrase
  - auto\_conn\_info\_t, 128
  - mw\_wifi\_auto\_connect\_ap\_info\_t, 197
  - wifi\_auto\_connect\_info\_f, 206
- password
  - wifi\_ap\_config\_t, 204
  - wifi\_sta\_config\_t, 219
- password\_length
  - wifi\_ap\_config\_t, 204
  - wifi\_sta\_config\_t, 220
- peer\_addr
  - LE\_CM\_CONNECTION\_COMPLETE\_IND\_T, 133
  - LE\_CM\_MSG\_DIRECT\_ADV\_REPORT\_IND\_T, 139
  - LE\_GAP\_ADVERTISING\_PARAM\_T, 153
- peer\_addr\_type
  - LE\_CM\_CONNECTION\_COMPLETE\_IND\_T, 134
  - LE\_CM\_MSG\_DIRECT\_ADV\_REPORT\_IND\_T, 139
  - LE\_GAP\_ADVERTISING\_PARAM\_T, 153
- peer\_id\_addr
  - LE\_SMP\_MSG\_PAIRING\_COMPLETE\_IND\_T, 191
- permit
  - LE\_GATT\_ATTR\_T, 156
- property
  - LE\_GATT\_MSG\_CHARACTERISTIC\_DECL\_IND\_T, 160
- psk
  - auto\_conn\_info\_t, 129
  - mw\_wifi\_auto\_connect\_ap\_info\_t, 197
  - wifi\_auto\_connect\_info\_f, 206
- pwr\_level
  - LE\_CM\_MSG\_READ\_ADV\_TX\_POWER\_CFM\_T, 144
- rand
  - LE\_CM\_MSG\_LTK\_REQ\_IND\_T, 143
  - LE\_SMP\_SC\_OOB\_DATA\_T, 194
- rear
  - auto\_connect\_cfg\_t, 130
  - MwFimAutoConnectCFG\_t, 198
- reason
  - LE\_CM\_MSG\_DISCONNECT\_COMPLETE\_IND\_T, 140
  - wifi\_event\_sta\_disconnected\_t, 211
- retryCount
  - auto\_connect\_cfg\_t, 131
- role
  - LE\_CM\_CONNECTION\_COMPLETE\_IND\_T, 134
- rsn\_ie
  - auto\_conn\_info\_t, 129
  - mw\_wifi\_auto\_connect\_ap\_info\_t, 197
  - wifi\_auto\_connect\_info\_f, 207

- auto\_conn\_info\_t, 129
    - LE\_CM\_MSG\_ADVERTISE\_REPORT\_IND\_↔  
T, 135
    - LE\_CM\_MSG\_DIRECT\_ADV\_REPORT\_IND\_T,  
139
    - LE\_CM\_MSG\_READ\_RSSI\_CFM\_T, 146
    - mw\_wifi\_auto\_connect\_ap\_info\_t, 197
    - wifi\_auto\_connect\_info\_f, 207
    - wifi\_fast\_scan\_threshold\_t, 213
    - wifi\_scan\_info\_t, 217
  - SECONDARY\_SERVICE\_DECL\_UUID128
    - BLE GATT APIs, 50
  - SECONDARY\_SERVICE\_DECL\_UUID16
    - BLE GATT APIs, 50
  - saArgv
    - T\_RfCmd, 199
  - sc
    - LE\_SMP\_MSG\_PAIRING\_ACTION\_IND\_T, 190
    - LE\_SMP\_MSG\_PAIRING\_COMPLETE\_IND\_T,  
191
    - LE\_SMP\_MSG\_SLAVE\_SECURITY\_REQUES↔  
T\_IND\_T, 193
  - scan\_done
    - wifi\_event\_info\_t, 209
  - scan\_id
    - wifi\_event\_sta\_scan\_done\_t, 212
  - scan\_method
    - wifi\_sta\_config\_t, 220
  - scan\_time
    - wifi\_scan\_config\_t, 215
  - scan\_type
    - wifi\_scan\_config\_t, 215
  - show\_hidden
    - wifi\_scan\_config\_t, 215
  - size
    - LE\_CM\_MSG\_READ\_RESOLVING\_LIST\_SIZE↔  
\_CFM\_T, 145
    - LE\_CM\_MSG\_READ\_WHITE\_LIST\_SIZE\_CFM↔  
\_T, 147
  - slave\_latency
    - LE\_CM\_MSG\_SIGNAL\_UPDATE\_REQ\_T, 150
  - sort\_method
    - wifi\_sta\_config\_t, 220
  - ssid
    - auto\_conn\_info\_t, 129
    - mw\_wifi\_auto\_connect\_ap\_info\_t, 197
    - wifi\_ap\_config\_t, 204
    - wifi\_auto\_connect\_info\_f, 207
    - wifi\_event\_sta\_connected\_t, 210
    - wifi\_event\_sta\_disconnected\_t, 211
    - wifi\_scan\_config\_t, 215
    - wifi\_scan\_info\_t, 217
    - wifi\_sta\_config\_t, 220
  - ssid\_hidden
    - wifi\_ap\_config\_t, 204
  - ssid\_len
    - wifi\_event\_sta\_connected\_t, 210
    - wifi\_event\_sta\_disconnected\_t, 211
  - ssid\_length
    - wifi\_ap\_config\_t, 204
    - wifi\_scan\_info\_t, 217
    - wifi\_sta\_config\_t, 220
  - sta\_config
    - wifi\_config\_t, 208
  - start\_hdl
    - LE\_GATT\_MSG\_INCLUDE\_SERVICE\_INFO\_I↔  
ND\_T, 171
    - LE\_GATT\_MSG\_SERVICE\_INFO\_IND\_T, 182
  - startHdl
    - LE\_GATT\_SERVICE\_T, 187
  - status
    - LE\_CM\_CONNECTION\_COMPLETE\_IND\_T, 134
    - LE\_CM\_MSG\_CONN\_UPDATE\_COMPLETE\_I↔  
ND\_T, 137
    - LE\_CM\_MSG\_DISCONNECT\_COMPLETE\_IN↔  
D\_T, 140
    - LE\_CM\_MSG\_ENCRYPTION\_CHANGE\_IND\_T,  
141
    - LE\_CM\_MSG\_ENCRYPTION\_REFRESH\_IND\_T,  
142
    - LE\_CM\_MSG\_INIT\_COMPLETE\_CFM\_T, 142
    - LE\_CM\_MSG\_READ\_ADV\_TX\_POWER\_CFM↔  
\_T, 144
    - LE\_CM\_MSG\_READ\_BD\_ADDR\_CFM\_T, 144
    - LE\_CM\_MSG\_READ\_CHANNEL\_MAP\_CFM\_T,  
145
    - LE\_CM\_MSG\_READ\_RESOLVING\_LIST\_SIZE↔  
\_CFM\_T, 146
    - LE\_CM\_MSG\_READ\_RSSI\_CFM\_T, 146
    - LE\_CM\_MSG\_READ\_TX\_POWER\_CFM\_T, 147
    - LE\_CM\_MSG\_READ\_WHITE\_LIST\_SIZE\_CFM↔  
\_T, 148
    - LE\_CM\_MSG\_SET\_DATA\_LENGTH\_CFM\_T,  
148
    - LE\_CM\_MSG\_SET\_DISCONNECT\_CFM\_T, 149
    - LE\_CM\_REQ\_STATUS\_T, 150
    - LE\_GATT\_MSG\_EXECUTE\_WRITE\_RELIABL↔  
E\_CFM\_T, 165
    - LE\_GATT\_MSG\_FIND\_ALL\_CHAR\_DESC\_CF↔  
M\_T, 166
    - LE\_GATT\_MSG\_FIND\_ALL\_PRIMARY\_SERVI↔  
CE\_CFM\_T, 167
    - LE\_GATT\_MSG\_FIND\_CHARACTERISTIC\_CF↔  
M\_T, 168
    - LE\_GATT\_MSG\_FIND\_INCLUDED\_SERVICE\_↔  
CFM\_T, 169
    - LE\_GATT\_MSG\_FIND\_PRIMARY\_SERVICE\_B↔  
Y\_UUID\_CFM\_T, 170
    - LE\_GATT\_MSG\_NOTIFY\_CFM\_T, 173
    - LE\_GATT\_MSG\_PREPARE\_WRITE\_RELIABL↔  
E\_CFM\_T, 176
    - LE\_GATT\_MSG\_READ\_CHAR\_VAL\_BY\_UUID↔  
\_CFM\_T, 177
    - LE\_GATT\_MSG\_READ\_CHARACTERISTIC\_V↔  
ALUE\_CFM\_T, 178

- LE\_GATT\_MSG\_READ\_LONG\_CHAR\_VAL\_C↔  
FM\_T, 179
- LE\_GATT\_MSG\_READ\_MULTIPLE\_CHAR\_VAL\_C↔  
L\_CFM\_T, 180
- LE\_GATT\_MSG\_SIGNED\_WRITE\_CFM\_T, 183
- LE\_GATT\_MSG\_WRITE\_CHAR\_VAL\_RELIAB↔  
LE\_CFM\_T, 184
- LE\_GATT\_MSG\_WRITE\_CHAR\_VALUE\_CFM↔  
\_T, 185
- LE\_GATT\_MSG\_WRITE\_LONG\_CHAR\_VALU↔  
E\_CFM\_T, 186
- LE\_GATT\_MSG\_WRITE\_NO\_RSP\_CFM\_T, 186
- LE\_SMP\_MSG\_ENCRYPTION\_REFRESH\_IND↔  
\_T, 188
- LE\_SMP\_MSG\_PAIRING\_COMPLETE\_IND\_T,  
191
- wifi\_event\_sta\_scan\_done\_t, 212
- supervision\_timeout
  - LE\_CM\_MSG\_CONN\_UPDATE\_COMPLETE\_I↔  
ND\_T, 137
  - LE\_GAP\_CONN\_PARAM\_T, 154
- supervision\_timeout
  - LE\_CM\_CONNECTION\_COMPLETE\_IND\_T, 134
- supported\_rates
  - auto\_conn\_info\_t, 129
  - mw\_wifi\_auto\_connect\_ap\_info\_t, 197
  - wifi\_auto\_connect\_info\_f, 207
- sv\_timeout
  - LE\_CONN\_PARA\_T, 151
- sv\_tmo
  - LE\_CM\_MSG\_CONN\_PARA\_REQ\_T, 136
- svc\_id
  - LE\_GATT\_SERVICE\_T, 187
- T\_HOUR
  - BLE MSG APIs, 74
- T\_MIN
  - BLE MSG APIs, 74
- T\_RfCmd, 199
  - iArgc, 199
  - saArgv, 199
  - u32Type, 199
- T\_RfEvt, 199
  - pParam, 200
  - u16RfMode, 200
  - u16RxCnt, 200
  - u16RxCrcOkCnt, 200
  - u32Freq, 200
  - u32Mode, 201
  - u32RfChannel, 201
  - u32Type, 201
  - u8Freq, 201
  - u8IpcEnable, 201
  - u8Len, 201
  - u8Pkt, 201
  - u8Reserved, 201
  - u8Status, 202
  - u8Unicast, 202
- T\_SEC
  - BLE MSG APIs, 74
- TASKHANDLER
  - BLE MSG APIs, 75
- TASKPACK
  - BLE MSG APIs, 76
- TASK
  - BLE MSG APIs, 75
- targetIdx
  - auto\_connect\_cfg\_t, 131
  - MwFimAutoConnectCFG\_t, 199
- Task
  - BLE MSG APIs, 75
- threshold
  - wifi\_sta\_config\_t, 220
- timeout\_multiplier
  - LE\_CM\_MSG\_SIGNAL\_UPDATE\_REQ\_T, 150
- tx\_power
  - LE\_CM\_MSG\_READ\_TX\_POWER\_CFM\_T, 147
- type
  - LE\_BT\_ADDR\_T, 132
  - LE\_GAP\_ADVERTISING\_PARAM\_T, 153
  - LE\_GAP\_SCAN\_PARAM\_T, 155
- u16RfMode
  - T\_RfEvt, 200
- u16RxCnt
  - T\_RfEvt, 200
- u16RxCrcOkCnt
  - T\_RfEvt, 200
- u32Freq
  - T\_RfEvt, 200
- u32Mode
  - T\_RfEvt, 201
- u32RfChannel
  - T\_RfEvt, 201
- u32Type
  - T\_RfCmd, 199
  - T\_RfEvt, 201
- u8Freq
  - T\_RfEvt, 201
- u8IpcEnable
  - T\_RfEvt, 201
- u8Len
  - T\_RfEvt, 201
- u8Pkt
  - T\_RfEvt, 201
- u8Reserved
  - T\_RfEvt, 201
- u8Status
  - T\_RfEvt, 202
- u8Unicast
  - T\_RfEvt, 202
- uFCAPNum
  - auto\_connect\_cfg\_t, 131
- uuid
  - LE\_GATT\_MSG\_CHAR\_DESCRIPTOR\_INFO↔  
IND\_T, 159
  - LE\_GATT\_MSG\_CHARACTERISTIC\_DECL\_IN↔  
FO\_IND\_T, 160

- LE\_GATT\_MSG\_INCLUDE\_SERVICE\_INFO\_IND\_T, 172
- LE\_GATT\_MSG\_SERVICE\_INFO\_IND\_T, 182
- val
  - LE\_GATT\_MSG\_CHARACTERISTIC\_VAL\_IND\_T, 162
  - LE\_GATT\_MSG\_INDICATE\_IND\_T, 173
  - LE\_GATT\_MSG\_NOTIFY\_IND\_T, 174
  - LE\_GATT\_MSG\_READ\_MULTIPLE\_CHAR\_VAL\_CFM\_T, 181
- val\_hdl
  - LE\_GATT\_MSG\_CHARACTERISTIC\_DECL\_IND\_T, 161
- WIFI APIs, 91
  - WIFI\_BEACON\_INTERVAL\_LENGTH, 92
  - WIFI\_CAPABILITY\_INFO\_LENGTH, 92
  - WIFI\_LENGTH\_802\_11, 92
  - WIFI\_LENGTH\_PASSPHRASE, 92
  - WIFI\_MAC\_ADDRESS\_LENGTH, 93
  - WIFI\_MAX\_LENGTH\_OF\_SSID, 93
  - WIFI\_MAX\_SCAN\_AP\_NUM, 93
  - WIFI\_MAX\_SUPPORTED\_RATES, 93
  - wifi\_event\_notify\_cb\_t, 93
  - wifi\_event\_process\_handler, 94
  - wifi\_install\_default\_event\_handlers, 94
  - wifi\_register\_event\_handler, 94
- WIFI Common APIs, 96
  - wifi\_event\_cb\_t, 96
  - wifi\_event\_loop\_init, 97
  - wifi\_event\_loop\_send, 98
  - wifi\_event\_loop\_set\_cb, 98
  - wifi\_event\_process\_handler, 99
- WIFI STA APIs, 100
  - wifi\_auto\_connect\_del\_ap\_info, 102
  - wifi\_auto\_connect\_get\_ap\_info, 103
  - wifi\_auto\_connect\_get\_ap\_num, 103
  - wifi\_auto\_connect\_get\_mode, 103
  - wifi\_auto\_connect\_init, 104
  - wifi\_auto\_connect\_set\_ap\_num, 104
  - wifi\_auto\_connect\_set\_mode, 104
  - wifi\_auto\_connect\_start, 106
  - wifi\_config\_get\_bandwidth, 106
  - wifi\_config\_get\_bssid, 107
  - wifi\_config\_get\_channel, 107
  - wifi\_config\_get\_mac\_address, 108
  - wifi\_config\_get\_ssid, 108
  - wifi\_config\_set\_bandwidth, 108
  - wifi\_config\_set\_bssid, 109
  - wifi\_config\_set\_channel, 109
  - wifi\_config\_set\_mac\_address, 111
  - wifi\_config\_set\_ssid, 111
  - wifi\_connection\_connect, 112
  - wifi\_connection\_disconnect\_ap, 112
  - wifi\_connection\_disconnect\_sta, 112
  - wifi\_connection\_get\_rssi, 113
  - wifi\_connection\_register\_event\_handler, 113
  - wifi\_connection\_unregister\_event\_handler, 114
  - wifi\_deinit, 114
  - wifi\_event\_handler\_t, 101
  - wifi\_fast\_connect\_get\_mode, 115
  - wifi\_fast\_connect\_set\_mode, 115
  - wifi\_fast\_connect\_start, 115
  - wifi\_get\_config, 116
  - wifi\_get\_fast\_conn\_mode, 116
  - wifi\_init, 116
  - wifi\_init\_complete\_cb\_t, 102
  - wifi\_result\_t, 102
  - wifi\_scan\_get\_ap\_list, 117
  - wifi\_scan\_get\_ap\_num, 117
  - wifi\_scan\_get\_ap\_records, 118
  - wifi\_scan\_scan\_stop, 118
  - wifi\_scan\_start, 118
  - wifi\_set\_config, 119
  - wifi\_sta\_get\_ap\_info, 119
  - wifi\_start, 120
  - wifi\_stop, 120
  - WIFI\_BEACON\_INTERVAL\_LENGTH
    - WIFI APIs, 92
  - WIFI\_CAPABILITY\_INFO\_LENGTH
    - WIFI APIs, 92
  - WIFI\_LENGTH\_802\_11
    - WIFI APIs, 92
  - WIFI\_LENGTH\_PASSPHRASE
    - WIFI APIs, 92
  - WIFI\_MAC\_ADDRESS\_LENGTH
    - WIFI APIs, 93
  - WIFI\_MAX\_LENGTH\_OF\_SSID
    - WIFI APIs, 93
  - WIFI\_MAX\_SCAN\_AP\_NUM
    - WIFI APIs, 93
  - WIFI\_MAX\_SUPPORTED\_RATES
    - WIFI APIs, 93
  - wifi\_active\_scan\_time\_t, 202
    - max, 202
    - min, 202
  - wifi\_ap\_config\_t, 203
    - auth\_mode, 203
    - beacon\_interval, 203
    - channel, 203
    - encrypt\_type, 204
    - max\_connection, 204
    - password, 204
    - password\_length, 204
    - ssid, 204
    - ssid\_hidden, 204
    - ssid\_length, 204
  - wifi\_auth\_mode\_t
    - Enumeration, 121
  - wifi\_auto\_connect\_del\_ap\_info
    - WIFI STA APIs, 102
  - wifi\_auto\_connect\_get\_ap\_info
    - WIFI STA APIs, 103
  - wifi\_auto\_connect\_get\_ap\_num
    - WIFI STA APIs, 103
  - wifi\_auto\_connect\_get\_mode

- WIFI STA APIs, 103
- wifi\_auto\_connect\_info\_f, 205
  - ap\_channel, 205
  - beacon\_interval, 205
  - bssid, 205
  - capabilities, 206
  - dtim\_prod, 206
  - fast\_connect, 206
  - free\_ocpy, 206
  - hid\_ssid, 206
  - latest\_beacon\_rx\_time, 206
  - passphrase, 206
  - psk, 206
  - rsn\_ie, 207
  - rss, 207
  - ssid, 207
  - supported\_rates, 207
  - wpa\_data, 207
  - wpa\_ie, 207
- wifi\_auto\_connect\_init
  - WIFI STA APIs, 104
- wifi\_auto\_connect\_set\_ap\_num
  - WIFI STA APIs, 104
- wifi\_auto\_connect\_set\_mode
  - WIFI STA APIs, 104
- wifi\_auto\_connect\_start
  - WIFI STA APIs, 106
- wifi\_bandwidth\_t
  - Enumeration, 122
- wifi\_cipher\_type\_t
  - Enumeration, 122
- wifi\_config\_get\_bandwidth
  - WIFI STA APIs, 106
- wifi\_config\_get\_bssid
  - WIFI STA APIs, 107
- wifi\_config\_get\_channel
  - WIFI STA APIs, 107
- wifi\_config\_get\_mac\_address
  - WIFI STA APIs, 108
- wifi\_config\_get\_ssid
  - WIFI STA APIs, 108
- wifi\_config\_set\_bandwidth
  - WIFI STA APIs, 108
- wifi\_config\_set\_bssid
  - WIFI STA APIs, 109
- wifi\_config\_set\_channel
  - WIFI STA APIs, 109
- wifi\_config\_set\_mac\_address
  - WIFI STA APIs, 111
- wifi\_config\_set\_ssid
  - WIFI STA APIs, 111
- wifi\_config\_t, 207
  - ap\_config, 208
  - sta\_config, 208
- wifi\_connection\_connect
  - WIFI STA APIs, 112
- wifi\_connection\_disconnect\_ap
  - WIFI STA APIs, 112
- wifi\_connection\_disconnect\_sta
  - WIFI STA APIs, 112
- wifi\_connection\_get\_rssi
  - WIFI STA APIs, 113
- wifi\_connection\_register\_event\_handler
  - WIFI STA APIs, 113
- wifi\_connection\_unregister\_event\_handler
  - WIFI STA APIs, 114
- wifi\_deinit
  - WIFI STA APIs, 114
- wifi\_event\_cb\_t
  - WIFI Common APIs, 96
- wifi\_event\_handler\_t
  - WIFI STA APIs, 101
- wifi\_event\_info\_t, 208
  - connected, 209
  - disconnected, 209
  - got\_ip, 209
  - scan\_done, 209
- wifi\_event\_loop\_init
  - WIFI Common APIs, 97
- wifi\_event\_loop\_send
  - WIFI Common APIs, 98
- wifi\_event\_loop\_set\_cb
  - WIFI Common APIs, 98
- wifi\_event\_notify\_cb\_t
  - WIFI APIs, 93
- wifi\_event\_process\_handler
  - WIFI APIs, 94
  - WIFI Common APIs, 99
- wifi\_event\_sta\_connected\_t, 209
  - authmode, 210
  - bssid, 210
  - channel, 210
  - ssid, 210
  - ssid\_len, 210
- wifi\_event\_sta\_disconnected\_t, 210
  - bssid, 211
  - reason, 211
  - ssid, 211
  - ssid\_len, 211
- wifi\_event\_sta\_got\_ip\_t, 211
  - ip\_changed, 212
- wifi\_event\_sta\_scan\_done\_t, 212
  - number, 212
  - scan\_id, 212
  - status, 212
- wifi\_event\_t
  - Enumeration, 122
- wifi\_fast\_connect\_get\_mode
  - WIFI STA APIs, 115
- wifi\_fast\_connect\_set\_mode
  - WIFI STA APIs, 115
- wifi\_fast\_connect\_start
  - WIFI STA APIs, 115
- wifi\_fast\_scan\_threshold\_t, 213
  - authmode, 213
  - rss, 213

- wifi\_get\_config
  - WIFI STA APIs, 116
- wifi\_get\_fast\_conn\_mode
  - WIFI STA APIs, 116
- wifi\_init
  - WIFI STA APIs, 116
- wifi\_init\_complete\_cb\_t
  - WIFI STA APIs, 102
- wifi\_init\_config\_t, 213
  - event\_handler, 214
  - magic, 214
- wifi\_install\_default\_event\_handlers
  - WIFI APIs, 94
- wifi\_mode\_t
  - Enumeration, 123
- wifi\_reason\_code\_t
  - Enumeration, 123
- wifi\_register\_event\_handler
  - WIFI APIs, 94
- wifi\_result\_t
  - WIFI STA APIs, 102
- wifi\_scan\_config\_t, 214
  - bssid, 215
  - channel, 215
  - scan\_time, 215
  - scan\_type, 215
  - show\_hidden, 215
  - ssid, 215
- wifi\_scan\_get\_ap\_list
  - WIFI STA APIs, 117
- wifi\_scan\_get\_ap\_num
  - WIFI STA APIs, 117
- wifi\_scan\_get\_ap\_records
  - WIFI STA APIs, 118
- wifi\_scan\_info\_t, 215
  - auth\_mode, 216
  - beacon\_interval, 216
  - bssid, 216
  - capability\_info, 216
  - channel, 216
  - group\_cipher, 217
  - pairwise\_cipher, 217
  - rsi, 217
  - ssid, 217
  - ssid\_length, 217
- wifi\_scan\_list\_t, 217
  - ap\_record, 218
  - num, 218
- wifi\_scan\_method\_t
  - Enumeration, 124
- wifi\_scan\_scan\_stop
  - WIFI STA APIs, 118
- wifi\_scan\_start
  - WIFI STA APIs, 118
- wifi\_scan\_time\_t, 218
  - active, 218
  - passive, 218
- wifi\_scan\_type\_t
  - Enumeration, 124
- wifi\_set\_config
  - WIFI STA APIs, 119
- wifi\_sort\_method\_t
  - Enumeration, 126
- wifi\_sta\_config\_t, 219
  - bssid, 219
  - bssid\_present, 219
  - password, 219
  - password\_length, 220
  - scan\_method, 220
  - sort\_method, 220
  - ssid, 220
  - ssid\_length, 220
  - threshold, 220
- wifi\_sta\_get\_ap\_info
  - WIFI STA APIs, 119
- wifi\_start
  - WIFI STA APIs, 120
- wifi\_stop
  - WIFI STA APIs, 120
- window
  - LE\_GAP\_SCAN\_PARAM\_T, 155
- wpa\_data
  - auto\_conn\_info\_t, 129
  - mw\_wifi\_auto\_connect\_ap\_info\_t, 197
  - wifi\_auto\_connect\_info\_f, 207
- wpa\_ie
  - auto\_conn\_info\_t, 129
  - mw\_wifi\_auto\_connect\_ap\_info\_t, 198
  - wifi\_auto\_connect\_info\_f, 207