OPL1000_WIFI_BLE_API_GUIDE

MP2.6

Generated by Doxygen 1.8.14

Contents

1 Module Index **1**

1.1 Modules 1

2 Data Structure Index **3**

2.1 Data Structures 3

3 Module Documentation **7**

3.1 BLE ALL APIs 7

3.1.1 Detailed Description 7

3.1.2 Function Documentation 7

3.1.2.1 LeSmpGetBondIdFromAddr() 7

3.2 BLE CM APIs 8

3.2.1 Detailed Description 9

3.2.2 Typedef Documentation 9

3.2.2.1 LE_CM_MSG_ADD_TO_RESOLVING_LIST_CFM_T 9

3.2.2.2 LE_CM_MSG_ADD_TO_WHITE_LIST_CFM_T 9

3.2.2.3 LE_CM_MSG_CANCEL_CONNECTION_CFM_T 10

3.2.2.4 LE_CM_MSG_CLEAR_RESOLVING_LIST_CFM_T 10

3.2.2.5 LE_CM_MSG_CLEAR_WHITE_LIST_CFM_T 10

3.2.2.6 LE_CM_MSG_CREATE_CONNECTION_CFM_T 10

3.2.2.7 LE_CM_MSG_ENTER_ADVERTISING_CFM_T 10

3.2.2.8 LE_CM_MSG_ENTER_SCANNING_CFM_T 10

3.2.2.9 LE_CM_MSG_EXIT_ADVERTISING_CFM_T 10

3.2.2.10 LE_CM_MSG_EXIT_SCANNING_CFM_T 10

3.2.2.11 LE_CM_MSG_PHY_UPDATE_COMPLETE_IND_T . . . 11
3.2.2.12 LE_CM_MSG_REMOVE_FROM_RESOLVING_LIST_CFM_T . . . 11
3.2.2.13 LE_CM_MSG_REMOVE_FROM_WHITE_LIST_CFM_T . . . 11
3.2.2.14 LE_CM_MSG_SET_ADVERTISING_DATA_CFM_T . . . 11
3.2.2.15 LE_CM_MSG_SET_ADVERTISING_PARAMS_CFM_T . . . 11
3.2.2.16 LE_CM_MSG_SET_CHANNEL_MAP_CFM_T . . . 11
3.2.2.17 LE_CM_MSG_SET_DEFAULT_PHY_CFM_T . . . 11
3.2.2.18 LE_CM_MSG_SET_RANDOM_ADDRESS_CFM_T . . . 11
3.2.2.19 LE_CM_MSG_SET_RPA_TIMEOUT_CFM_T . . . 12
3.2.2.20 LE_CM_MSG_SET_SCAN_PARAMS_CFM_T . . . 12
3.2.2.21 LE_CM_MSG_SET_SCAN_RSP_DATA_CFM_T . . . 12
3.2.3 Enumeration Type Documentation . . . 12
3.2.3.1 anonymous enum . . . 12
3.2.4 Function Documentation . . . 13
3.2.4.1 LeCmInit() . . . 13
3.3 BLE GAP APIs . . . 15
3.3.1 Detailed Description . . . 17
3.3.2 Macro Definition Documentation . . . 17
3.3.2.1 GAP_ADTYPE_128BIT_COMPLETE . . . 17
3.3.2.2 GAP_ADTYPE_128BIT_MORE . . . 17
3.3.2.3 GAP_ADTYPE_16BIT_COMPLETE . . . 18
3.3.2.4 GAP_ADTYPE_16BIT_MORE . . . 18
3.3.2.5 GAP_ADTYPE_32BIT_COMPLETE . . . 18
3.3.2.6 GAP_ADTYPE_32BIT_MORE . . . 18
3.3.2.7 GAP_ADTYPE_3D_INFO_DATA . . . 18
3.3.2.8 GAP_ADTYPE_ADV_INTERVAL . . . 18
3.3.2.9 GAP_ADTYPE_APPEARANCE . . . 18
3.3.2.10 GAP_ADTYPE_FLAGS . . . 18
3.3.2.11 GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED . . . 19
3.3.2.12 GAP_ADTYPE_FLAGS_GENERAL . . . 19

3.3.2.13 GAP_ADTYPE_FLAGS_LIMITED 19
3.3.2.14 GAP_ADTYPE_LE_BD_ADDR 19
3.3.2.15 GAP_ADTYPE_LE_ROLE 19
3.3.2.16 GAP_ADTYPE_LOCAL_NAME_COMPLETE 19
3.3.2.17 GAP_ADTYPE_LOCAL_NAME_SHORT 19
3.3.2.18 GAP_ADTYPE_MANUFACTURER_SPECIFIC 19
3.3.2.19 GAP_ADTYPE_OOB_CLASS_OF_DEVICE 20
3.3.2.20 GAP_ADTYPE_OOB_SIMPLE_PAIRING_HASHC 20
3.3.2.21 GAP_ADTYPE_OOB_SIMPLE_PAIRING_RANDR 20
3.3.2.22 GAP_ADTYPE_POWER_LEVEL 20
3.3.2.23 GAP_ADTYPE_PUBLIC_TARGET_ADDR 20
3.3.2.24 GAP_ADTYPE_RANDOM_TARGET_ADDR 20
3.3.2.25 GAP_ADTYPE_SERVICE_DATA 20
3.3.2.26 GAP_ADTYPE_SERVICE_DATA_128BIT 20
3.3.2.27 GAP_ADTYPE_SERVICE_DATA_32BIT 21
3.3.2.28 GAP_ADTYPE_SERVICES_LIST_128BIT 21
3.3.2.29 GAP_ADTYPE_SERVICES_LIST_16BIT 21
3.3.2.30 GAP_ADTYPE_SIGNED_DATA 21
3.3.2.31 GAP_ADTYPE_SIMPLE_PAIRING_HASHC_256 21
3.3.2.32 GAP_ADTYPE_SIMPLE_PAIRING_RANDR_256 21
3.3.2.33 GAP_ADTYPE_SLAVE_CONN_INTERVAL_RANGE 21
3.3.2.34 GAP_ADTYPE_SM_OOB_FLAG 21
3.3.2.35 GAP_ADTYPE_SM_TK 22
3.3.2.36 GAP_PUBLIC_ADDR 22
3.3.2.37 GAP_RAND_ADDR_NRPA 22
3.3.2.38 GAP_RAND_ADDR_RPA 22
3.3.2.39 GAP_RAND_ADDR_STATIC 22
3.3.2.40 GAP_SCAN_TYPE_ACTIVE 22
3.3.2.41 GAP_SCAN_TYPE_PASSIVE 22
3.3.2.42 GAP_TX_PWR_CURR_VAL 22

3.3.2.43 GAP_TX_PWR_MAX_VAL . . . 23
3.3.2.44 GAPBOND_IO_CAP_DISPLAY_ONLY . . . 23
3.3.2.45 GAPBOND_IO_CAP_DISPLAY_YES_NO . . . 23
3.3.2.46 GAPBOND_IO_CAP_KEYBOARD_DISPLAY . . . 23
3.3.2.47 GAPBOND_IO_CAP_KEYBOARD_ONLY . . . 23
3.3.2.48 GAPBOND_IO_CAP_NO_INPUT_NO_OUTPUT . . . 23
3.3.2.49 GAPBOND_PAIRING_MODE_INITIATE . . . 23
3.3.2.50 GAPBOND_PAIRING_MODE_NO_PAIRING . . . 23
3.3.2.51 GAPBOND_PAIRING_MODE_WAIT_FOR_REQ . . . 24
3.3.2.52 LE_GAP_ADV_MAX_SIZE . . . 24
3.3.3 Function Documentation . . . 24
3.3.3.1 LeGapAddToResolvingList() . . . 24
3.3.3.2 LeGapAddToWhiteList() . . . 24
3.3.3.3 LeGapAdvertisingEnable() . . . 25
3.3.3.4 LeGapCentralConnectReq() . . . 25
3.3.3.5 LeGapCentralSetDataChannel() . . . 25
3.3.3.6 LeGapClearResolvingList() . . . 27
3.3.3.7 LeGapClearWhiteList() . . . 27
3.3.3.8 LeGapConnectCancelReq() . . . 27
3.3.3.9 LeGapConnParaRequestRsp() . . . 27
3.3.3.10 LeGapConnUpdateRequest() . . . 28
3.3.3.11 LeGapConnUpdateResponse() . . . 28
3.3.3.12 LeGapDisconnectReq() . . . 29
3.3.3.13 LeGapGenRandAddr() . . . 29
3.3.3.14 LeGapGetBtAddr() . . . 29
3.3.3.15 LeGapReadAdvChannelTxPower() . . . 30
3.3.3.16 LeGapReadChannelMap() . . . 30
3.3.3.17 LeGapReadPhy() . . . 30
3.3.3.18 LeGapReadResolvingListSize() . . . 30
3.3.3.19 LeGapReadRssi() . . . 30

3.3.3.20 LeGapReadTxPower() . . . 31
3.3.3.21 LeGapReadWhiteListSize() . . . 31
3.3.3.22 LeGapRemoveFromWhiteList() . . . 31
3.3.3.23 LeGapScanningReq() . . . 32
3.3.3.24 LeGapSetAdvData() . . . 32
3.3.3.25 LeGapSetAdvParameter() . . . 33
3.3.3.26 LeGapSetConnParameter() . . . 33
3.3.3.27 LeGapSetDataChannelPduLen() . . . 33
3.3.3.28 LeGapSetDefaultPhy() . . . 34
3.3.3.29 LeGapSetPhy() . . . 34
3.3.3.30 LeGapSetRandAddr() . . . 34
3.3.3.31 LeGapSetRpaTimeout() . . . 35
3.3.3.32 LeGapSetStaticAddr() . . . 35
3.3.3.33 LeSetScanParameter() . . . 35
3.3.3.34 LeSetScanRspData() . . . 36
3.4 BLE GATT APIs . . . 37
3.4.1 Detailed Description . . . 41
3.4.2 Macro Definition Documentation . . . 41
3.4.2.1 CHAR_AGGREGATE_DESCRIPTOR . . . 41
3.4.2.2 CHAR_CLIENT_CONFIG_DESCRIPTOR . . . 42
3.4.2.3 CHAR_DECL_UUID16_ATTR_VAL . . . 42
3.4.2.4 CHAR_EXT_PROP_DESCRIPTOR . . . 42
3.4.2.5 CHAR_PRESENT_FORMAT_DESCRIPTOR . . . 42
3.4.2.6 CHAR_SERVER_CONFIG_DESCRIPTOR . . . 42
3.4.2.7 CHAR_USER_DESC_DESCRIPTOR . . . 42
3.4.2.8 CHARACTERISTIC_DECL_UUID128 . . . 43
3.4.2.9 CHARACTERISTIC_DECL_UUID16 . . . 43
3.4.2.10 CHARACTERISTIC_UUID128 . . . 43
3.4.2.11 CHARACTERISTIC_UUID16 . . . 43
3.4.2.12 GATT_CHAR_AGG_FORMAT_UUID . . . 43

3.4.2.13 GATT_CHAR_EXT_PROPS_UUID 43
3.4.2.14 GATT_CHAR_FORMAT_UUID 44
3.4.2.15 GATT_CHAR_USER_DESC_UUID 44
3.4.2.16 GATT_CHARACTERISTIC_UUID 44
3.4.2.17 GATT_CLIENT_CHAR_CFG_UUID 44
3.4.2.18 GATT_EXT_REPORT_REF_UUID 44
3.4.2.19 GATT_INCLUDE_UUID 44
3.4.2.20 GATT_PRIMARY_SERVICE_UUID 44
3.4.2.21 GATT_REPORT_REF_UUID 44
3.4.2.22 GATT_SECONDARY_SERVICE_UUID 45
3.4.2.23 GATT_SERV_CHAR_CFG_UUID 45
3.4.2.24 GATT_VALID_RANGE_UUID 45
3.4.2.25 INCLUDE_DECL_UUID128 45
3.4.2.26 INCLUDE_DECL_UUID128_ATTR_VAL 45
3.4.2.27 INCLUDE_DECL_UUID16_ATTR_VAL 45
3.4.2.28 INCLUDE_DECL_UUINT16 45
3.4.2.29 LE_ATT_UUID_SIZE 46
3.4.2.30 LE_GATT_CHAR_PROP_AUTH 46
3.4.2.31 LE_GATT_CHAR_PROP_BCAST 46
3.4.2.32 LE_GATT_CHAR_PROP_EXT_PROP 46
3.4.2.33 LE_GATT_CHAR_PROP_IND 46
3.4.2.34 LE_GATT_CHAR_PROP_NTF 46
3.4.2.35 LE_GATT_CHAR_PROP_RD 46
3.4.2.36 LE_GATT_CHAR_PROP_WR 47
3.4.2.37 LE_GATT_CHAR_PROP_WR_NO_RESP 47
3.4.2.38 LE_GATT_CLIENT_CFG_INDICATION 47
3.4.2.39 LE_GATT_CLIENT_CFG_NOTIFICATION 47
3.4.2.40 LE_GATT_EXT_PROP_RELIABLE_WR 47
3.4.2.41 LE_GATT_EXT_PROP_WR_AUX 47
3.4.2.42 LE_GATT_FLAG_PREPARE_WRITE 47

3.4.2.43 LE_GATT_FLAG_WRITE_CMD . . . 47
3.4.2.44 LE_GATT_FLAG_WRITE_REQ . . . 48
3.4.2.45 LE_GATT_PERM_AUTH_READABLE . . . 48
3.4.2.46 LE_GATT_PERM_AUTH_WRITABLE . . . 48
3.4.2.47 LE_GATT_PERM_NONE . . . 48
3.4.2.48 LE_GATT_PERM_READ . . . 48
3.4.2.49 LE_GATT_PERM_RELIABLE_WRITE . . . 48
3.4.2.50 LE_GATT_PERM_WRITE_CMD . . . 48
3.4.2.51 LE_GATT_PERM_WRITE_REQ . . . 48
3.4.2.52 LE_GATT_PERMIT_AUTHEN_READ . . . 49
3.4.2.53 LE_GATT_PERMIT_AUTHEN_WRITE . . . 49
3.4.2.54 LE_GATT_PERMIT_AUTHOR_READ . . . 49
3.4.2.55 LE_GATT_PERMIT_AUTHOR_WRITE . . . 49
3.4.2.56 LE_GATT_PERMIT_ENCRYPT_READ . . . 49
3.4.2.57 LE_GATT_PERMIT_ENCRYPT_WRITE . . . 49
3.4.2.58 LE_GATT_PERMIT_READ . . . 49
3.4.2.59 LE_GATT_PERMIT_READABLE . . . 49
3.4.2.60 LE_GATT_PERMIT_SC_AUTHEN_READ . . . 50
3.4.2.61 LE_GATT_PERMIT_SC_AUTHEN_WRITE . . . 50
3.4.2.62 LE_GATT_PERMIT_WRITABLE . . . 50
3.4.2.63 LE_GATT_PERMIT_WRITE . . . 50
3.4.2.64 PRIMARY_SERVICE_DECL_UUID128 . . . 50
3.4.2.65 PRIMARY_SERVICE_DECL_UUID16 . . . 50
3.4.2.66 SECONDARY_SERVICE_DECL_UUID128 . . . 50
3.4.2.67 SECONDARY_SERVICE_DECL_UUID16 . . . 51
3.4.3 Enumeration Type Documentation . . . 51
3.4.3.1 anonymous enum . . . 51
3.4.4 Function Documentation . . . 52
3.4.4.1 LeGattAccessReadRsp() . . . 52
3.4.4.2 LeGattAccessWriteRsp() . . . 52

3.4.4.3 LeGattChangeAttrVal() . . . 53
3.4.4.4 LeGattCharValConfirmation() . . . 53
3.4.4.5 LeGattCharValIndicate() . . . 54
3.4.4.6 LeGattCharValNotify() . . . 54
3.4.4.7 LeGattExchangeMtuReq() . . . 55
3.4.4.8 LeGattExchangeMtuRsp() . . . 55
3.4.4.9 LeGattExecuteWriteCharValReliable() . . . 56
3.4.4.10 LeGattFindAllCharacteristic() . . . 56
3.4.4.11 LeGattFindAllCharDescriptor() . . . 56
3.4.4.12 LeGattFindAllPrimaryService() . . . 57
3.4.4.13 LeGattFindCharacteristicByUuid() . . . 57
3.4.4.14 LeGattFindIncludedService() . . . 58
3.4.4.15 LeGattFindPrimaryServiceByUuid() . . . 58
3.4.4.16 LeGattGetAttrHandle() . . . 59
3.4.4.17 LeGattGetAttrVal() . . . 59
3.4.4.18 LeGattGetAttrValLen() . . . 59
3.4.4.19 LeGattGetAttrValMaxLen() . . . 61
3.4.4.20 LeGattInit() . . . 61
3.4.4.21 LeGattModifyAttrVal() . . . 62
3.4.4.22 LeGattPrepareWriteCharValReliable() . . . 62
3.4.4.23 LeGattReadCharValByUuid() . . . 63
3.4.4.24 LeGattReadCharValue() . . . 63
3.4.4.25 LeGattReadLongCharVal() . . . 64
3.4.4.26 LeGattReadMultipleCharVal() . . . 64
3.4.4.27 LeGattRegisterIncludeService() . . . 64
3.4.4.28 LeGattRegisterService() . . . 65
3.4.4.29 LeGattSignedWriteNoRsp() . . . 65
3.4.4.30 LeGattStopCurrentProcedure() . . . 66
3.4.4.31 LeGattWriteCharVal() . . . 66
3.4.4.32 LeGattWriteCharValReliable() . . . 67

3.4.4.33 LeGattWriteLongCharVal() . . . 67
3.4.4.34 LeGattWriteNoRsp() . . . 68
3.4.5 Variable Documentation . . . 68
3.4.5.1 gcCharacteristicUuid . . . 68
3.4.5.2 gcCharAggregateUuid . . . 68
3.4.5.3 gcCharExtPropUuid . . . 69
3.4.5.4 gcCharFormatUuid . . . 69
3.4.5.5 gcCharUserDescUuid . . . 69
3.4.5.6 gcClientCharConfigUuid . . . 69
3.4.5.7 gcExtReportRefUuid . . . 69
3.4.5.8 gcIncludeUuid . . . 69
3.4.5.9 gcPrimaryServiceUuid . . . 69
3.4.5.10 gcReportRefUuid . . . 69
3.4.5.11 gcSecondaryServiceUuid . . . 70
3.4.5.12 gcServerCharConfigUuid . . . 70
3.4.5.13 gcValidRangeUuid . . . 70
3.5 BLE MSG APIs . . . 71
3.5.1 Detailed Description . . . 72
3.5.2 Macro Definition Documentation . . . 72
3.5.2.1 LE_ATT_MSG_BASE . . . 72
3.5.2.2 LE_CM_MSG_BASE . . . 72
3.5.2.3 LE_GATT_MSG_BASE . . . 73
3.5.2.4 LE_HCI_MSG_BASE . . . 73
3.5.2.5 LE_L2CAP_MSG_BASE . . . 73
3.5.2.6 LE_SMP_MSG_BASE . . . 73
3.5.2.7 LE_SYS_MSG_BASE . . . 73
3.5.2.8 MESSAGE_ALLOCATE . . . 73
3.5.2.9 MESSAGE_BULID . . . 73
3.5.2.10 MESSAGE_DATA_BULID . . . 74
3.5.2.11 MESSAGE_OFFSET . . . 74

3.5.2.12 T_HOUR . . . 74
3.5.2.13 T_MIN . . . 74
3.5.2.14 T_SEC . . . 74
3.5.3 Typedef Documentation . . . 74
3.5.3.1 MESSAGE . . . 74
3.5.3.2 MESSAGEID . . . 75
3.5.3.3 MsgData . . . 75
3.5.3.4 MsgLock . . . 75
3.5.3.5 MSGLOCK . . . 75
3.5.3.6 MSGSUBID . . . 75
3.5.3.7 MSGTIMER . . . 75
3.5.3.8 Task . . . 75
3.5.3.9 TASK . . . 75
3.5.3.10 TASKHANDLER . . . 76
3.5.3.11 TASKPACK . . . 76
3.5.4 Enumeration Type Documentation . . . 76
3.5.4.1 anonymous enum . . . 76
3.5.5 Function Documentation . . . 76
3.5.5.1 LeCancelAllMessage() . . . 76
3.5.5.2 LeCancelAllSubMessage() . . . 77
3.5.5.3 LeCancelFirstMessage() . . . 77
3.5.5.4 LeCancelFirstSubMessage() . . . 78
3.5.5.5 LeGetSubMsgId() . . . 78
3.5.5.6 LeHostCreateTask() . . . 78
3.5.5.7 LeHostMessageLoop() . . . 79
3.5.5.8 LeSendMessage() . . . 79
3.5.5.9 LeSendMessageAfter() . . . 79
3.5.5.10 LeSendMessageUnlock() . . . 80
3.5.5.11 LeSendSubMessage() . . . 80
3.5.5.12 LeSendSubMessageAfter() . . . 81

3.5.5.13 LeSendSubMessageUnlock() . . . 81
3.6 BLE SMP APIs . . . 83
3.6.1 Detailed Description . . . 84
3.6.2 Macro Definition Documentation . . . 84
3.6.2.1 LE_MAX_BOND_COUNT . . . 84
3.6.2.2 LE_SM_IO_CAP_DISP_ONLY . . . 84
3.6.2.3 LE_SM_IO_CAP_DISP_YES_NO . . . 84
3.6.2.4 LE_SM_IO_CAP_KEYBOARD_DISP . . . 85
3.6.2.5 LE_SM_IO_CAP_KEYBOARD_ONLY . . . 85
3.6.2.6 LE_SM_IO_CAP_NO_IO . . . 85
3.6.2.7 LE_SM_PAIR_MITM_NO . . . 85
3.6.2.8 LE_SM_PAIR_MITM_YES . . . 85
3.6.2.9 LE_SM_PAIR_OOB_NO . . . 85
3.6.2.10 LE_SM_PAIR_OOB_YES . . . 85
3.6.2.11 LE_SM_PAIR_SC_NO . . . 85
3.6.2.12 LE_SM_PAIR_SC_YES . . . 86
3.6.3 Enumeration Type Documentation . . . 86
3.6.3.1 anonymous enum . . . 86
3.6.3.2 anonymous enum . . . 86
3.6.4 Function Documentation . . . 87
3.6.4.1 LeSmpInit() . . . 87
3.6.4.2 LeSmpOobAuthDataRsp() . . . 87
3.6.4.3 LeSmpOobPresent() . . . 87
3.6.4.4 LeSmpPasskeyInput() . . . 88
3.6.4.5 LeSmpScOobComputeConfirmVal() . . . 88
3.6.4.6 LeSmpScOobDataRsp() . . . 89
3.6.4.7 LeSmpSecurityReq() . . . 89
3.6.4.8 LeSmpSecurityRsp() . . . 89
3.6.4.9 LeSmpSetDefaultConfig() . . . 90
3.6.4.10 LeSmpUserConfirmRsp() . . . 90

3.7 WIFI APIs . . . 91
3.7.1 Detailed Description . . . 92
3.7.2 Macro Definition Documentation . . . 92
3.7.2.1 WIFI_BEACON_INTERVAL_LENGTH . . . 92
3.7.2.2 WIFI_CAPABILITY_INFO_LENGTH . . . 93
3.7.2.3 WIFI_LENGTH_802_11 . . . 93
3.7.2.4 WIFI_LENGTH_PASSPHRASE . . . 93
3.7.2.5 WIFI_MAC_ADDRESS_LENGTH . . . 93
3.7.2.6 WIFI_MAC_NUM_OF_CHANNELS . . . 93
3.7.2.7 WIFI_MAX_LENGTH_OF_SSID . . . 93
3.7.2.8 WIFI_MAX_SCAN_AP_NUM . . . 94
3.7.2.9 WIFI_MAX_SUPPORTED_RATES . . . 94
3.7.3 Typedef Documentation . . . 94
3.7.3.1 wifi_ap_record_t . . . 94
3.7.3.2 wifi_event_notify_cb_t . . . 94
3.7.4 Enumeration Type Documentation . . . 94
3.7.4.1 wifi_auto_connet_mode_e . . . 94
3.7.5 Function Documentation . . . 95
3.7.5.1 wifi_event_process_handler() . . . 95
3.7.5.2 wifi_install_default_event_handlers() . . . 95
3.7.5.3 wifi_register_event_handler() . . . 96
3.8 WIFI Common APIs . . . 97
3.8.1 Detailed Description . . . 97
3.8.2 Typedef Documentation . . . 97
3.8.2.1 wifi_event_cb_t . . . 97
3.8.3 Function Documentation . . . 97
3.8.3.1 wifi_event_loop_init() . . . 97
3.8.3.2 wifi_event_loop_send() . . . 98
3.8.3.3 wifi_event_loop_set_cb() . . . 98
3.8.3.4 wifi_event_process_handler() . . . 99

3.9 WIFI STA APIs . . . 100
3.9.1 Detailed Description . . . 104
3.9.2 Macro Definition Documentation . . . 104
3.9.2.1 WIFI_READY_TIME . . . 104
3.9.3 Typedef Documentation . . . 104
3.9.3.1 wifi_auto_connect_clear_ap_info_fp_t . . . 105
3.9.3.2 wifi_auto_connect_get_ap_info_fp_t . . . 105
3.9.3.3 wifi_auto_connect_get_ap_num_fp_t . . . 105
3.9.3.4 wifi_auto_connect_get_mode_fp_t . . . 105
3.9.3.5 wifi_auto_connect_get_saved_ap_num_fp_t . . . 105
3.9.3.6 wifi_auto_connect_init_fp_t . . . 105
3.9.3.7 wifi_auto_connect_reset_fp_t . . . 105
3.9.3.8 wifi_auto_connect_set_ap_num_fp_t . . . 106
3.9.3.9 wifi_auto_connect_set_mode_fp_t . . . 106
3.9.3.10 wifi_auto_connect_start_fp_t . . . 106
3.9.3.11 wifi_auto_connect_update_ch_fp_t . . . 106
3.9.3.12 wifi_config_get_bandwidth_fp_t . . . 106
3.9.3.13 wifi_config_get_bssid_fp_t . . . 106
3.9.3.14 wifi_config_get_channel_fp_t . . . 106
3.9.3.15 wifi_config_get_dtim_interval_fp_t . . . 106
3.9.3.16 wifi_config_get_listen_interval_fp_t . . . 107
3.9.3.17 wifi_config_get_mac_address_fp_t . . . 107
3.9.3.18 wifi_config_get_mac_tx_data_rate_fp_t . . . 107
3.9.3.19 wifi_config_get_opmode_fp_t . . . 107
3.9.3.20 wifi_config_get_skip_dtim_fp_t . . . 107
3.9.3.21 wifi_config_get_ssid_fp_t . . . 107
3.9.3.22 wifi_config_get_sta_mac_address_from_flash_fp_t . . . 107
3.9.3.23 wifi_config_set_bandwidth_fp_t . . . 107
3.9.3.24 wifi_config_set_bssid_fp_t . . . 108
3.9.3.25 wifi_config_set_channel_fp_t . . . 108

3.9.3.26 wifi_config_set_dtim_interval_fp_t . . . 108
3.9.3.27 wifi_config_set_listen_interval_fp_t . . . 108
3.9.3.28 wifi_config_set_mac_address_fp_t . . . 108
3.9.3.29 wifi_config_set_mac_tx_data_rate_fp_t . . . 108
3.9.3.30 wifi_config_set_opmode_fp_t . . . 108
3.9.3.31 wifi_config_set_skip_dtim_fp_t . . . 108
3.9.3.32 wifi_config_set_ssid_fp_t . . . 109
3.9.3.33 wifi_connection_connect_fp_t . . . 109
3.9.3.34 wifi_connection_connect_from_ac_index_fp_t . . . 109
3.9.3.35 wifi_connection_connect_from_ac_list_fp_t . . . 109
3.9.3.36 wifi_connection_disconnect_ap_fp_t . . . 109
3.9.3.37 wifi_connection_disconnect_sta_fp_t . . . 109
3.9.3.38 wifi_connection_get_rssi_fp_t . . . 109
3.9.3.39 wifi_connection_register_event_handler_fp_t . . . 110
3.9.3.40 wifi_connection_scan_start_fp_t . . . 110
3.9.3.41 wifi_connection_unregister_event_handler_fp_t . . . 110
3.9.3.42 wifi_convert_auth_mode_fp_t . . . 110
3.9.3.43 wifi_deinit_fp_t . . . 110
3.9.3.44 wifi_event_handler_t . . . 110
3.9.3.45 wifi_fast_connect_get_mode_fp_t . . . 111
3.9.3.46 wifi_fast_connect_set_mode_fp_t . . . 111
3.9.3.47 wifi_fast_connect_start_fp_t . . . 111
3.9.3.48 wifi_get_config_fp_t . . . 111
3.9.3.49 wifi_init_complete_cb_t . . . 111
3.9.3.50 wifi_init_fp_t . . . 112
3.9.3.51 wifi_result_t . . . 112
3.9.3.52 wifi_scan_get_ap_list_fp_t . . . 112
3.9.3.53 wifi_scan_get_ap_num_fp_t . . . 112
3.9.3.54 wifi_scan_get_ap_records_fp_t . . . 112
3.9.3.55 wifi_scan_start_fp_t . . . 112

3.9.3.56 wifi_scan_stop_fp_t . . . 112
3.9.3.57 wifi_set_config_fp_t . . . 113
3.9.3.58 wifi_sta_get_ap_info_fp_t . . . 113
3.9.3.59 wifi_start_fp_t . . . 113
3.9.3.60 wifi_stop_fp_t . . . 113
3.9.4 Function Documentation . . . 113
3.9.4.1 wifi_auto_connect_clear_ap_info() . . . 113
3.9.4.2 wifi_auto_connect_get_ap_info() . . . 114
3.9.4.3 wifi_auto_connect_get_ap_num() . . . 114
3.9.4.4 wifi_auto_connect_get_mode() . . . 115
3.9.4.5 wifi_auto_connect_get_saved_ap_num() . . . 115
3.9.4.6 wifi_auto_connect_init() . . . 116
3.9.4.7 wifi_auto_connect_reset() . . . 116
3.9.4.8 wifi_auto_connect_set_ap_num() . . . 116
3.9.4.9 wifi_auto_connect_set_mode() . . . 117
3.9.4.10 wifi_auto_connect_start() . . . 117
3.9.4.11 wifi_auto_connect_update_ch() . . . 118
3.9.4.12 wifi_config_get_bandwidth() . . . 118
3.9.4.13 wifi_config_get_bssid() . . . 119
3.9.4.14 wifi_config_get_channel() . . . 119
3.9.4.15 wifi_config_get_dtim_interval() . . . 120
3.9.4.16 wifi_config_get_listen_interval() . . . 120
3.9.4.17 wifi_config_get_mac_address() . . . 120
3.9.4.18 wifi_config_get_mac_tx_data_rate() . . . 121
3.9.4.19 wifi_config_get_opmode() . . . 121
3.9.4.20 wifi_config_get_skip_dtim() . . . 121
3.9.4.21 wifi_config_get_ssid() . . . 122
3.9.4.22 wifi_config_get_sta_mac_address_from_flash() . . . 122
3.9.4.23 wifi_config_set_bandwidth() . . . 122
3.9.4.24 wifi_config_set_bssid() . . . 123

3.9.4.25 wifi_config_set_channel() . . . 123
3.9.4.26 wifi_config_set_dtim_interval() . . . 124
3.9.4.27 wifi_config_set_listen_interval() . . . 124
3.9.4.28 wifi_config_set_mac_address() . . . 124
3.9.4.29 wifi_config_set_mac_tx_data_rate() . . . 125
3.9.4.30 wifi_config_set_opmode() . . . 125
3.9.4.31 wifi_config_set_skip_dtim() . . . 125
3.9.4.32 wifi_config_set_ssid() . . . 126
3.9.4.33 wifi_connection_connect() . . . 127
3.9.4.34 wifi_connection_connect_from_ac_index() . . . 127
3.9.4.35 wifi_connection_connect_from_ac_list() . . . 128
3.9.4.36 wifi_connection_disconnect_ap() . . . 128
3.9.4.37 wifi_connection_disconnect_sta() . . . 128
3.9.4.38 wifi_connection_get_rssi() . . . 129
3.9.4.39 wifi_connection_register_event_handler() . . . 129
3.9.4.40 wifi_connection_scan_start() . . . 130
3.9.4.41 wifi_connection_unregister_event_handler() . . . 130
3.9.4.42 wifi_convert_auth_mode() . . . 131
3.9.4.43 wifi_deinit() . . . 131
3.9.4.44 wifi_fast_connect_get_mode() . . . 131
3.9.4.45 wifi_fast_connect_set_mode() . . . 132
3.9.4.46 wifi_fast_connect_start() . . . 132
3.9.4.47 wifi_get_config() . . . 133
3.9.4.48 wifi_init() . . . 133
3.9.4.49 wifi_scan_get_ap_list() . . . 134
3.9.4.50 wifi_scan_get_ap_num() . . . 134
3.9.4.51 wifi_scan_get_ap_records() . . . 135
3.9.4.52 wifi_scan_start() . . . 135
3.9.4.53 wifi_scan_stop() . . . 136
3.9.4.54 wifi_set_config() . . . 136

3.9.4.55 wifi_sta_get_ap_info() . . . 137
3.9.4.56 wifi_start() . . . 137
3.9.4.57 wifi_stop() . . . 137
3.9.5 Variable Documentation . . . 138
3.9.5.1 wifi_auto_connect_clear_ap_info_api . . . 138
3.9.5.2 wifi_auto_connect_get_ap_info_api . . . 138
3.9.5.3 wifi_auto_connect_get_ap_num_api . . . 138
3.9.5.4 wifi_auto_connect_get_mode_api . . . 138
3.9.5.5 wifi_auto_connect_get_saved_ap_num_api . . . 138
3.9.5.6 wifi_auto_connect_init_api . . . 138
3.9.5.7 wifi_auto_connect_reset_api . . . 138
3.9.5.8 wifi_auto_connect_set_ap_num_api . . . 139
3.9.5.9 wifi_auto_connect_set_mode_api . . . 139
3.9.5.10 wifi_auto_connect_start_api . . . 139
3.9.5.11 wifi_auto_connect_update_ch_api . . . 139
3.9.5.12 wifi_config_get_bandwidth_api . . . 139
3.9.5.13 wifi_config_get_bssid_api . . . 139
3.9.5.14 wifi_config_get_channel_api . . . 139
3.9.5.15 wifi_config_get_dtim_interval_api . . . 139
3.9.5.16 wifi_config_get_listen_interval_api . . . 140
3.9.5.17 wifi_config_get_mac_address_api . . . 140
3.9.5.18 wifi_config_get_mac_tx_data_rate_api . . . 140
3.9.5.19 wifi_config_get_opmode_api . . . 140
3.9.5.20 wifi_config_get_skip_dtim_api . . . 140
3.9.5.21 wifi_config_get_ssid_api . . . 140
3.9.5.22 wifi_config_get_sta_mac_address_from_flash_api . . . 140
3.9.5.23 wifi_config_set_bandwidth_api . . . 140
3.9.5.24 wifi_config_set_bssid_api . . . 141
3.9.5.25 wifi_config_set_channel_api . . . 141
3.9.5.26 wifi_config_set_dtim_interval_api . . . 141

3.9.5.27 wifi_config_set_listen_interval_api . . . 141
3.9.5.28 wifi_config_set_mac_address_api . . . 141
3.9.5.29 wifi_config_set_mac_tx_data_rate_api . . . 141
3.9.5.30 wifi_config_set_opmode_api . . . 141
3.9.5.31 wifi_config_set_skip_dtim_api . . . 141
3.9.5.32 wifi_config_set_ssid_api . . . 142
3.9.5.33 wifi_connection_connect_api . . . 142
3.9.5.34 wifi_connection_connect_from_ac_index_api . . . 142
3.9.5.35 wifi_connection_connect_from_ac_list_api . . . 142
3.9.5.36 wifi_connection_disconnect_ap_api . . . 142
3.9.5.37 wifi_connection_disconnect_sta_api . . . 142
3.9.5.38 wifi_connection_get_rssi_api . . . 142
3.9.5.39 wifi_connection_register_event_handler_api . . . 142
3.9.5.40 wifi_connection_scan_start_api . . . 143
3.9.5.41 wifi_connection_unregister_event_handler_api . . . 143
3.9.5.42 wifi_convert_auth_mode_api . . . 143
3.9.5.43 wifi_deinit_api . . . 143
3.9.5.44 wifi_fast_connect_get_mode_api . . . 143
3.9.5.45 wifi_fast_connect_set_mode_api . . . 143
3.9.5.46 wifi_fast_connect_start_api . . . 143
3.9.5.47 wifi_get_config_api . . . 143
3.9.5.48 wifi_init_api . . . 144
3.9.5.49 wifi_scan_get_ap_list_api . . . 144
3.9.5.50 wifi_scan_get_ap_num_api . . . 144
3.9.5.51 wifi_scan_get_ap_records_api . . . 144
3.9.5.52 wifi_scan_start_api . . . 144
3.9.5.53 wifi_scan_stop_api . . . 144
3.9.5.54 wifi_set_config_api . . . 144
3.9.5.55 wifi_sta_get_ap_info_api . . . 144
3.9.5.56 wifi_start_api . . . 144

3.9.5.57 wifi_stop_api . . . 144
3.10 Enumeration . . . 145
3.10.1 Detailed Description . . . 145
3.10.2 Enumeration Type Documentation . . . 145
3.10.2.1 wifi_auth_mode_t . . . 146
3.10.2.2 wifi_bandwidth_t . . . 146
3.10.2.3 wifi_cipher_type_t . . . 146
3.10.2.4 wifi_event_t . . . 147
3.10.2.5 wifi_mac_data_rate_t . . . 147
3.10.2.6 wifi_mode_t . . . 147
3.10.2.7 wifi_reason_code_t . . . 148
3.10.2.8 wifi_scan_method_t . . . 149
3.10.2.9 wifi_scan_type_t . . . 149
3.10.2.10 wifi_sort_method_t . . . 149

4 Data Structure Documentation **151**

4.1 _wpa_ie_data Struct Reference . . . 151
4.1.1 Field Documentation . . . 151
4.1.1.1 capabilities . . . 151
4.1.1.2 group_cipher . . . 151
4.1.1.3 key_mgmt . . . 152
4.1.1.4 mgmt_group_cipher . . . 152
4.1.1.5 num_pmkid . . . 152
4.1.1.6 pairwise_cipher . . . 152
4.1.1.7 pmkid . . . 152
4.1.1.8 proto . . . 152
4.2 asso_data Struct Reference . . . 152
4.2.1 Field Documentation . . . 153
4.2.1.1 eap_workaround . . . 153
4.2.1.2 eapol_flags . . . 153
4.2.1.3 group_cipher . . . 153

4.2.1.4 key_mgmt . . . 153
4.2.1.5 leap . . . 153
4.2.1.6 mgmt_group_cipher . . . 154
4.2.1.7 non_leap . . . 154
4.2.1.8 pairwise_cipher . . . 154
4.2.1.9 passphrase . . . 154
4.2.1.10 proto . . . 154
4.2.1.11 psk . . . 154
4.2.1.12 psk_set . . . 154
4.3 auto_conn_info_t Struct Reference . . . 154
4.3.1 Field Documentation . . . 155
4.3.1.1 ap_channel . . . 155
4.3.1.2 beacon_interval . . . 155
4.3.1.3 bssid . . . 155
4.3.1.4 capabilities . . . 155
4.3.1.5 dtim_prod . . . 156
4.3.1.6 fast_connect . . . 156
4.3.1.7 free_ocpy . . . 156
4.3.1.8 hid_ssid . . . 156
4.3.1.9 hid_ssid_len . . . 156
4.3.1.10 latest_beacon_rx_time . . . 156
4.3.1.11 passphrase . . . 156
4.3.1.12 psk . . . 156
4.3.1.13 rsn_ie . . . 157
4.3.1.14 rssi . . . 157
4.3.1.15 ssid . . . 157
4.3.1.16 ssid_len . . . 157
4.3.1.17 supported_rates . . . 157
4.3.1.18 wpa_data . . . 157
4.3.1.19 wpa_ie . . . 157

4.4 auto_connect_cfg_t Struct Reference . . . 157
4.4.1 Field Documentation . . . 158
4.4.1.1 flag . . . 158
4.4.1.2 front . . . 158
4.4.1.3 max_save_num . . . 158
4.4.1.4 pFCInfo . . . 158
4.4.1.5 rear . . . 158
4.4.1.6 retryCount . . . 159
4.4.1.7 targetIdx . . . 159
4.4.1.8 uFCApNum . . . 159
4.5 event_msg_t Struct Reference . . . 159
4.5.1 Detailed Description . . . 159
4.5.2 Field Documentation . . . 159
4.5.2.1 event . . . 159
4.5.2.2 length . . . 160
4.5.2.3 param . . . 160
4.6 hap_control_t Struct Reference . . . 160
4.6.1 Field Documentation . . . 160
4.6.1.1 hap_ap_info . . . 160
4.6.1.2 hap_bitvector . . . 160
4.6.1.3 hap_en . . . 160
4.6.1.4 hap_final_index . . . 161
4.6.1.5 hap_index . . . 161
4.6.1.6 hap_ssid . . . 161
4.7 LE_BT_ADDR_T Struct Reference . . . 161
4.7.1 Field Documentation . . . 161
4.7.1.1 addr . . . 161
4.7.1.2 type . . . 161
4.8 LE_CM_CONNECTION_COMPLETE_IND_T Struct Reference . . . 162
4.8.1 Field Documentation . . . 162

4.8.1.1 conn_hdl . . . 162
4.8.1.2 conn_interval . . . 162
4.8.1.3 conn_latency . . . 162
4.8.1.4 dev_id . . . 162
4.8.1.5 peer_addr . . . 163
4.8.1.6 peer_addr_type . . . 163
4.8.1.7 role . . . 163
4.8.1.8 status . . . 163
4.8.1.9 supervison_timeout . . . 163
4.9 LE_CM_MSG_ADVERTISE_REPORT_IND_T Struct Reference . . . 163
4.9.1 Field Documentation . . . 164
4.9.1.1 addr . . . 164
4.9.1.2 addr_type . . . 164
4.9.1.3 data . . . 164
4.9.1.4 event_type . . . 164
4.9.1.5 len . . . 164
4.9.1.6 rssi . . . 164
4.10 LE_CM_MSG_CONN_PARA_REQ_T Struct Reference . . . 164
4.10.1 Field Documentation . . . 165
4.10.1.1 conn_hdl . . . 165
4.10.1.2 itv_max . . . 165
4.10.1.3 itv_min . . . 165
4.10.1.4 latency . . . 165
4.10.1.5 sv_tmo . . . 165
4.11 LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T Struct Reference . . . 165
4.11.1 Field Documentation . . . 166
4.11.1.1 conn_hdl . . . 166
4.11.1.2 interval . . . 166
4.11.1.3 latency . . . 166
4.11.1.4 status . . . 166

4.11.1.5 supervision_timeout . . . 166
4.12 LE_CM_MSG_DATA_LEN_CHANGE_IND_T Struct Reference . . . 166
4.12.1 Field Documentation . . . 167
4.12.1.1 conn_hdl . . . 167
4.12.1.2 max_rx_octets . . . 167
4.12.1.3 max_rx_time . . . 167
4.12.1.4 max_tx_octets . . . 167
4.12.1.5 max_tx_time . . . 167
4.13 LE_CM_MSG_DIRECT_ADV_REPORT_IND_T Struct Reference . . . 167
4.13.1 Field Documentation . . . 168
4.13.1.1 direct_addr . . . 168
4.13.1.2 direct_addr_type . . . 168
4.13.1.3 peer_addr . . . 168
4.13.1.4 peer_addr_type . . . 168
4.13.1.5 rssi . . . 168
4.14 LE_CM_MSG_DISCONNECT_COMPLETE_IND_T Struct Reference . . . 168
4.14.1 Field Documentation . . . 169
4.14.1.1 conn_hdl . . . 169
4.14.1.2 reason . . . 169
4.14.1.3 status . . . 169
4.15 LE_CM_MSG_ENCRYPTION_CHANGE_IND_T Struct Reference . . . 169
4.15.1 Field Documentation . . . 169
4.15.1.1 conn_hdl . . . 170
4.15.1.2 devid . . . 170
4.15.1.3 enabled . . . 170
4.15.1.4 status . . . 170
4.16 LE_CM_MSG_ENCRYPTION_REFRESH_IND_T Struct Reference . . . 170
4.16.1 Field Documentation . . . 170
4.16.1.1 conn_hdl . . . 170
4.16.1.2 devid . . . 171

4.16.1.3 enabled . . . 171
4.16.1.4 status . . . 171
4.17 LE_CM_MSG_INIT_COMPLETE_CFM_T Struct Reference . . . 171
4.17.1 Field Documentation . . . 171
4.17.1.1 status . . . 171
4.18 LE_CM_MSG_LTK_REQ_IND_T Struct Reference . . . 171
4.18.1 Field Documentation . . . 172
4.18.1.1 conn_hdl . . . 172
4.18.1.2 devid . . . 172
4.18.1.3 ediv . . . 172
4.18.1.4 rand . . . 172
4.19 LE_CM_MSG_READ_ADV_TX_POWER_CFM_T Struct Reference . . . 172
4.19.1 Field Documentation . . . 173
4.19.1.1 pwr_level . . . 173
4.19.1.2 status . . . 173
4.20 LE_CM_MSG_READ_BD_ADDR_CFM_T Struct Reference . . . 173
4.20.1 Field Documentation . . . 173
4.20.1.1 bd_addr . . . 173
4.20.1.2 status . . . 173
4.21 LE_CM_MSG_READ_CHANNEL_MAP_CFM_T Struct Reference . . . 174
4.21.1 Field Documentation . . . 174
4.21.1.1 ch_map . . . 174
4.21.1.2 conn_hdl . . . 174
4.21.1.3 status . . . 174
4.22 LE_CM_MSG_READ_PHY_CFM_T Struct Reference . . . 174
4.22.1 Field Documentation . . . 175
4.22.1.1 conn_hdl . . . 175
4.22.1.2 rx_phy . . . 175
4.22.1.3 status . . . 175
4.22.1.4 tx_phy . . . 175

4.23 LE_CM_MSG_READ_RESOLVING_LIST_SIZE_CFM_T Struct Reference 175
4.23.1 Field Documentation 175
4.23.1.1 size 175
4.23.1.2 status 176
4.24 LE_CM_MSG_READ_RSSI_CFM_T Struct Reference 176
4.24.1 Field Documentation 176
4.24.1.1 conn_hdl 176
4.24.1.2 rssi 176
4.24.1.3 status 176
4.25 LE_CM_MSG_READ_TX_POWER_CFM_T Struct Reference 176
4.25.1 Field Documentation 177
4.25.1.1 conn_hdl 177
4.25.1.2 status 177
4.25.1.3 tx_power 177
4.26 LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM_T Struct Reference 177
4.26.1 Field Documentation 177
4.26.1.1 size 178
4.26.1.2 status 178
4.27 LE_CM_MSG_SET_DATA_LENGTH_CFM_T Struct Reference 178
4.27.1 Field Documentation 178
4.27.1.1 conn_hdl 178
4.27.1.2 status 178
4.28 LE_CM_MSG_SET_DISCONNECT_CFM_T Struct Reference 178
4.28.1 Field Documentation 179
4.28.1.1 handle 179
4.28.1.2 status 179
4.29 LE_CM_MSG_SET_PHY_CFM_T Struct Reference 179
4.29.1 Field Documentation 179
4.29.1.1 conn_hdl 179
4.29.1.2 status 179

4.30 LE_CM_MSG_SIGNAL_UPDATE_REQ_T Struct Reference . . . 180
4.30.1 Field Documentation . . . 180
4.30.1.1 conn_hdl . . . 180
4.30.1.2 identifier . . . 180
4.30.1.3 interval_max . . . 180
4.30.1.4 interval_min . . . 180
4.30.1.5 slave_latency . . . 180
4.30.1.6 timeout_multiplier . . . 181
4.31 LE_CM_REQ_STATUS_T Struct Reference . . . 181
4.31.1 Field Documentation . . . 181
4.31.1.1 status . . . 181
4.32 LE_CONN_PARA_T Struct Reference . . . 181
4.32.1 Field Documentation . . . 181
4.32.1.1 itv_max . . . 181
4.32.1.2 itv_min . . . 182
4.32.1.3 latency . . . 182
4.32.1.4 sv_timeout . . . 182
4.33 LE_GAP_ADVERTISING_PARAM_T Struct Reference . . . 182
4.33.1 Field Documentation . . . 182
4.33.1.1 channel_map . . . 182
4.33.1.2 filter_policy . . . 183
4.33.1.3 interval_max . . . 183
4.33.1.4 interval_min . . . 183
4.33.1.5 own_addr_type . . . 183
4.33.1.6 peer_addr . . . 183
4.33.1.7 peer_addr_type . . . 183
4.33.1.8 type . . . 183
4.34 LE_GAP_CONN_PARAM_T Struct Reference . . . 183
4.34.1 Field Documentation . . . 184
4.34.1.1 interval_max . . . 184

4.34.1.2 interval_min . . . 184
4.34.1.3 latency . . . 184
4.34.1.4 supervision_timeout . . . 184
4.35 LE_GAP_SCAN_PARAM_T Struct Reference . . . 184
4.35.1 Field Documentation . . . 185
4.35.1.1 filter_policy . . . 185
4.35.1.2 interval . . . 185
4.35.1.3 own_addr_type . . . 185
4.35.1.4 type . . . 185
4.35.1.5 window . . . 185
4.36 LE_GATT_ATTR_T Struct Reference . . . 185
4.36.1 Field Documentation . . . 186
4.36.1.1 format . . . 186
4.36.1.2 handle . . . 186
4.36.1.3 len . . . 186
4.36.1.4 maxLen . . . 186
4.36.1.5 permit . . . 186
4.36.1.6 pUuid . . . 186
4.36.1.7 pVal . . . 186
4.37 LE_GATT_MSG_ACCESS_READ_IND_T Struct Reference . . . 187
4.37.1 Field Documentation . . . 187
4.37.1.1 conn_hdl . . . 187
4.37.1.2 devid . . . 187
4.37.1.3 handle . . . 187
4.37.1.4 offset . . . 187
4.38 LE_GATT_MSG_ACCESS_WRITE_IND_T Struct Reference . . . 187
4.38.1 Field Documentation . . . 188
4.38.1.1 conn_hdl . . . 188
4.38.1.2 devid . . . 188
4.38.1.3 flag . . . 188

4.38.1.4 handle . . . 188
4.38.1.5 len . . . 188
4.38.1.6 offset . . . 189
4.38.1.7 pVal . . . 189
4.39 LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND_T Struct Reference . . . 189
4.39.1 Field Documentation . . . 189
4.39.1.1 conn_hdl . . . 189
4.39.1.2 devid . . . 189
4.39.1.3 format . . . 189
4.39.1.4 handle . . . 190
4.39.1.5 uuid . . . 190
4.40 LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND_T Struct Reference . . . 190
4.40.1 Field Documentation . . . 190
4.40.1.1 conn_hdl . . . 190
4.40.1.2 devid . . . 190
4.40.1.3 format . . . 191
4.40.1.4 handle . . . 191
4.40.1.5 property . . . 191
4.40.1.6 uuid . . . 191
4.40.1.7 val_hdl . . . 191
4.41 LE_GATT_MSG_CHARACTERISTIC_VAL_IND_T Struct Reference . . . 191
4.41.1 Field Documentation . . . 192
4.41.1.1 att_err . . . 192
4.41.1.2 conn_hdl . . . 192
4.41.1.3 devid . . . 192
4.41.1.4 handle . . . 192
4.41.1.5 len . . . 192
4.41.1.6 offset . . . 192
4.41.1.7 val . . . 192
4.42 LE_GATT_MSG_CONFIRMATION_CFM_T Struct Reference . . . 193

4.42.1 Field Documentation . . . 193
4.42.1.1 conn_hdl . . . 193
4.42.1.2 devid . . . 193
4.42.1.3 handle . . . 193
4.43 LE_GATT_MSG_EXCHANGE_MTU_CFM_T Struct Reference . . . 193
4.43.1 Field Documentation . . . 194
4.43.1.1 conn_hdl . . . 194
4.43.1.2 current_rx_mtu . . . 194
4.43.1.3 devid . . . 194
4.44 LE_GATT_MSG_EXCHANGE_MTU_IND_T Struct Reference . . . 194
4.44.1 Field Documentation . . . 194
4.44.1.1 client_rx_mtu . . . 194
4.44.1.2 conn_hdl . . . 195
4.44.1.3 devid . . . 195
4.45 LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM_T Struct Reference . . . 195
4.45.1 Field Documentation . . . 195
4.45.1.1 att_err . . . 195
4.45.1.2 conn_hdl . . . 195
4.45.1.3 devid . . . 195
4.45.1.4 err_hdl . . . 196
4.45.1.5 status . . . 196
4.46 LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM_T Struct Reference . . . 196
4.46.1 Field Documentation . . . 196
4.46.1.1 att_err . . . 196
4.46.1.2 conn_hdl . . . 196
4.46.1.3 devid . . . 196
4.46.1.4 handle . . . 197
4.46.1.5 status . . . 197
4.47 LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_CFM_T Struct Reference . . . 197
4.47.1 Field Documentation . . . 197

4.47.1.1 att_err . . . 197
4.47.1.2 conn_hdl . . . 197
4.47.1.3 devid . . . 197
4.47.1.4 handle . . . 198
4.47.1.5 status . . . 198
4.48 LE_GATT_MSG_FIND_CHARACTERISTIC_CFM_T Struct Reference . . . 198
4.48.1 Field Documentation . . . 198
4.48.1.1 att_err . . . 198
4.48.1.2 conn_hdl . . . 198
4.48.1.3 devid . . . 198
4.48.1.4 handle . . . 199
4.48.1.5 status . . . 199
4.49 LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM_T Struct Reference . . . 199
4.49.1 Field Documentation . . . 199
4.49.1.1 att_err . . . 199
4.49.1.2 conn_hdl . . . 199
4.49.1.3 devid . . . 199
4.49.1.4 handle . . . 200
4.49.1.5 status . . . 200
4.50 LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM_T Struct Reference . . . 200
4.50.1 Field Documentation . . . 200
4.50.1.1 att_err . . . 200
4.50.1.2 conn_hdl . . . 200
4.50.1.3 devid . . . 200
4.50.1.4 handle . . . 201
4.50.1.5 status . . . 201
4.51 LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T Struct Reference . . . 201
4.51.1 Field Documentation . . . 201
4.51.1.1 conn_hdl . . . 201
4.51.1.2 devid . . . 201

4.51.1.3 end_hdl . . . 202
4.51.1.4 format . . . 202
4.51.1.5 handle . . . 202
4.51.1.6 start_hdl . . . 202
4.51.1.7 uuid . . . 202
4.52 LE_GATT_MSG_INDICATE_IND_T Struct Reference . . . 202
4.52.1 Field Documentation . . . 202
4.52.1.1 conn_hdl . . . 203
4.52.1.2 devid . . . 203
4.52.1.3 handle . . . 203
4.52.1.4 len . . . 203
4.52.1.5 val . . . 203
4.53 LE_GATT_MSG_NOTIFY_CFM_T Struct Reference . . . 203
4.53.1 Field Documentation . . . 203
4.53.1.1 conn_hdl . . . 204
4.53.1.2 devid . . . 204
4.53.1.3 handle . . . 204
4.53.1.4 status . . . 204
4.54 LE_GATT_MSG_NOTIFY_IND_T Struct Reference . . . 204
4.54.1 Field Documentation . . . 204
4.54.1.1 conn_hdl . . . 204
4.54.1.2 devid . . . 205
4.54.1.3 handle . . . 205
4.54.1.4 len . . . 205
4.54.1.5 val . . . 205
4.55 LE_GATT_MSG_OPERATION_TIMEOUT_T Struct Reference . . . 205
4.55.1 Field Documentation . . . 205
4.55.1.1 att_op . . . 205
4.55.1.2 conn_hdl . . . 206
4.55.1.3 devid . . . 206

4.56 LE_GATT_MSG_PREPARE_WRITE_RELIABLE_CFM_T Struct Reference 206
4.56.1 Field Documentation 206
4.56.1.1 att_err 206
4.56.1.2 conn_hdl 206
4.56.1.3 devid 206
4.56.1.4 handle 207
4.56.1.5 status 207
4.57 LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_CFM_T Struct Reference 207
4.57.1 Field Documentation 207
4.57.1.1 att_err 207
4.57.1.2 conn_hdl 207
4.57.1.3 devid 207
4.57.1.4 handle 208
4.57.1.5 status 208
4.58 LE_GATT_MSG_READ_CHARACTERISTIC_VALUE_CFM_T Struct Reference 208
4.58.1 Field Documentation 208
4.58.1.1 att_err 208
4.58.1.2 conn_hdl 208
4.58.1.3 devid 208
4.58.1.4 handle 209
4.58.1.5 status 209
4.59 LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM_T Struct Reference 209
4.59.1 Field Documentation 209
4.59.1.1 att_err 209
4.59.1.2 conn_hdl 209
4.59.1.3 devid 209
4.59.1.4 handle 210
4.59.1.5 status 210
4.60 LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL_CFM_T Struct Reference 210
4.60.1 Field Documentation 210

4.60.1.1 att_err . . . 210
4.60.1.2 conn_hdl . . . 210
4.60.1.3 devid . . . 211
4.60.1.4 err_hdl . . . 211
4.60.1.5 len . . . 211
4.60.1.6 status . . . 211
4.60.1.7 val . . . 211
4.61 LE_GATT_MSG_SERVICE_INFO_IND_T Struct Reference . . . 211
4.61.1 Field Documentation . . . 212
4.61.1.1 conn_hdl . . . 212
4.61.1.2 devid . . . 212
4.61.1.3 end_hdl . . . 212
4.61.1.4 format . . . 212
4.61.1.5 start_hdl . . . 212
4.61.1.6 uuid . . . 212
4.62 LE_GATT_MSG_SIGNED_WRITE_CFM_T Struct Reference . . . 212
4.62.1 Field Documentation . . . 213
4.62.1.1 conn_hdl . . . 213
4.62.1.2 devid . . . 213
4.62.1.3 handle . . . 213
4.62.1.4 status . . . 213
4.63 LE_GATT_MSG_WRITE_CHAR_VAL_RELIABLE_CFM_T Struct Reference . . . 213
4.63.1 Field Documentation . . . 214
4.63.1.1 att_err . . . 214
4.63.1.2 conn_hdl . . . 214
4.63.1.3 devid . . . 214
4.63.1.4 handle . . . 214
4.63.1.5 status . . . 214
4.64 LE_GATT_MSG_WRITE_CHAR_VALUE_CFM_T Struct Reference . . . 214
4.64.1 Field Documentation . . . 215

4.64.1.1 att_err . . . 215
4.64.1.2 conn_hdl . . . 215
4.64.1.3 devid . . . 215
4.64.1.4 handle . . . 215
4.64.1.5 status . . . 215
4.65 LE_GATT_MSG_WRITE_LONG_CHAR_VALUE_CFM_T Struct Reference . . . 215
4.65.1 Field Documentation . . . 216
4.65.1.1 att_err . . . 216
4.65.1.2 conn_hdl . . . 216
4.65.1.3 devid . . . 216
4.65.1.4 handle . . . 216
4.65.1.5 status . . . 216
4.66 LE_GATT_MSG_WRITE_NO_RSP_CFM_T Struct Reference . . . 216
4.66.1 Field Documentation . . . 217
4.66.1.1 conn_hdl . . . 217
4.66.1.2 devid . . . 217
4.66.1.3 handle . . . 217
4.66.1.4 status . . . 217
4.67 LE_GATT_SERVICE_T Struct Reference . . . 217
4.67.1 Field Documentation . . . 217
4.67.1.1 endHdl . . . 218
4.67.1.2 pAttr . . . 218
4.67.1.3 startHdl . . . 218
4.67.1.4 svc_id . . . 218
4.68 LE_SMP_MSG_ENCRYPTION_CHANGE_IND_T Struct Reference . . . 218
4.68.1 Field Documentation . . . 218
4.68.1.1 conn_hdl . . . 218
4.68.1.2 enable . . . 219
4.69 LE_SMP_MSG_ENCRYPTION_REFRESH_IND_T Struct Reference . . . 219
4.69.1 Field Documentation . . . 219

4.69.1.1 conn_hdl . . . 219
4.69.1.2 status . . . 219
4.70 LE_SMP_MSG_OOB_DATA_REQUEST_IND_T Struct Reference . . . 219
4.70.1 Field Documentation . . . 219
4.70.1.1 conn_hdl . . . 220
4.71 LE_SMP_MSG_PAIRING_ACTION_IND_T Struct Reference . . . 220
4.71.1 Field Documentation . . . 220
4.71.1.1 action . . . 220
4.71.1.2 conn_hdl . . . 220
4.71.1.3 lost_bond . . . 220
4.71.1.4 sc . . . 220
4.72 LE_SMP_MSG_PAIRING_COMPLETE_IND_T Struct Reference . . . 221
4.72.1 Field Documentation . . . 221
4.72.1.1 authenticated . . . 221
4.72.1.2 bonded . . . 221
4.72.1.3 conn_hdl . . . 221
4.72.1.4 peer_id_addr . . . 221
4.72.1.5 sc . . . 221
4.72.1.6 status . . . 222
4.73 LE_SMP_MSG_PASSKEY_DISPLAY_IND_T Struct Reference . . . 222
4.73.1 Field Documentation . . . 222
4.73.1.1 conn_hdl . . . 222
4.73.1.2 passkey . . . 222
4.74 LE_SMP_MSG_PASSKEY_INPUT_IND_T Struct Reference . . . 222
4.74.1 Field Documentation . . . 222
4.74.1.1 conn_hdl . . . 223
4.75 LE_SMP_MSG_SC_OOB_DATA_REQUEST_IND_T Struct Reference . . . 223
4.75.1 Field Documentation . . . 223
4.75.1.1 conn_hdl . . . 223
4.76 LE_SMP_MSG_SLAVE_SECURITY_REQUEST_IND_T Struct Reference . . . 223

4.76.1 Field Documentation . . . 223
4.76.1.1 bondable . . . 224
4.76.1.2 conn_hdl . . . 224
4.76.1.3 keypress . . . 224
4.76.1.4 mitm . . . 224
4.76.1.5 sc . . . 224
4.77 LE_SMP_MSG_USER_CONFIRM_IND_T Struct Reference . . . 224
4.77.1 Field Documentation . . . 224
4.77.1.1 confirm_num . . . 225
4.77.1.2 conn_hdl . . . 225
4.78 LE_SMP_SC_OOB_DATA_T Struct Reference . . . 225
4.78.1 Field Documentation . . . 225
4.78.1.1 confirm . . . 225
4.78.1.2 rand . . . 225
4.79 LE_SYS_MSG_BUF_OVERFLOW_T Struct Reference . . . 225
4.79.1 Field Documentation . . . 226
4.79.1.1 conn_hdl . . . 226
4.80 mw_blewifi_cbs_store_t Struct Reference . . . 226
4.80.1 Field Documentation . . . 226
4.80.1.1 manufacture_name . . . 226
4.81 mw_wifi_auto_connect_ap_info_t Struct Reference . . . 226
4.81.1 Field Documentation . . . 227
4.81.1.1 ap_channel . . . 227
4.81.1.2 beacon_interval . . . 227
4.81.1.3 bssid . . . 227
4.81.1.4 capabilities . . . 227
4.81.1.5 dtim_prod . . . 228
4.81.1.6 fast_connect . . . 228
4.81.1.7 free_ocpy . . . 228
4.81.1.8 hid_ssid . . . 228

4.81.1.9 hid_ssid_len . . . 228
4.81.1.10 latest_beacon_rx_time . . . 228
4.81.1.11 passphrase . . . 228
4.81.1.12 psk . . . 228
4.81.1.13 rsn_ie . . . 229
4.81.1.14 rssi . . . 229
4.81.1.15 ssid . . . 229
4.81.1.16 ssid_len . . . 229
4.81.1.17 supported_rates . . . 229
4.81.1.18 wpa_data . . . 229
4.81.1.19 wpa_ie . . . 229
4.82 mw_wifi_sta_info_t Struct Reference . . . 229
4.82.1 Field Documentation . . . 230
4.82.1.1 au8Dot11MACAddress . . . 230
4.82.1.2 u8SkipDtimPeriods . . . 230
4.83 MwFimAutoConnectCFG_t Struct Reference . . . 230
4.83.1 Field Documentation . . . 230
4.83.1.1 flag . . . 230
4.83.1.2 front . . . 231
4.83.1.3 max_save_num . . . 231
4.83.1.4 rear . . . 231
4.83.1.5 targetIdx . . . 231
4.84 rx_eapol_data Struct Reference . . . 231
4.84.1 Field Documentation . . . 231
4.84.1.1 frame_buffer . . . 231
4.84.1.2 frame_length . . . 232
4.85 S_WIFI_MLME_SCAN_CFG Struct Reference . . . 232
4.85.1 Detailed Description . . . 232
4.85.2 Field Documentation . . . 232
4.85.2.1 ptScanReport . . . 232

4.85.2.2 tScanType . . . 232
4.85.2.3 u32ActiveScanDur . . . 232
4.85.2.4 u32PassiveScanDur . . . 233
4.85.2.5 u8aBssid . . . 233
4.85.2.6 u8aSsid . . . 233
4.85.2.7 u8Channel . . . 233
4.85.2.8 u8MaxScanApNum . . . 233
4.85.2.9 u8ResendCnt . . . 233
4.86 scan_info_t Struct Reference . . . 233
4.86.1 Field Documentation . . . 234
4.86.1.1 ap_channel . . . 234
4.86.1.2 beacon_interval . . . 234
4.86.1.3 bssid . . . 234
4.86.1.4 capabilities . . . 234
4.86.1.5 dtim_prod . . . 235
4.86.1.6 free_ocpy . . . 235
4.86.1.7 latest_beacon_rx_time . . . 235
4.86.1.8 rsn_ie . . . 235
4.86.1.9 rssi . . . 235
4.86.1.10 ssid . . . 235
4.86.1.11 ssid_len . . . 235
4.86.1.12 supported_rates . . . 235
4.86.1.13 wpa_data . . . 236
4.86.1.14 wpa_ie . . . 236
4.87 scan_report_t Struct Reference . . . 236
4.87.1 Field Documentation . . . 236
4.87.1.1 pScanInfo . . . 236
4.87.1.2 uScanApNum . . . 236
4.88 T_RfCmd Struct Reference . . . 236
4.88.1 Field Documentation . . . 237

4.88.1.1 iArgc . . . 237
4.88.1.2 saArgv . . . 237
4.88.1.3 u32Type . . . 237
4.89 T_RfDefEvt Struct Reference . . . 237
4.89.1 Field Documentation . . . 237
4.89.1.1 u32Type . . . 237
4.89.1.2 u8aData . . . 238
4.89.1.3 u8Status . . . 238
4.90 T_RfEvt Struct Reference . . . 238
4.90.1 Field Documentation . . . 238
4.90.1.1 i8Rssi . . . 238
4.90.1.2 pParam . . . 239
4.90.1.3 u16RfMode . . . 239
4.90.1.4 u16RxCnt . . . 239
4.90.1.5 u16RxCrcOkCnt . . . 239
4.90.1.6 u32Freq . . . 239
4.90.1.7 u32Mode . . . 239
4.90.1.8 u32RfChannel . . . 239
4.90.1.9 u32Type . . . 239
4.90.1.10 u8Freq . . . 240
4.90.1.11 u8IpcEnable . . . 240
4.90.1.12 u8Len . . . 240
4.90.1.13 u8Phy . . . 240
4.90.1.14 u8Pkt . . . 240
4.90.1.15 u8Reserved . . . 240
4.90.1.16 u8Status . . . 240
4.90.1.17 u8Unicast . . . 240
4.91 wifi_active_scan_time_t Struct Reference . . . 241
4.91.1 Detailed Description . . . 241
4.91.2 Field Documentation . . . 241

4.91.2.1 max . . . 241
4.91.2.2 min . . . 241
4.92 wifi_ap_config_t Struct Reference . . . 241
4.92.1 Detailed Description . . . 242
4.92.2 Field Documentation . . . 242
4.92.2.1 auth_mode . . . 242
4.92.2.2 beacon_interval . . . 242
4.92.2.3 channel . . . 242
4.92.2.4 encrypt_type . . . 242
4.92.2.5 max_connection . . . 243
4.92.2.6 password . . . 243
4.92.2.7 password_length . . . 243
4.92.2.8 ssid . . . 243
4.92.2.9 ssid_hidden . . . 243
4.92.2.10 ssid_length . . . 243
4.93 wifi_auto_connect_info_t Struct Reference . . . 243
4.93.1 Detailed Description . . . 244
4.93.2 Field Documentation . . . 244
4.93.2.1 ap_channel . . . 244
4.93.2.2 beacon_interval . . . 244
4.93.2.3 bssid . . . 244
4.93.2.4 capabilities . . . 244
4.93.2.5 dtim_prod . . . 245
4.93.2.6 fast_connect . . . 245
4.93.2.7 hid_ssid . . . 245
4.93.2.8 rssi . . . 245
4.93.2.9 ssid . . . 245
4.93.2.10 supported_rates . . . 245
4.94 wifi_cmd_t Struct Reference . . . 245
4.94.1 Field Documentation . . . 246

4.94.1.1 arg1 . . . 246
4.94.1.2 arg2 . . . 246
4.94.1.3 cmd_type . . . 246
4.94.1.4 prvData . . . 246
4.95 wifi_config_t Union Reference . . . 246
4.95.1 Detailed Description . . . 246
4.95.2 Field Documentation . . . 247
4.95.2.1 ap_config . . . 247
4.95.2.2 sta_config . . . 247
4.96 wifi_event_info_t Union Reference . . . 247
4.96.1 Detailed Description . . . 247
4.96.2 Field Documentation . . . 247
4.96.2.1 connected . . . 247
4.96.2.2 disconnected . . . 248
4.96.2.3 got_ip . . . 248
4.96.2.4 scan_done . . . 248
4.97 wifi_event_sta_connected_t Struct Reference . . . 248
4.97.1 Detailed Description . . . 248
4.97.2 Field Documentation . . . 248
4.97.2.1 authmode . . . 248
4.97.2.2 bssid . . . 249
4.97.2.3 channel . . . 249
4.97.2.4 ssid . . . 249
4.97.2.5 ssid_len . . . 249
4.98 wifi_event_sta_disconnected_t Struct Reference . . . 249
4.98.1 Detailed Description . . . 249
4.98.2 Field Documentation . . . 249
4.98.2.1 bssid . . . 250
4.98.2.2 reason . . . 250
4.98.2.3 ssid . . . 250

4.98.2.4 ssid_len . . . 250
4.99 wifi_event_sta_got_ip_t Struct Reference . . . 250
4.99.1 Field Documentation . . . 250
4.99.1.1 ip_changed . . . 250
4.100 wifi_event_sta_scan_done_t Struct Reference . . . 251
4.100.1 Detailed Description . . . 251
4.100.2 Field Documentation . . . 251
4.100.2.1 number . . . 251
4.100.2.2 scan_id . . . 251
4.100.2.3 status . . . 251
4.101 wifi_evt_t Struct Reference . . . 251
4.101.1 Field Documentation . . . 252
4.101.1.1 evt_type . . . 252
4.101.1.2 prvData . . . 252
4.102 wifi_fast_scan_threshold_t Struct Reference . . . 252
4.102.1 Detailed Description . . . 252
4.102.2 Field Documentation . . . 252
4.102.2.1 authmode . . . 252
4.102.2.2 rssi . . . 253
4.103 wifi_init_config_t Struct Reference . . . 253
4.103.1 Detailed Description . . . 253
4.103.2 Field Documentation . . . 253
4.103.2.1 event_handler . . . 253
4.103.2.2 magic . . . 253
4.104 wifi_scan_config_t Struct Reference . . . 253
4.104.1 Detailed Description . . . 254
4.104.2 Field Documentation . . . 254
4.104.2.1 bssid . . . 254
4.104.2.2 channel . . . 254
4.104.2.3 scan_time . . . 254

4.104.2.4 scan_type . . . 254
4.104.2.5 show_hidden . . . 254
4.104.2.6 ssid . . . 255
4.105 wifi_scan_info_t Struct Reference . . . 255
4.105.1 Detailed Description . . . 255
4.105.2 Field Documentation . . . 255
4.105.2.1 auth_mode . . . 255
4.105.2.2 beacon_interval . . . 255
4.105.2.3 bssid . . . 256
4.105.2.4 capability_info . . . 256
4.105.2.5 channel . . . 256
4.105.2.6 dtim_period . . . 256
4.105.2.7 group_cipher . . . 256
4.105.2.8 pairwise_cipher . . . 256
4.105.2.9 rssi . . . 256
4.105.2.10 ssid . . . 257
4.105.2.11 ssid_length . . . 257
4.106 wifi_scan_list_t Struct Reference . . . 257
4.106.1 Detailed Description . . . 257
4.106.2 Field Documentation . . . 257
4.106.2.1 ap_record . . . 257
4.106.2.2 num . . . 257
4.107 wifi_scan_time_t Union Reference . . . 258
4.107.1 Detailed Description . . . 258
4.107.2 Field Documentation . . . 258
4.107.2.1 active . . . 258
4.107.2.2 passive . . . 258
4.108 wifi_sta_config_t Struct Reference . . . 258
4.108.1 Detailed Description . . . 259
4.108.2 Field Documentation . . . 259

4.108.2.1 bssid . . . 259
4.108.2.2 bssid_present . . . 259
4.108.2.3 password . . . 259
4.108.2.4 password_length . . . 259
4.108.2.5 scan_method . . . 259
4.108.2.6 sort_method . . . 259
4.108.2.7 ssid . . . 260
4.108.2.8 ssid_length . . . 260
4.108.2.9 threshold . . . 260
4.109 wifi_wpa_ie_data_t Struct Reference . . . 260
4.109.1 Detailed Description . . . 260
4.109.2 Field Documentation . . . 260
4.109.2.1 capabilities . . . 261
4.109.2.2 group_cipher . . . 261
4.109.2.3 key_mgmt . . . 261
4.109.2.4 mgmt_group_cipher . . . 261
4.109.2.5 num_pmkid . . . 261
4.109.2.6 pairwise_cipher . . . 261
4.109.2.7 pmkid . . . 261
4.109.2.8 proto . . . 261

Index **263**

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

- BLE ALL APIs . . . 7
  - BLE CM APIs . . . 8
  - BLE GAP APIs . . . 15
  - BLE GATT APIs . . . 37
  - BLE MSG APIs . . . 71
  - BLE SMP APIs . . . 83
- WIFI APIs . . . 91
  - WIFI Common APIs . . . 97
  - WIFI STA APIs . . . 100
  - Enumeration . . . 145

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

_wpa_ie_data . . . 151
asso_data . . . 152
auto_conn_info_t . . . 154
auto_connect_cfg_t . . . 157
event_msg_t
    Send information to event by event_msg_t . . . 159
hap_control_t . . . 160
LE_BT_ADDR_T . . . 161
LE_CM_CONNECTION_COMPLETE_IND_T . . . 162
LE_CM_MSG_ADVERTISE_REPORT_IND_T . . . 163
LE_CM_MSG_CONN_PARA_REQ_T . . . 164
LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T . . . 165
LE_CM_MSG_DATA_LEN_CHANGE_IND_T . . . 166
LE_CM_MSG_DIRECT_ADV_REPORT_IND_T . . . 167
LE_CM_MSG_DISCONNECT_COMPLETE_IND_T . . . 168
LE_CM_MSG_ENCRYPTION_CHANGE_IND_T . . . 169
LE_CM_MSG_ENCRYPTION_REFRESH_IND_T . . . 170
LE_CM_MSG_INIT_COMPLETE_CFM_T . . . 171
LE_CM_MSG_LTK_REQ_IND_T . . . 171
LE_CM_MSG_READ_ADV_TX_POWER_CFM_T . . . 172
LE_CM_MSG_READ_BD_ADDR_CFM_T . . . 173
LE_CM_MSG_READ_CHANNEL_MAP_CFM_T . . . 174
LE_CM_MSG_READ_PHY_CFM_T . . . 174
LE_CM_MSG_READ_RESOLVING_LIST_SIZE_CFM_T . . . 175
LE_CM_MSG_READ_RSSI_CFM_T . . . 176
LE_CM_MSG_READ_TX_POWER_CFM_T . . . 176
LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM_T . . . 177
LE_CM_MSG_SET_DATA_LENGTH_CFM_T . . . 178
LE_CM_MSG_SET_DISCONNECT_CFM_T . . . 178
LE_CM_MSG_SET_PHY_CFM_T . . . 179
LE_CM_MSG_SIGNAL_UPDATE_REQ_T . . . 180
LE_CM_REQ_STATUS_T . . . 181
LE_CONN_PARA_T . . . 181
LE_GAP_ADVERTISING_PARAM_T . . . 182
LE_GAP_CONN_PARAM_T . . . 183

LE_GAP_SCAN_PARAM_T . . . 184
LE_GATT_ATTR_T . . . 185
LE_GATT_MSG_ACCESS_READ_IND_T . . . 187
LE_GATT_MSG_ACCESS_WRITE_IND_T . . . 187
LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND_T . . . 189
LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND_T . . . 190
LE_GATT_MSG_CHARACTERISTIC_VAL_IND_T . . . 191
LE_GATT_MSG_CONFIRMATION_CFM_T . . . 193
LE_GATT_MSG_EXCHANGE_MTU_CFM_T . . . 193
LE_GATT_MSG_EXCHANGE_MTU_IND_T . . . 194
LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM_T . . . 195
LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM_T . . . 196
LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_CFM_T . . . 197
LE_GATT_MSG_FIND_CHARACTERISTIC_CFM_T . . . 198
LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM_T . . . 199
LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM_T . . . 200
LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T . . . 201
LE_GATT_MSG_INDICATE_IND_T . . . 202
LE_GATT_MSG_NOTIFY_CFM_T . . . 203
LE_GATT_MSG_NOTIFY_IND_T . . . 204
LE_GATT_MSG_OPERATION_TIMEOUT_T . . . 205
LE_GATT_MSG_PREPARE_WRITE_RELIABLE_CFM_T . . . 206
LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_CFM_T . . . 207
LE_GATT_MSG_READ_CHARACTERISTIC_VALUE_CFM_T . . . 208
LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM_T . . . 209
LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL_CFM_T . . . 210
LE_GATT_MSG_SERVICE_INFO_IND_T . . . 211
LE_GATT_MSG_SIGNED_WRITE_CFM_T . . . 212
LE_GATT_MSG_WRITE_CHAR_VAL_RELIABLE_CFM_T . . . 213
LE_GATT_MSG_WRITE_CHAR_VALUE_CFM_T . . . 214
LE_GATT_MSG_WRITE_LONG_CHAR_VALUE_CFM_T . . . 215
LE_GATT_MSG_WRITE_NO_RSP_CFM_T . . . 216
LE_GATT_SERVICE_T . . . 217
LE_SMP_MSG_ENCRYPTION_CHANGE_IND_T . . . 218
LE_SMP_MSG_ENCRYPTION_REFRESH_IND_T . . . 219
LE_SMP_MSG_OOB_DATA_REQUEST_IND_T . . . 219
LE_SMP_MSG_PAIRING_ACTION_IND_T . . . 220
LE_SMP_MSG_PAIRING_COMPLETE_IND_T . . . 221
LE_SMP_MSG_PASSKEY_DISPLAY_IND_T . . . 222
LE_SMP_MSG_PASSKEY_INPUT_IND_T . . . 222
LE_SMP_MSG_SC_OOB_DATA_REQUEST_IND_T . . . 223
LE_SMP_MSG_SLAVE_SECURITY_REQUEST_IND_T . . . 223
LE_SMP_MSG_USER_CONFIRM_IND_T . . . 224
LE_SMP_SC_OOB_DATA_T . . . 225
LE_SYS_MSG_BUF_OVERFLOW_T . . . 225
mw_blewifi_cbs_store_t . . . 226
mw_wifi_auto_connect_ap_info_t . . . 226
mw_wifi_sta_info_t . . . 229
MwFimAutoConnectCFG_t . . . 230
rx_eapol_data . . . 231
S_WIFI_MLME_SCAN_CFG . . . 232
scan_info_t . . . 233
scan_report_t . . . 236
T_RfCmd . . . 236
T_RfDefEvt . . . 237
T_RfEvt . . . 238
wifi_active_scan_time_t
Range of active scan times per channel . . . 241

wifi_ap_config_t
This structure is the Wi-Fi configuration for initialization for Soft-AP mode . . . 241
wifi_auto_connect_info_t
This structure is the Wi-Fi auto connect for save in the flash (FIM) . . . 243
wifi_cmd_t . . . 245
wifi_config_t
Wi-Fi configuration for initialization . . . 246
wifi_event_info_t
Wifi_event_info_t . . . 247
wifi_event_sta_connected_t
Wifi_event_sta_connected_t . . . 248
wifi_event_sta_disconnected_t
Wifi_event_sta_disconnected_t . . . 249
wifi_event_sta_got_ip_t . . . 250
wifi_event_sta_scan_done_t
Wifi_event_sta_scan_done_t . . . 251
wifi_evt_t . . . 251
wifi_fast_scan_threshold_t
Structure describing parameters for a Wi-Fi fast scan . . . 252
wifi_init_config_t
WiFi stack configuration parameters . . . 253
wifi_scan_config_t
Parameters for an SSID scan . . . 253
wifi_scan_info_t
This structure defines the inforamtion of scanned APs . . . 255
wifi_scan_list_t
This structure defines the list of scanned APs with their corresponding information . . . 257
wifi_scan_time_t
Aggregate of active & passive scan time per channel . . . 258
wifi_sta_config_t
This structure is the Wi-Fi configuration for initialization for STA mode . . . 258
wifi_wpa_ie_data_t
This structure is the Wi-Fi auto connect with wpa information for save in the flash (FIM) . . . 260

Chapter 3

Module Documentation

3.1 BLE ALL APIs

BLE ALL APIs.

Modules

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

Functions

- UINT8 LeSmpGetBondIdFromAddr (LE_BT_ADDR_T *peer_addr)

3.1.1 Detailed Description

BLE ALL APIs.

3.1.2 Function Documentation

3.1.2.1 LeSmpGetBondIdFromAddr()

```
UINT8 LeSmpGetBondIdFromAddr (
            LE_BT_ADDR_T * peer_addr )
```

3.2 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_PHY_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_SET_PHY_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_MSG_READ_PHY_CFM_T LE_CM_MSG_PHY_UPDATE_COMPLETE_IND_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_DEFAULT_PHY_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_CF
  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_
  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_WHI
  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_RESOLVIN
  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_CHAN
  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_READ_PHY_CFM, LE_CM_MSG_SET_DEFAULT_PHY_CFM,
  LE_CM_MSG_SET_PHY_CFM,
  LE_CM_MSG_PHY_UPDATE_COMPLETE_IND, LE_CM_MSG_TOP }

  BLE connection management message id.

Functions

- void LeCmInit (TASK appTask)

  BLE Connection Management Module Init.

3.2.1 Detailed Description

3.2.2 Typedef Documentation

3.2.2.1 LE_CM_MSG_ADD_TO_RESOLVING_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_ADD_TO_RESOLVING_LIST_CFM_T
```

3.2.2.2 LE_CM_MSG_ADD_TO_WHITE_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_ADD_TO_WHITE_LIST_CFM_T
```

3.2.2.3 LE_CM_MSG_CANCEL_CONNECTION_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_CANCEL_CONNECTION_CFM_T
```

3.2.2.4 LE_CM_MSG_CLEAR_RESOLVING_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_CLEAR_RESOLVING_LIST_CFM_T
```

3.2.2.5 LE_CM_MSG_CLEAR_WHITE_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_CLEAR_WHITE_LIST_CFM_T
```

3.2.2.6 LE_CM_MSG_CREATE_CONNECTION_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_CREATE_CONNECTION_CFM_T
```

3.2.2.7 LE_CM_MSG_ENTER_ADVERTISING_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_ENTER_ADVERTISING_CFM_T
```

3.2.2.8 LE_CM_MSG_ENTER_SCANNING_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_ENTER_SCANNING_CFM_T
```

3.2.2.9 LE_CM_MSG_EXIT_ADVERTISING_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_EXIT_ADVERTISING_CFM_T
```

3.2.2.10 LE_CM_MSG_EXIT_SCANNING_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_EXIT_SCANNING_CFM_T
```

3.2.2.11 LE_CM_MSG_PHY_UPDATE_COMPLETE_IND_T

```
typedef LE_CM_MSG_READ_PHY_CFM_T LE_CM_MSG_PHY_UPDATE_COMPLETE_IND_T
```

3.2.2.12 LE_CM_MSG_REMOVE_FROM_RESOLVING_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_REMOVE_FROM_RESOLVING_LIST_CFM_T
```

3.2.2.13 LE_CM_MSG_REMOVE_FROM_WHITE_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_REMOVE_FROM_WHITE_LIST_CFM_T
```

3.2.2.14 LE_CM_MSG_SET_ADVERTISING_DATA_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_ADVERTISING_DATA_CFM_T
```

3.2.2.15 LE_CM_MSG_SET_ADVERTISING_PARAMS_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_ADVERTISING_PARAMS_CFM_T
```

3.2.2.16 LE_CM_MSG_SET_CHANNEL_MAP_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_CHANNEL_MAP_CFM_T
```

3.2.2.17 LE_CM_MSG_SET_DEFAULT_PHY_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_DEFAULT_PHY_CFM_T
```

3.2.2.18 LE_CM_MSG_SET_RANDOM_ADDRESS_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_RANDOM_ADDRESS_CFM_T
```

3.2.2.19 LE_CM_MSG_SET_RPA_TIMEOUT_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_RPA_TIMEOUT_CFM_T
```

3.2.2.20 LE_CM_MSG_SET_SCAN_PARAMS_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_SCAN_PARAMS_CFM_T
```

3.2.2.21 LE_CM_MSG_SET_SCAN_RSP_DATA_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_SCAN_RSP_DATA_CFM_T
```

3.2.3 Enumeration Type Documentation

3.2.3.1 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

Enumerator

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
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_READ_PHY_CFM	
LE_CM_MSG_SET_DEFAULT_PHY_CFM	
LE_CM_MSG_SET_PHY_CFM	
LE_CM_MSG_PHY_UPDATE_COMPLETE_IND	
LE_CM_MSG_TOP	top of CM message id

3.2.4 Function Documentation

3.2.4.1 LeCmInit()

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

BLE Connection Management Module Init.

Parameters

the	reference of BLE task.

Returns

None.

3.3 BLE GAP APIs

Data Structures

- struct LE_GAP_ADVERTISING_PARAM_T
- struct LE_GAP_CONN_PARAM_T
- struct LE_GAP_SCAN_PARAM_T

Macros

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

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

Functions

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

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

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

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

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

  Central set data channel.
- LE_ERR_STATE LeGapClearResolvingList (void)

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

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

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

  Connection parameters request response.
- LE_ERR_STATE 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.
- LE_ERR_STATE LeGapReadPhy (UINT16 conn_hdl)
- 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 LeGapSetDefaultPhy (UINT8 tx, UINT8 rx)
- LE_ERR_STATE LeGapSetPhy (UINT16 conn_hdl, UINT8 tx, UINT8 rx, UINT16 option)
- 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.

3.3.1 Detailed Description

3.3.2 Macro Definition Documentation

3.3.2.1 GAP_ADTYPE_128BIT_COMPLETE

```
#define GAP_ADTYPE_128BIT_COMPLETE 0x07
```

3.3.2.2 GAP_ADTYPE_128BIT_MORE

```
#define GAP_ADTYPE_128BIT_MORE 0x06
```

3.3.2.3 GAP_ADTYPE_16BIT_COMPLETE

```
#define GAP_ADTYPE_16BIT_COMPLETE 0x03
```

3.3.2.4 GAP_ADTYPE_16BIT_MORE

```
#define GAP_ADTYPE_16BIT_MORE 0x02
```

3.3.2.5 GAP_ADTYPE_32BIT_COMPLETE

```
#define GAP_ADTYPE_32BIT_COMPLETE 0x05
```

3.3.2.6 GAP_ADTYPE_32BIT_MORE

```
#define GAP_ADTYPE_32BIT_MORE 0x04
```

3.3.2.7 GAP_ADTYPE_3D_INFO_DATA

```
#define GAP_ADTYPE_3D_INFO_DATA 0x3D
```

3.3.2.8 GAP_ADTYPE_ADV_INTERVAL

```
#define GAP_ADTYPE_ADV_INTERVAL 0x1A
```

3.3.2.9 GAP_ADTYPE_APPEARANCE

```
#define GAP_ADTYPE_APPEARANCE 0x19
```

3.3.2.10 GAP_ADTYPE_FLAGS

```
#define GAP_ADTYPE_FLAGS 0x01
```

3.3.2.11 GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED

```
#define GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED 0x04
```

3.3.2.12 GAP_ADTYPE_FLAGS_GENERAL

```
#define GAP_ADTYPE_FLAGS_GENERAL 0x02
```

3.3.2.13 GAP_ADTYPE_FLAGS_LIMITED

```
#define GAP_ADTYPE_FLAGS_LIMITED 0x01
```

3.3.2.14 GAP_ADTYPE_LE_BD_ADDR

```
#define GAP_ADTYPE_LE_BD_ADDR 0x1B
```

3.3.2.15 GAP_ADTYPE_LE_ROLE

```
#define GAP_ADTYPE_LE_ROLE 0x1C
```

3.3.2.16 GAP_ADTYPE_LOCAL_NAME_COMPLETE

```
#define GAP_ADTYPE_LOCAL_NAME_COMPLETE 0x09
```

3.3.2.17 GAP_ADTYPE_LOCAL_NAME_SHORT

```
#define GAP_ADTYPE_LOCAL_NAME_SHORT 0x08
```

3.3.2.18 GAP_ADTYPE_MANUFACTURER_SPECIFIC

```
#define GAP_ADTYPE_MANUFACTURER_SPECIFIC 0xFF
```

3.3.2.19 GAP_ADTYPE_OOB_CLASS_OF_DEVICE

```
#define GAP_ADTYPE_OOB_CLASS_OF_DEVICE 0x0D
```

3.3.2.20 GAP_ADTYPE_OOB_SIMPLE_PAIRING_HASHC

```
#define GAP_ADTYPE_OOB_SIMPLE_PAIRING_HASHC 0x0E
```

3.3.2.21 GAP_ADTYPE_OOB_SIMPLE_PAIRING_RANDR

```
#define GAP_ADTYPE_OOB_SIMPLE_PAIRING_RANDR 0x0F
```

3.3.2.22 GAP_ADTYPE_POWER_LEVEL

```
#define GAP_ADTYPE_POWER_LEVEL 0x0A
```

3.3.2.23 GAP_ADTYPE_PUBLIC_TARGET_ADDR

```
#define GAP_ADTYPE_PUBLIC_TARGET_ADDR 0x17
```

3.3.2.24 GAP_ADTYPE_RANDOM_TARGET_ADDR

```
#define GAP_ADTYPE_RANDOM_TARGET_ADDR 0x18
```

3.3.2.25 GAP_ADTYPE_SERVICE_DATA

```
#define GAP_ADTYPE_SERVICE_DATA 0x16
```

3.3.2.26 GAP_ADTYPE_SERVICE_DATA_128BIT

```
#define GAP_ADTYPE_SERVICE_DATA_128BIT 0x21
```

3.3.2.27 GAP_ADTYPE_SERVICE_DATA_32BIT

```
#define GAP_ADTYPE_SERVICE_DATA_32BIT 0x20
```

3.3.2.28 GAP_ADTYPE_SERVICES_LIST_128BIT

```
#define GAP_ADTYPE_SERVICES_LIST_128BIT 0x15
```

3.3.2.29 GAP_ADTYPE_SERVICES_LIST_16BIT

```
#define GAP_ADTYPE_SERVICES_LIST_16BIT 0x14
```

3.3.2.30 GAP_ADTYPE_SIGNED_DATA

```
#define GAP_ADTYPE_SIGNED_DATA 0x13
```

3.3.2.31 GAP_ADTYPE_SIMPLE_PAIRING_HASHC_256

```
#define GAP_ADTYPE_SIMPLE_PAIRING_HASHC_256 0x1D
```

3.3.2.32 GAP_ADTYPE_SIMPLE_PAIRING_RANDR_256

```
#define GAP_ADTYPE_SIMPLE_PAIRING_RANDR_256 0x1E
```

3.3.2.33 GAP_ADTYPE_SLAVE_CONN_INTERVAL_RANGE

```
#define GAP_ADTYPE_SLAVE_CONN_INTERVAL_RANGE 0x12
```

3.3.2.34 GAP_ADTYPE_SM_OOB_FLAG

```
#define GAP_ADTYPE_SM_OOB_FLAG 0x11
```

3.3.2.35 GAP_ADTYPE_SM_TK

```
#define GAP_ADTYPE_SM_TK 0x10
```

3.3.2.36 GAP_PUBLIC_ADDR

```
#define GAP_PUBLIC_ADDR 0
```

3.3.2.37 GAP_RAND_ADDR_NRPA

```
#define GAP_RAND_ADDR_NRPA 2
```

3.3.2.38 GAP_RAND_ADDR_RPA

```
#define GAP_RAND_ADDR_RPA 3
```

3.3.2.39 GAP_RAND_ADDR_STATIC

```
#define GAP_RAND_ADDR_STATIC 1
```

3.3.2.40 GAP_SCAN_TYPE_ACTIVE

```
#define GAP_SCAN_TYPE_ACTIVE 1
```

3.3.2.41 GAP_SCAN_TYPE_PASSIVE

```
#define GAP_SCAN_TYPE_PASSIVE 0
```

3.3.2.42 GAP_TX_PWR_CURR_VAL

```
#define GAP_TX_PWR_CURR_VAL 0
```

3.3.2.43 GAP_TX_PWR_MAX_VAL

```
#define GAP_TX_PWR_MAX_VAL 1
```

3.3.2.44 GAPBOND_IO_CAP_DISPLAY_ONLY

```
#define GAPBOND_IO_CAP_DISPLAY_ONLY 0x00
```

3.3.2.45 GAPBOND_IO_CAP_DISPLAY_YES_NO

```
#define GAPBOND_IO_CAP_DISPLAY_YES_NO 0x01
```

3.3.2.46 GAPBOND_IO_CAP_KEYBOARD_DISPLAY

```
#define GAPBOND_IO_CAP_KEYBOARD_DISPLAY 0x04
```

3.3.2.47 GAPBOND_IO_CAP_KEYBOARD_ONLY

```
#define GAPBOND_IO_CAP_KEYBOARD_ONLY 0x02
```

3.3.2.48 GAPBOND_IO_CAP_NO_INPUT_NO_OUTPUT

```
#define GAPBOND_IO_CAP_NO_INPUT_NO_OUTPUT 0x03
```

3.3.2.49 GAPBOND_PAIRING_MODE_INITIATE

```
#define GAPBOND_PAIRING_MODE_INITIATE 0x02
```

3.3.2.50 GAPBOND_PAIRING_MODE_NO_PAIRING

```
#define GAPBOND_PAIRING_MODE_NO_PAIRING 0x00
```

3.3.2.51 GAPBOND_PAIRING_MODE_WAIT_FOR_REQ

```
#define GAPBOND_PAIRING_MODE_WAIT_FOR_REQ 0x01
```

3.3.2.52 LE_GAP_ADV_MAX_SIZE

```
#define LE_GAP_ADV_MAX_SIZE 31
```

3.3.3 Function Documentation

3.3.3.1 LeGapAddToResolvingList()

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

Add device to resolving-list.

Parameters

bt_addr	BT device address.
irk	IRK, Identity Resolving Key

Returns

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

3.3.3.2 LeGapAddToWhiteList()

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

Add device to whitelist.

Parameters

bt_addr	BT device address.

Returns

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

3.3.3.3 LeGapAdvertisingEnable()

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

Enable or disable advertising function.

Parameters

start	TRUE is enable, FALSE is disable.

Returns

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

3.3.3.4 LeGapCentralConnectReq()

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

Central connect request.

Parameters

taddr	advertisers device address.
own_addr_type	owner address type.

Returns

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

3.3.3.5 LeGapCentralSetDataChannel()

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

Central set data channel.

Parameters

ch	data channel.

Returns

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

3.3.3.6 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.

3.3.3.7 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.

3.3.3.8 LeGapConnectCancelReq()

```
LE_ERR_STATE LeGapConnectCancelReq (
            void  )
```

Cancel connect request.

Returns

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

3.3.3.9 LeGapConnParaRequestRsp()

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

Connection parameters request response.

Parameters

conn_hdl	connection handle.
accept	TRUE is accept, FALSE is not.

Returns

None.

3.3.3.10 LeGapConnUpdateRequest()

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

Connection parameters update request.

Parameters

conn_hdl	connection handle.
para	update connection parameters.

Returns

None.

3.3.3.11 LeGapConnUpdateResponse()

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

Connection parameters update response.

Parameters

conn_hdl	connection handle.
identifier	TBD
accept	accept request, or not.

Returns

None.

3.3.3.12 LeGapDisconnectReq()

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

Disconnect the physical connection.

Parameters

conn_hdl	connection handle.

Returns

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

3.3.3.13 LeGapGenRandAddr()

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

Called to generation random address.

Parameters

type	address type.
addr	address.

Returns

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

3.3.3.14 LeGapGetBtAddr()

```
void LeGapGetBtAddr (
             void  )
```

Get owner device address.

3.3.3.15 LeGapReadAdvChannelTxPower()

```
void LeGapReadAdvChannelTxPower (
            void  )
```

Read ADV channel txpower.

3.3.3.16 LeGapReadChannelMap()

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

Read channel map.

Parameters

conn_hdl	connection handle.

Returns

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

3.3.3.17 LeGapReadPhy()

```
LE_ERR_STATE LeGapReadPhy (
            UINT16 conn_hdl )
```

3.3.3.18 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.

3.3.3.19 LeGapReadRssi()

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

Read RSSI value from controller.

Parameters

conn_hdl	connection handle.

Returns

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

3.3.3.20 LeGapReadTxPower()

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

Read tx power value for the specified connection.

Parameters

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

Returns

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

3.3.3.21 LeGapReadWhiteListSize()

```
void LeGapReadWhiteListSize (
             void  )
```

Read whitelist size in the controller.

3.3.3.22 LeGapRemoveFromWhiteList()

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

Remove device from whitelist.

Remove device from resolving-list.

Parameters

bt_addr	BT device address.

Returns

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

3.3.3.23 LeGapScanningReq()

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

Request scanning start.

Parameters

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

Returns

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

3.3.3.24 LeGapSetAdvData()

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

Called to set ADV data.

Parameters

len	ADV data length.
data	ADV data.

Returns

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

3.3.3.25 LeGapSetAdvParameter()

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

Called to set ADV parameters.

Parameters

params	advertising params.

Returns

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

3.3.3.26 LeGapSetConnParameter()

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

Called to set connection parameters.

Parameters

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

Returns

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

3.3.3.27 LeGapSetDataChannelPduLen()

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

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

Set data channel PDU length.

Parameters

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

Returns

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

3.3.3.28 LeGapSetDefaultPhy()

```
LE_ERR_STATE LeGapSetDefaultPhy (
            UINT8 tx,
            UINT8 rx )
```

3.3.3.29 LeGapSetPhy()

```
LE_ERR_STATE LeGapSetPhy (
            UINT16 conn_hdl,
            UINT8 tx,
            UINT8 rx,
            UINT16 option )
```

3.3.3.30 LeGapSetRandAddr()

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

Called to set random address.

Parameters

addr	the random address which should be set.

Returns

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

3.3.3.31 LeGapSetRpaTimeout()

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

Set resolvable private address timeout.

Parameters

timeout	RPA_Timeout, measured in seconds.

Returns

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

3.3.3.32 LeGapSetStaticAddr()

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

Called to set static address.

Parameters

addr	the static address which should be set.

Returns

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

3.3.3.33 LeSetScanParameter()

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

Called to set scan parameters.

Parameters

params	scan parameters.

Returns

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

3.3.3.34 LeSetScanRspData()

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

Called to set scan response data.

Parameters

len	scan response data length.
data	scan response data.

Returns

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

3.4 BLE GATT APIs

Data Structures

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

Macros

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

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

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

- #define LE_GATT_PERMIT_ENCRYPT_READ (0x0010)
- #define LE_GATT_PERMIT_ENCRYPT_WRITE (0x0020)
- #define LE_GATT_PERMIT_READ (0x0001)
- #define LE_GATT_PERMIT_READABLE (LE_GATT_PERMIT_READ | LE_GATT_PERMIT_AUTHEN_READ | LE_GATT_PERMIT_AUTHOR_READ | LE_GATT_PERMIT_ENCRYPT_READ | LE_GATT_PERMIT_SC_AUTHEN_READ)
- #define LE_GATT_PERMIT_SC_AUTHEN_READ (0x0100)
- #define LE_GATT_PERMIT_SC_AUTHEN_WRITE (0x0200)
- #define LE_GATT_PERMIT_WRITABLE (LE_GATT_PERMIT_WRITE | LE_GATT_PERMIT_AUTHEN_WRITE | LE_GATT_PERMIT_AUTHOR_WRITE | LE_GATT_PERMIT_ENCRYPT_WRITE | LE_GATT_PERMIT_SC_AUTHEN_WRIT
- #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_EXCHANG
  LE_GATT_MSG_ACCESS_READ_IND,
  LE_GATT_MSG_ACCESS_WRITE_IND, LE_GATT_MSG_SERVICE_INFO_IND, LE_GATT_MSG_FIND_ALL_PRIMARY_SE
  LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM,
  LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND, 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_TOP }

  BLE GATT message id.

Functions

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

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

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

  Change attribute value.
- LE_ERR_STATE LeGattCharValConfirmation (UINT16 conn_hdl)

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

  Gatt characteristic value indication.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  Called to register a service.

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

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

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

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

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

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

  Write without response.

Variables

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

3.4.1 Detailed Description

3.4.2 Macro Definition Documentation

3.4.2.1 CHAR_AGGREGATE_DESCRIPTOR

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

3.4.2.2 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)}
```

3.4.2.3 CHAR_DECL_UUID16_ATTR_VAL

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

3.4.2.4 CHAR_EXT_PROP_DESCRIPTOR

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

3.4.2.5 CHAR_PRESENT_FORMAT_DESCRIPTOR

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

3.4.2.6 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)}
```

3.4.2.7 CHAR_USER_DESC_DESCRIPTOR

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

3.4.2.8 CHARACTERISTIC_DECL_UUID128

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

3.4.2.9 CHARACTERISTIC_DECL_UUID16

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

3.4.2.10 CHARACTERISTIC_UUID128

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

3.4.2.11 CHARACTERISTIC_UUID16

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

3.4.2.12 GATT_CHAR_AGG_FORMAT_UUID

```
#define GATT_CHAR_AGG_FORMAT_UUID 0x2905
```

3.4.2.13 GATT_CHAR_EXT_PROPS_UUID

```
#define GATT_CHAR_EXT_PROPS_UUID 0x2900
```

3.4.2.14 GATT_CHAR_FORMAT_UUID

```
#define GATT_CHAR_FORMAT_UUID 0x2904
```

3.4.2.15 GATT_CHAR_USER_DESC_UUID

```
#define GATT_CHAR_USER_DESC_UUID 0x2901
```

3.4.2.16 GATT_CHARACTERISTIC_UUID

```
#define GATT_CHARACTERISTIC_UUID 0x2803
```

3.4.2.17 GATT_CLIENT_CHAR_CFG_UUID

```
#define GATT_CLIENT_CHAR_CFG_UUID 0x2902
```

3.4.2.18 GATT_EXT_REPORT_REF_UUID

```
#define GATT_EXT_REPORT_REF_UUID 0x2907
```

3.4.2.19 GATT_INCLUDE_UUID

```
#define GATT_INCLUDE_UUID 0x2802
```

3.4.2.20 GATT_PRIMARY_SERVICE_UUID

```
#define GATT_PRIMARY_SERVICE_UUID 0x2800
```

3.4.2.21 GATT_REPORT_REF_UUID

```
#define GATT_REPORT_REF_UUID 0x2908
```

3.4.2.22 GATT_SECONDARY_SERVICE_UUID

```
#define GATT_SECONDARY_SERVICE_UUID 0x2801
```

3.4.2.23 GATT_SERV_CHAR_CFG_UUID

```
#define GATT_SERV_CHAR_CFG_UUID 0x2903
```

3.4.2.24 GATT_VALID_RANGE_UUID

```
#define GATT_VALID_RANGE_UUID 0x2906
```

3.4.2.25 INCLUDE_DECL_UUID128

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

3.4.2.26 INCLUDE_DECL_UUID128_ATTR_VAL

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

3.4.2.27 INCLUDE_DECL_UUID16_ATTR_VAL

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

3.4.2.28 INCLUDE_DECL_UUINT16

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

3.4.2.29 LE_ATT_UUID_SIZE

```
#define LE_ATT_UUID_SIZE 2
```

3.4.2.30 LE_GATT_CHAR_PROP_AUTH

```
#define LE_GATT_CHAR_PROP_AUTH 0x40
```

3.4.2.31 LE_GATT_CHAR_PROP_BCAST

```
#define LE_GATT_CHAR_PROP_BCAST 0x01
```

Characteristic Properties Bit.

3.4.2.32 LE_GATT_CHAR_PROP_EXT_PROP

```
#define LE_GATT_CHAR_PROP_EXT_PROP 0x80
```

3.4.2.33 LE_GATT_CHAR_PROP_IND

```
#define LE_GATT_CHAR_PROP_IND 0x20
```

3.4.2.34 LE_GATT_CHAR_PROP_NTF

```
#define LE_GATT_CHAR_PROP_NTF 0x10
```

3.4.2.35 LE_GATT_CHAR_PROP_RD

```
#define LE_GATT_CHAR_PROP_RD 0x02
```

3.4.2.36 LE_GATT_CHAR_PROP_WR

```
#define LE_GATT_CHAR_PROP_WR 0x08
```

3.4.2.37 LE_GATT_CHAR_PROP_WR_NO_RESP

```
#define LE_GATT_CHAR_PROP_WR_NO_RESP 0x04
```

3.4.2.38 LE_GATT_CLIENT_CFG_INDICATION

```
#define LE_GATT_CLIENT_CFG_INDICATION 0x02
```

3.4.2.39 LE_GATT_CLIENT_CFG_NOTIFICATION

```
#define LE_GATT_CLIENT_CFG_NOTIFICATION 0x01
```

3.4.2.40 LE_GATT_EXT_PROP_RELIABLE_WR

```
#define LE_GATT_EXT_PROP_RELIABLE_WR 0x0001
```

3.4.2.41 LE_GATT_EXT_PROP_WR_AUX

```
#define LE_GATT_EXT_PROP_WR_AUX 0x0002
```

3.4.2.42 LE_GATT_FLAG_PREPARE_WRITE

```
#define LE_GATT_FLAG_PREPARE_WRITE 0x02
```

3.4.2.43 LE_GATT_FLAG_WRITE_CMD

```
#define LE_GATT_FLAG_WRITE_CMD 0x01
```

3.4.2.44 LE_GATT_FLAG_WRITE_REQ

```
#define LE_GATT_FLAG_WRITE_REQ 0x00
```

3.4.2.45 LE_GATT_PERM_AUTH_READABLE

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

3.4.2.46 LE_GATT_PERM_AUTH_WRITABLE

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

3.4.2.47 LE_GATT_PERM_NONE

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

3.4.2.48 LE_GATT_PERM_READ

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

3.4.2.49 LE_GATT_PERM_RELIABLE_WRITE

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

3.4.2.50 LE_GATT_PERM_WRITE_CMD

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

3.4.2.51 LE_GATT_PERM_WRITE_REQ

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

3.4.2.52 LE_GATT_PERMIT_AUTHEN_READ

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

3.4.2.53 LE_GATT_PERMIT_AUTHEN_WRITE

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

3.4.2.54 LE_GATT_PERMIT_AUTHOR_READ

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

3.4.2.55 LE_GATT_PERMIT_AUTHOR_WRITE

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

3.4.2.56 LE_GATT_PERMIT_ENCRYPT_READ

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

3.4.2.57 LE_GATT_PERMIT_ENCRYPT_WRITE

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

3.4.2.58 LE_GATT_PERMIT_READ

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

3.4.2.59 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)
```

3.4.2.60 LE_GATT_PERMIT_SC_AUTHEN_READ

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

3.4.2.61 LE_GATT_PERMIT_SC_AUTHEN_WRITE

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

3.4.2.62 LE_GATT_PERMIT_WRITABLE

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

3.4.2.63 LE_GATT_PERMIT_WRITE

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

3.4.2.64 PRIMARY_SERVICE_DECL_UUID128

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

3.4.2.65 PRIMARY_SERVICE_DECL_UUID16

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

3.4.2.66 SECONDARY_SERVICE_DECL_UUID128

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

3.4.2.67 SECONDARY_SERVICE_DECL_UUID16

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

3.4.3 Enumeration Type Documentation

3.4.3.1 anonymous enum

```
anonymous enum
```

BLE GATT message id.

Enumerator

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

Enumerator

LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_↩ CFM	execute write reliable confirm
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_TOP	top of GATT message id

3.4.4 Function Documentation

3.4.4.1 LeGattAccessReadRsp()

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

Gatt access read response.

Parameters

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

Returns

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

3.4.4.2 LeGattAccessWriteRsp()

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

Gatt access write response.

Parameters

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

Returns

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

3.4.4.3 LeGattChangeAttrVal()

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

Change attribute value.

Parameters

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

Returns

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

3.4.4.4 LeGattCharValConfirmation()

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

Prepare write characteristic value response.

Parameters

conn_hdl	connection handle.

Returns

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

3.4.4.5 LeGattCharValIndicate()

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

Gatt characteristic value indication.

Parameters

conn_hdl	connection handle.
hdl	characteristic value handle.
len	value length.
pval	value.

Returns

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

3.4.4.6 LeGattCharValNotify()

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

Gatt characteristic value notification.

Parameters

conn_hdl	connection handle.
hdl	characteristic value handle.
len	value length.
pval	value.

Returns

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

3.4.4.7 LeGattExchangeMtuReq()

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

Exchange MTU request.

Parameters

conn_hdl	connection handle.
mtu	MTU.

Returns

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

3.4.4.8 LeGattExchangeMtuRsp()

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

Exchange MTU response.

Parameters

conn_hdl	connection handle.
mtu	MTU.

Returns

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

3.4.4.9 LeGattExecuteWriteCharValReliable()

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

Execute write characteristic value request.

Parameters

conn_hdl	connection handle.
yesno	execute write or not.

Returns

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

3.4.4.10 LeGattFindAllCharacteristic()

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

Find all characteristic.

Parameters

conn_hdl	connection handle.
start_hdl	start handle.
end_hdl	end handle.

Returns

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

3.4.4.11 LeGattFindAllCharDescriptor()

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

Find all characteristic description.

Parameters

conn_hdl	connection handle.
start_hdl	start handle.
end_hdl	end handle.

Returns

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

3.4.4.12 LeGattFindAllPrimaryService()

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

Find all primary service.

Parameters

conn_hdl	connection handle.

Returns

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

3.4.4.13 LeGattFindCharacteristicByUuid()

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

Find characteristic by UUID.

Parameters

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

Returns

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

3.4.4.14 LeGattFindIncludedService()

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

Find include service.

Parameters

conn_hdl	connection handle.
start_hdl	start handle.
end_hdl	end handle.

Returns

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

3.4.4.15 LeGattFindPrimaryServiceByUuid()

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

Find primary service by UUID.

Parameters

conn_hdl	connection handle.
format	UUID type.
uuid	UUID.

Returns

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

3.4.4.16 LeGattGetAttrHandle()

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

Get attribute handle.

Parameters

svc	service.
attr↩ Id	attribute index of service.

Returns

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

3.4.4.17 LeGattGetAttrVal()

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

Get attribute value.

Parameters

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

Returns

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

3.4.4.18 LeGattGetAttrValLen()

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

Get the length of attribute value.

Parameters

svc	service.
attr↩ Id	attribute index of service.

Returns

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

3.4.4.19 LeGattGetAttrValMaxLen()

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

Get the max length of attribute value.

Parameters

svc	service.
attr↩ Id	attribute index of service.

Returns

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

3.4.4.20 LeGattInit()

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

BLE Gatt module init.

Parameters

appTask	the reference of BLE task.

Returns

None.

3.4.4.21 LeGattModifyAttrVal()

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

Modify attribute value.

Parameters

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

Returns

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

3.4.4.22 LeGattPrepareWriteCharValReliable()

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

Prepare write characteristic value request.

Parameters

conn_hdl	connection handle.
handle	characteristic value handle.
offset	offset written.
len	length written.
val	value.

Returns

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

3.4.4.23 LeGattReadCharValByUuid()

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

Read a characteristic value by UUID.

Parameters

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

Returns

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

3.4.4.24 LeGattReadCharValue()

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

Read a characteristic value.

Parameters

conn_hdl	connection handle.
handle	characteristic value handle.

Returns

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

3.4.4.25 LeGattReadLongCharVal()

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

Read a long characteristic value.

Parameters

conn_hdl	connection handle.
handle	characteristic value handle.
offset	characteristic value offset.

Returns

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

3.4.4.26 LeGattReadMultipleCharVal()

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

Read Multiple characteristic values.

Parameters

conn_hdl	connection handle.
count	handle count.
handle	handle table.

Returns

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

3.4.4.27 LeGattRegisterIncludeService()

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

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

Called to register an include service.

Parameters

inc_hdl	include service handle.
start_hdl	start handle.
end_hdl	end handle.
uuid	include service UUID.

Returns

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

3.4.4.28 LeGattRegisterService()

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

Called to register a service.

Parameters

attrTable	service attribute table.
numAttr	the attribute number of service.

Returns

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

3.4.4.29 LeGattSignedWriteNoRsp()

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

Signed write without response.

Parameters

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

Returns

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

3.4.4.30 LeGattStopCurrentProcedure()

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

Stop current procedure.

Parameters

conn_hdl	connection handle.

Returns

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

3.4.4.31 LeGattWriteCharVal()

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

Write characteristic value.

Parameters

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

Returns

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

3.4.4.32 LeGattWriteCharValReliable()

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

Write characteristic value reliable.

Parameters

conn_hdl	connection handle.
handle	characteristic value handle.
offset	offset written.
len	length written.
val	value.

Returns

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

3.4.4.33 LeGattWriteLongCharVal()

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

Write long characteristic value.

Parameters

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

Returns

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

3.4.4.34 LeGattWriteNoRsp()

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

Write without response.

Parameters

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

Returns

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

3.4.5 Variable Documentation

3.4.5.1 gcCharacteristicUuid

```
const UINT16 gcCharacteristicUuid
```

3.4.5.2 gcCharAggregateUuid

```
const UINT16 gcCharAggregateUuid
```

3.4.5.3 gcCharExtPropUuid

```
const UINT16 gcCharExtPropUuid
```

3.4.5.4 gcCharFormatUuid

```
const UINT16 gcCharFormatUuid
```

3.4.5.5 gcCharUserDescUuid

```
const UINT16 gcCharUserDescUuid
```

3.4.5.6 gcClientCharConfigUuid

```
const UINT16 gcClientCharConfigUuid
```

3.4.5.7 gcExtReportRefUuid

```
const UINT16 gcExtReportRefUuid
```

3.4.5.8 gcIncludeUuid

```
const UINT16 gcIncludeUuid
```

3.4.5.9 gcPrimaryServiceUuid

```
const UINT16 gcPrimaryServiceUuid
```

3.4.5.10 gcReportRefUuid

```
const UINT16 gcReportRefUuid
```

3.4.5.11 gcSecondaryServiceUuid

```
const UINT16 gcSecondaryServiceUuid
```

3.4.5.12 gcServerCharConfigUuid

```
const UINT16 gcServerCharConfigUuid
```

3.4.5.13 gcValidRangeUuid

```
const UINT16 gcValidRangeUuid
```

3.5 BLE MSG APIs

Data Structures

- struct LE_SYS_MSG_BUF_OVERFLOW_T

Macros

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

Typedefs

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

Enumerations

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

  BLE system message id.

Functions

- UINT16 LeCancelAllMessage (TASK task, MESSAGEID id)

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

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

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

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

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

  Create BLE task.
- void LeHostMessageLoop (void)

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

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

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

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

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

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

  Send sub message until lock is 0.

3.5.1 Detailed Description

3.5.2 Macro Definition Documentation

3.5.2.1 LE_ATT_MSG_BASE

```
#define LE_ATT_MSG_BASE 0x1400
```

3.5.2.2 LE_CM_MSG_BASE

```
#define LE_CM_MSG_BASE 0x1100
```

3.5.2.3 LE_GATT_MSG_BASE

```
#define LE_GATT_MSG_BASE 0x1500
```

3.5.2.4 LE_HCI_MSG_BASE

```
#define LE_HCI_MSG_BASE 0x1000
```

3.5.2.5 LE_L2CAP_MSG_BASE

```
#define LE_L2CAP_MSG_BASE 0x1200
```

3.5.2.6 LE_SMP_MSG_BASE

```
#define LE_SMP_MSG_BASE 0x1300
```

3.5.2.7 LE_SYS_MSG_BASE

```
#define LE_SYS_MSG_BASE 0x8000
```

3.5.2.8 MESSAGE_ALLOCATE

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

3.5.2.9 MESSAGE_BULID

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

3.5.2.10 MESSAGE_DATA_BULID

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

3.5.2.11 MESSAGE_OFFSET

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

3.5.2.12 T_HOUR

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

3.5.2.13 T_MIN

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

3.5.2.14 T_SEC

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

3.5.3 Typedef Documentation

3.5.3.1 MESSAGE

```
typedef MsgData MESSAGE
```

3.5.3.2 MESSAGEID

```
typedef UINT16 MESSAGEID
```

3.5.3.3 MsgData

```
typedef void const* MsgData
```

3.5.3.4 MsgLock

```
typedef const UINT8* MsgLock
```

3.5.3.5 MSGLOCK

```
typedef MsgLock MSGLOCK
```

3.5.3.6 MSGSUBID

```
typedef UINT16 MSGSUBID
```

3.5.3.7 MSGTIMER

```
typedef UINT32 MSGTIMER
```

3.5.3.8 Task

```
typedef TASKPACK* Task
```

3.5.3.9 TASK

```
typedef Task TASK
```

3.5.3.10 TASKHANDLER

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

3.5.3.11 TASKPACK

```
typedef void** TASKPACK
```

3.5.4 Enumeration Type Documentation

3.5.4.1 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

3.5.5 Function Documentation

3.5.5.1 LeCancelAllMessage()

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

Cancel all message in queue.

Parameters

task	task.
id	message id.

Returns

0 is ok, others is error.

3.5.5.2 LeCancelAllSubMessage()

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

Cancel all sub message in queue.

Parameters

task	the task of recvice message.
id	message id.
subId	sub message id.

Returns

0 is ok, others is error.

3.5.5.3 LeCancelFirstMessage()

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

Cancel the first message in queue.

Parameters

task	task.
id	message id.

Returns

True is ok, false is error.

3.5.5.4 LeCancelFirstSubMessage()

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

Cancel the first sub message in queue.

Parameters

task	the task of recvice message.
id	message id.
subId	sub message id.

Returns

True is ok, false is error.

3.5.5.5 LeGetSubMsgId()

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

Get sub message id.

Parameters

sub	message id.

Returns

0 is ok, others is error.

3.5.5.6 LeHostCreateTask()

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

Create BLE task.

Parameters

task	the reference of BLE task.
hdl	callback handle of BLE task.

Returns

TRUE is success, FALSE is failed.

3.5.5.7 LeHostMessageLoop()

```
void LeHostMessageLoop (
            void  )
```

message loop run.

Returns

None.

3.5.5.8 LeSendMessage()

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

Send message to BLE task.

Parameters

task	reference of BLE task.
msg↩ Id	message ID.
msg	message.

Returns

None.

3.5.5.9 LeSendMessageAfter()

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

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

Delay, then send message to BLE task.

Parameters

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

Returns

None.

3.5.5.10 LeSendMessageUnlock()

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

Send message until lock is 0.

Parameters

task	the task of recvice message.
id	message id.
msg	message.
lock	lock number.

Returns

None.

3.5.5.11 LeSendSubMessage()

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

Send sub message.

Parameters

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

Returns

None.

3.5.5.12 LeSendSubMessageAfter()

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

Delay, then send sub message.

Parameters

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

Returns

None.

3.5.5.13 LeSendSubMessageUnlock()

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

Send sub message until lock is 0.

Parameters

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

Returns

None.

3.6 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_CO
  LE_SMP_MSG_ENCRYPTION_CHANGE_IND,
  LE_SMP_MSG_ENCRYPTION_REFRESH_IND, LE_SMP_MSG_PAIRING_COMPLETE_IND, LE_SMP_LONG_TERM_KEY_
  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.

3.6.1 Detailed Description

3.6.2 Macro Definition Documentation

3.6.2.1 LE_MAX_BOND_COUNT

```
#define LE_MAX_BOND_COUNT 8
```

3.6.2.2 LE_SM_IO_CAP_DISP_ONLY

```
#define LE_SM_IO_CAP_DISP_ONLY 0x00
```

display only

3.6.2.3 LE_SM_IO_CAP_DISP_YES_NO

```
#define LE_SM_IO_CAP_DISP_YES_NO 0x01
```

display + yes or no

3.6.2.4 LE_SM_IO_CAP_KEYBOARD_DISP

```
#define LE_SM_IO_CAP_KEYBOARD_DISP 0x04
```

display + keyboard

3.6.2.5 LE_SM_IO_CAP_KEYBOARD_ONLY

```
#define LE_SM_IO_CAP_KEYBOARD_ONLY 0x02
```

keyboard only

3.6.2.6 LE_SM_IO_CAP_NO_IO

```
#define LE_SM_IO_CAP_NO_IO 0x03
```

no input and output

3.6.2.7 LE_SM_PAIR_MITM_NO

```
#define LE_SM_PAIR_MITM_NO 0x00
```

3.6.2.8 LE_SM_PAIR_MITM_YES

```
#define LE_SM_PAIR_MITM_YES 0x01
```

3.6.2.9 LE_SM_PAIR_OOB_NO

```
#define LE_SM_PAIR_OOB_NO 0x00
```

3.6.2.10 LE_SM_PAIR_OOB_YES

```
#define LE_SM_PAIR_OOB_YES 0x01
```

3.6.2.11 LE_SM_PAIR_SC_NO

```
#define LE_SM_PAIR_SC_NO 0x00
```

3.6.2.12 LE_SM_PAIR_SC_YES

```
#define LE_SM_PAIR_SC_YES 0x01
```

3.6.3 Enumeration Type Documentation

3.6.3.1 anonymous enum

```
anonymous enum
```

BLE SMP message id.

Enumerator

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

3.6.3.2 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

3.6.4 Function Documentation

3.6.4.1 LeSmpInit()

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

BLE SMP Module Init.

Parameters

appTask	the reference of BLE task.

Returns

None.

3.6.4.2 LeSmpOobAuthDataRsp()

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

SMP OOB authenticate data response.

Parameters

conn_hdl	connection handle.
data	response data.
len	data length.

Returns

None.

3.6.4.3 LeSmpOobPresent()

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

SMP OOB present.

Parameters

conn_hdl	connection handle.
oob_present	present or not.

Returns

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

3.6.4.4 LeSmpPasskeyInput()

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

Input passkey.

Parameters

conn_hdl	connection handle.
passkey	passkey.

Returns

None.

3.6.4.5 LeSmpScOobComputeConfirmVal()

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

SMP secure connection OOB compute confirm value.

Parameters

rand	random data.
confirm	confirm data.

Returns

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

3.6.4.6 LeSmpScOobDataRsp()

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

OOB data response.

Parameters

conn_hdl	connection handld.
our_rand	our random data.
peer	peer OOB data.

Returns

None.

3.6.4.7 LeSmpSecurityReq()

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

BLE SMP security request.

Parameters

conn_hdl	connection handle.

Returns

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

3.6.4.8 LeSmpSecurityRsp()

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

BLE SMP security request.

Parameters

conn_hdl	connection handle.
accept	TRUE is accept, FALSE is not.

Returns

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

3.6.4.9 LeSmpSetDefaultConfig()

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

Set default configure for pairing.

Parameters

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

Returns

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

3.6.4.10 LeSmpUserConfirmRsp()

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

User confirm response.

Parameters

conn_hdl	connection handle.
accept	yes or no.

Returns

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

3.7 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_t

  This structure is the Wi-Fi auto connect for save in the flash (FIM).
- union wifi_config_t

  Wi-Fi configuration for initialization.
- struct wifi_fast_scan_threshold_t

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

  WiFi stack configuration parameters.
- struct wifi_scan_config_t

  Parameters for an SSID scan.
- struct wifi_scan_info_t

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

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

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

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

  This structure is the Wi-Fi auto connect with wpa information for save in the flash (FIM).

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_MAC_NUM_OF_CHANNELS (14)

  maximum number of WIFI channels

- #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 data rate

Typedefs

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

Enumerations

- enum wifi_auto_connet_mode_e { WIFI_AUTO_CONNECT_DISABLE, WIFI_AUTO_CONNECT_ENABLE }

  WiFi auto connect mode parameters.

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)

  instaall default event handler for wifi event (internal use)

- int wifi_register_event_handler (wifi_event_t idx, wifi_event_handler_t handler)

  register wifi event handelrt (internal use)

3.7.1 Detailed Description

WIFI APIs.

3.7.2 Macro Definition Documentation

3.7.2.1 WIFI_BEACON_INTERVAL_LENGTH

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

Beacon interval length in a frame header.

3.7.2.2 WIFI_CAPABILITY_INFO_LENGTH

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

Length of capability information in a frame header.

3.7.2.3 WIFI_LENGTH_802_11

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

Length of 802.11 MAC header.

3.7.2.4 WIFI_LENGTH_PASSPHRASE

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

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

3.7.2.5 WIFI_MAC_ADDRESS_LENGTH

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

MAC address length.

3.7.2.6 WIFI_MAC_NUM_OF_CHANNELS

```
#define WIFI_MAC_NUM_OF_CHANNELS (14)
```

maximum number of WIFI channels

3.7.2.7 WIFI_MAX_LENGTH_OF_SSID

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

The maximum length of SSID.

3.7.2.8 WIFI_MAX_SCAN_AP_NUM

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

maximum number of ap list items which can stored

3.7.2.9 WIFI_MAX_SUPPORTED_RATES

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

maximum number of supported data rate

3.7.3 Typedef Documentation

3.7.3.1 wifi_ap_record_t

```
typedef wifi_scan_info_t wifi_ap_record_t
```

3.7.3.2 wifi_event_notify_cb_t

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

3.7.4 Enumeration Type Documentation

3.7.4.1 wifi_auto_connet_mode_e

```
enum wifi_auto_connet_mode_e
```

WiFi auto connect mode parameters.

Enumerator

WIFI_AUTO_CONNECT_DISABLE	
WIFI_AUTO_CONNECT_ENABLE	

3.7.5 Function Documentation

3.7.5.1 wifi_event_process_handler()

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

Default event handler for system events.

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

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

Parameters

in	*event*	event type Set the event type,Options are • WIFI_EVENT_INIT_COMPLETE • 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
in	*payload*	Data block that transmitted to event
in	*length*	The length of data block

Returns

0 : success
other : failed

3.7.5.2 wifi_install_default_event_handlers()

```
void wifi_install_default_event_handlers (
            void  )
```

instaall default event handler for wifi event (internal use)

3.7.5.3 wifi_register_event_handler()

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

register wifi event handelrt (internal use)

Parameters

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

Returns

0 : success
other : failed

3.8 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
- 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.

3.8.1 Detailed Description

3.8.2 Typedef Documentation

3.8.2.1 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.

3.8.3 Function Documentation

3.8.3.1 wifi_event_loop_init()

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

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

Parameters

cb	: application specified event callback

Returns

0 : success
other : failed

3.8.3.2 wifi_event_loop_send()

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

Send an event to event task.

Attention

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

Parameters

event_msg_t	* msg: Send information to event by msg

Returns

0 : success
other : failed

3.8.3.3 wifi_event_loop_set_cb()

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

Set application specified event callback function.

Attention

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

Parameters

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

3.8.3.4 wifi_event_process_handler()

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

Default event handler for system events.

This function performs default handling of system events.

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

Parameters

in	*event*	event type Set the event type,Options are • WIFI_EVENT_INIT_COMPLETE • 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
in	*payload*	Data block transmitted to event
in	*length*	The length of the data block

Returns

0 : success
other : failed

3.9 WIFI STA APIs

Macros

- #define WIFI_READY_TIME 2000

Typedefs

- typedef int(∗ wifi_auto_connect_clear_ap_info_fp_t) (uint8_t index)
- typedef int(∗ wifi_auto_connect_get_ap_info_fp_t) (uint8_t index, wifi_auto_connect_info_t ∗info)
- typedef int(∗ wifi_auto_connect_get_ap_num_fp_t) (uint8_t ∗num)
- typedef int(∗ wifi_auto_connect_get_mode_fp_t) (uint8_t ∗mode)
- typedef int(∗ wifi_auto_connect_get_saved_ap_num_fp_t) (uint8_t ∗num)
- typedef int(∗ wifi_auto_connect_init_fp_t) (void)
- typedef int(∗ wifi_auto_connect_reset_fp_t) (void)
- typedef int(∗ wifi_auto_connect_set_ap_num_fp_t) (uint8_t num)
- typedef int(∗ wifi_auto_connect_set_mode_fp_t) (uint8_t mode)
- typedef int(∗ wifi_auto_connect_start_fp_t) (void)
- typedef int(∗ wifi_auto_connect_update_ch_fp_t) (uint8_t ac_index, uint8_t channel)
- typedef int(∗ wifi_config_get_bandwidth_fp_t) (wifi_mode_t interface, wifi_bandwidth_t ∗bandwidth)
- typedef int(∗ wifi_config_get_bssid_fp_t) (uint8_t ∗bssid)
- typedef int(∗ wifi_config_get_channel_fp_t) (wifi_mode_t interface, uint8_t ∗channel)
- typedef int(∗ wifi_config_get_dtim_interval_fp_t) (uint8_t ∗interval)
- typedef int(∗ wifi_config_get_listen_interval_fp_t) (uint8_t ∗interval)
- typedef int(∗ wifi_config_get_mac_address_fp_t) (wifi_mode_t interface, uint8_t ∗address)
- typedef int(∗ wifi_config_get_mac_tx_data_rate_fp_t) (wifi_mac_data_rate_t ∗data_rate)
- typedef int(∗ wifi_config_get_opmode_fp_t) (uint8_t ∗mode)
- typedef int(∗ wifi_config_get_skip_dtim_fp_t) (uint8_t ∗value)
- typedef int(∗ wifi_config_get_ssid_fp_t) (uint8_t ∗ssid, uint8_t ∗ssid_length)
- typedef int(∗ wifi_config_get_sta_mac_address_from_flash_fp_t) (uint8_t ∗bssid)
- typedef int(∗ wifi_config_set_bandwidth_fp_t) (wifi_mode_t interface, wifi_bandwidth_t bandwidth)
- typedef int(∗ wifi_config_set_bssid_fp_t) (uint8_t ∗bssid)
- typedef int(∗ wifi_config_set_channel_fp_t) (wifi_mode_t interface, uint8_t channel)
- typedef int(∗ wifi_config_set_dtim_interval_fp_t) (uint8_t interval)
- typedef int(∗ wifi_config_set_listen_interval_fp_t) (uint8_t interval)
- typedef int(∗ wifi_config_set_mac_address_fp_t) (wifi_mode_t interface, uint8_t ∗address)
- typedef int(∗ wifi_config_set_mac_tx_data_rate_fp_t) (wifi_mac_data_rate_t data_rate)
- typedef int(∗ wifi_config_set_opmode_fp_t) (uint8_t mode)
- typedef int(∗ wifi_config_set_skip_dtim_fp_t) (uint8_t value, bool save_flash)
- typedef int(∗ wifi_config_set_ssid_fp_t) (wifi_mode_t interface, uint8_t ∗ssid, uint8_t ssid_length)
- typedef int(∗ wifi_connection_connect_fp_t) (wifi_config_t ∗config)
- typedef int(∗ wifi_connection_connect_from_ac_index_fp_t) (uint8_t index)
- typedef int(∗ wifi_connection_connect_from_ac_list_fp_t) (wifi_config_t ∗config)
- typedef int(∗ wifi_connection_disconnect_ap_fp_t) (void)
- typedef int(∗ wifi_connection_disconnect_sta_fp_t) (uint8_t ∗address)
- typedef int(∗ wifi_connection_get_rssi_fp_t) (int8_t ∗rssi)
- typedef int(∗ wifi_connection_register_event_handler_fp_t) (wifi_event_t event, wifi_event_handler_t handler)
- typedef int(∗ wifi_connection_scan_start_fp_t) (uint8_t ∗ssid, uint8_t ssid_length, uint8_t ∗bssid, uint8_↩ t scan_mode, uint8_t scan_option)
- typedef int(∗ wifi_connection_unregister_event_handler_fp_t) (wifi_event_t event, wifi_event_handler_t handler)
- typedef int(∗ wifi_convert_auth_mode_fp_t) (scan_info_t ∗pinfo, int privacy)
- typedef int(∗ wifi_deinit_fp_t) (void)

- 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 int(∗ wifi_fast_connect_get_mode_fp_t) (uint8_t ap_index, uint8_t ∗mode)
- typedef int(∗ wifi_fast_connect_set_mode_fp_t) (uint8_t ap_index, uint8_t mode)
- typedef int(∗ wifi_fast_connect_start_fp_t) (uint8_t ap_index)
- typedef int(∗ wifi_get_config_fp_t) (wifi_mode_t interface, wifi_config_t ∗conf)
- typedef void(∗ wifi_init_complete_cb_t) (void ∗ctx)

  Initialization of complete callback function.
- typedef int(∗ wifi_init_fp_t) (const wifi_init_config_t ∗config, wifi_init_complete_cb_t init_cb)
- typedef int32_t wifi_result_t
- typedef int(∗ wifi_scan_get_ap_list_fp_t) (wifi_scan_list_t ∗scan_list)
- typedef int(∗ wifi_scan_get_ap_num_fp_t) (uint16_t ∗number)
- typedef int(∗ wifi_scan_get_ap_records_fp_t) (uint16_t ∗number, wifi_scan_info_t ∗ap_records)
- typedef int(∗ wifi_scan_start_fp_t) (const wifi_scan_config_t ∗config, bool block)
- typedef int(∗ wifi_scan_stop_fp_t) (void)
- typedef int(∗ wifi_set_config_fp_t) (wifi_mode_t interface, wifi_config_t ∗conf)
- typedef int(∗ wifi_sta_get_ap_info_fp_t) (wifi_ap_record_t ∗ap_info)
- typedef int(∗ wifi_start_fp_t) (void)
- typedef int(∗ wifi_stop_fp_t) (void)

Functions

- int wifi_auto_connect_clear_ap_info (uint8_t index)

  Clear the AP information which index in the.
- int wifi_auto_connect_get_ap_info (uint8_t index, wifi_auto_connect_info_t ∗info)

  Get the AP information.
- int wifi_auto_connect_get_ap_num (uint8_t ∗num)

  Get the maximum number of AP information.
- int wifi_auto_connect_get_mode (uint8_t ∗mode)

  Get the auto connect mode.
- int wifi_auto_connect_get_saved_ap_num (uint8_t ∗num)

  Get the current number of AP save in flash.
- int wifi_auto_connect_init (void)

  Initialize function of auto connect.
- int wifi_auto_connect_reset (void)

  Reset all of auto/fast connect configuration.
- int wifi_auto_connect_set_ap_num (uint8_t num)

  Set the maximum number of AP information.
- int wifi_auto_connect_set_mode (uint8_t mode)

  Set the auto connect mode.
- int wifi_auto_connect_start (void)

  Start auto connect mechanism.
- int wifi_auto_connect_update_ch (uint8_t ac_index, uint8_t channel)

  Update the channel which AP index in auto connect list.
- 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_dtim_interval (uint8_t *interval)
- int wifi_config_get_listen_interval (uint8_t *interval)
- int wifi_config_get_mac_address (wifi_mode_t interface, uint8_t *address)

  Get mac of specified interface.
- int wifi_config_get_mac_tx_data_rate (wifi_mac_data_rate_t *data_rate)

  Get the Mac tx data rate in current wifi setting of OPL1000.
- int wifi_config_get_opmode (uint8_t *mode)
- int wifi_config_get_skip_dtim (uint8_t *value)

  Get the Skip DTIM value in current wifi setting of OPL1000.
- int wifi_config_get_ssid (uint8_t *ssid, uint8_t *ssid_length)

  Get ssid value of AP.
- int wifi_config_get_sta_mac_address_from_flash (uint8_t *bssid)

  Get mac address of station from flash.
- 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_dtim_interval (uint8_t interval)
- int wifi_config_set_listen_interval (uint8_t interval)
- 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_mac_tx_data_rate (wifi_mac_data_rate_t data_rate)

  Set the Mac tx data rate setting of OPL1000.
- int wifi_config_set_opmode (uint8_t mode)
- int wifi_config_set_skip_dtim (uint8_t value, bool save_flash)

  Set the Skip DTIM value of OPL1000.
- 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_connect_from_ac_index (uint8_t index)

  Connect OPL1000 Wi-Fi station to certain AP by auto connect index.
- int wifi_connection_connect_from_ac_list (wifi_config_t *config)

  Connect OPL1000 Wi-Fi station to certain AP in auto connect list.
- 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_scan_start (uint8_t *ssid, uint8_t ssid_length, uint8_t *bssid, uint8_t scan_mode, uint8↩ _t scan_option)
- int wifi_connection_unregister_event_handler (wifi_event_t event, wifi_event_handler_t handler)

  unregister wifi call back handler
- int wifi_convert_auth_mode (scan_info_t *pinfo, int privacy)
- int wifi_deinit (void)

  De-init Wi-Fi Initialization and Configuration functions.

- int wifi_fast_connect_get_mode (uint8_t ap_index, uint8_t *mode)

  Get the fast connect mode.
- int wifi_fast_connect_set_mode (uint8_t ap_index, uint8_t mode)

  Set the fast connect mode.
- int wifi_fast_connect_start (uint8_t ap_index)

  Start fast connect mechanism.
- int wifi_get_config (wifi_mode_t interface, wifi_config_t *conf)

  Get configuration of specified interface.
- 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_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_scan_stop (void)

  Stop scanning process.
- int wifi_set_config (wifi_mode_t interface, wifi_config_t *conf)

  Set configuration of OPL1000 STA.
- int wifi_sta_get_ap_info (wifi_ap_record_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.

Variables

- wifi_auto_connect_clear_ap_info_fp_t wifi_auto_connect_clear_ap_info_api
- wifi_auto_connect_get_ap_info_fp_t wifi_auto_connect_get_ap_info_api
- wifi_auto_connect_get_ap_num_fp_t wifi_auto_connect_get_ap_num_api
- wifi_auto_connect_get_mode_fp_t wifi_auto_connect_get_mode_api
- wifi_auto_connect_get_saved_ap_num_fp_t wifi_auto_connect_get_saved_ap_num_api
- wifi_auto_connect_init_fp_t wifi_auto_connect_init_api
- wifi_auto_connect_reset_fp_t wifi_auto_connect_reset_api
- wifi_auto_connect_set_ap_num_fp_t wifi_auto_connect_set_ap_num_api
- wifi_auto_connect_set_mode_fp_t wifi_auto_connect_set_mode_api
- wifi_auto_connect_start_fp_t wifi_auto_connect_start_api
- wifi_auto_connect_update_ch_fp_t wifi_auto_connect_update_ch_api
- wifi_config_get_bandwidth_fp_t wifi_config_get_bandwidth_api
- wifi_config_get_bssid_fp_t wifi_config_get_bssid_api
- wifi_config_get_channel_fp_t wifi_config_get_channel_api
- wifi_config_get_dtim_interval_fp_t wifi_config_get_dtim_interval_api
- wifi_config_get_listen_interval_fp_t wifi_config_get_listen_interval_api
- wifi_config_get_mac_address_fp_t wifi_config_get_mac_address_api
- wifi_config_get_mac_tx_data_rate_fp_t wifi_config_get_mac_tx_data_rate_api
- wifi_config_get_opmode_fp_t wifi_config_get_opmode_api
- wifi_config_get_skip_dtim_fp_t wifi_config_get_skip_dtim_api

- wifi_config_get_ssid_fp_t wifi_config_get_ssid_api
- wifi_config_get_sta_mac_address_from_flash_fp_t wifi_config_get_sta_mac_address_from_flash_api
- wifi_config_set_bandwidth_fp_t wifi_config_set_bandwidth_api
- wifi_config_set_bssid_fp_t wifi_config_set_bssid_api
- wifi_config_set_channel_fp_t wifi_config_set_channel_api
- wifi_config_set_dtim_interval_fp_t wifi_config_set_dtim_interval_api
- wifi_config_set_listen_interval_fp_t wifi_config_set_listen_interval_api
- wifi_config_set_mac_address_fp_t wifi_config_set_mac_address_api
- wifi_config_set_mac_tx_data_rate_fp_t wifi_config_set_mac_tx_data_rate_api
- wifi_config_set_opmode_fp_t wifi_config_set_opmode_api
- wifi_config_set_skip_dtim_fp_t wifi_config_set_skip_dtim_api
- wifi_config_set_ssid_fp_t wifi_config_set_ssid_api
- wifi_connection_connect_fp_t wifi_connection_connect_api
- wifi_connection_connect_from_ac_index_fp_t wifi_connection_connect_from_ac_index_api
- wifi_connection_connect_from_ac_list_fp_t wifi_connection_connect_from_ac_list_api
- wifi_connection_disconnect_ap_fp_t wifi_connection_disconnect_ap_api
- wifi_connection_disconnect_sta_fp_t wifi_connection_disconnect_sta_api
- wifi_connection_get_rssi_fp_t wifi_connection_get_rssi_api
- wifi_connection_register_event_handler_fp_t wifi_connection_register_event_handler_api
- wifi_connection_scan_start_fp_t wifi_connection_scan_start_api
- wifi_connection_unregister_event_handler_fp_t wifi_connection_unregister_event_handler_api
- wifi_convert_auth_mode_fp_t wifi_convert_auth_mode_api
- wifi_deinit_fp_t wifi_deinit_api
- wifi_fast_connect_get_mode_fp_t wifi_fast_connect_get_mode_api
- wifi_fast_connect_set_mode_fp_t wifi_fast_connect_set_mode_api
- wifi_fast_connect_start_fp_t wifi_fast_connect_start_api
- wifi_get_config_fp_t wifi_get_config_api
- wifi_init_fp_t wifi_init_api
- wifi_scan_get_ap_list_fp_t wifi_scan_get_ap_list_api
- wifi_scan_get_ap_num_fp_t wifi_scan_get_ap_num_api
- wifi_scan_get_ap_records_fp_t wifi_scan_get_ap_records_api
- wifi_scan_start_fp_t wifi_scan_start_api
- wifi_scan_stop_fp_t wifi_scan_stop_api
- wifi_set_config_fp_t wifi_set_config_api
- wifi_sta_get_ap_info_fp_t wifi_sta_get_ap_info_api
- wifi_start_fp_t wifi_start_api
- wifi_stop_fp_t wifi_stop_api

3.9.1 Detailed Description

3.9.2 Macro Definition Documentation

3.9.2.1 WIFI_READY_TIME

```
#define WIFI_READY_TIME 2000
```

3.9.3 Typedef Documentation

3.9.3.1 wifi_auto_connect_clear_ap_info_fp_t

```
typedef int(* wifi_auto_connect_clear_ap_info_fp_t) (uint8_t index)
```

3.9.3.2 wifi_auto_connect_get_ap_info_fp_t

```
typedef int(* wifi_auto_connect_get_ap_info_fp_t) (uint8_t index, wifi_auto_connect_info_t
*info)
```

3.9.3.3 wifi_auto_connect_get_ap_num_fp_t

```
typedef int(* wifi_auto_connect_get_ap_num_fp_t) (uint8_t *num)
```

3.9.3.4 wifi_auto_connect_get_mode_fp_t

```
typedef int(* wifi_auto_connect_get_mode_fp_t) (uint8_t *mode)
```

3.9.3.5 wifi_auto_connect_get_saved_ap_num_fp_t

```
typedef int(* wifi_auto_connect_get_saved_ap_num_fp_t) (uint8_t *num)
```

3.9.3.6 wifi_auto_connect_init_fp_t

```
typedef int(* wifi_auto_connect_init_fp_t) (void)
```

3.9.3.7 wifi_auto_connect_reset_fp_t

```
typedef int(* wifi_auto_connect_reset_fp_t) (void)
```

3.9.3.8 wifi_auto_connect_set_ap_num_fp_t

```
typedef int(* wifi_auto_connect_set_ap_num_fp_t) (uint8_t num)
```

3.9.3.9 wifi_auto_connect_set_mode_fp_t

```
typedef int(* wifi_auto_connect_set_mode_fp_t) (uint8_t mode)
```

3.9.3.10 wifi_auto_connect_start_fp_t

```
typedef int(* wifi_auto_connect_start_fp_t) (void)
```

3.9.3.11 wifi_auto_connect_update_ch_fp_t

```
typedef int(* wifi_auto_connect_update_ch_fp_t) (uint8_t ac_index, uint8_t channel)
```

3.9.3.12 wifi_config_get_bandwidth_fp_t

```
typedef int(* wifi_config_get_bandwidth_fp_t) (wifi_mode_t interface, wifi_bandwidth_t *bandwidth)
```

3.9.3.13 wifi_config_get_bssid_fp_t

```
typedef int(* wifi_config_get_bssid_fp_t) (uint8_t *bssid)
```

3.9.3.14 wifi_config_get_channel_fp_t

```
typedef int(* wifi_config_get_channel_fp_t) (wifi_mode_t interface, uint8_t *channel)
```

3.9.3.15 wifi_config_get_dtim_interval_fp_t

```
typedef int(* wifi_config_get_dtim_interval_fp_t) (uint8_t *interval)
```

3.9.3.16 wifi_config_get_listen_interval_fp_t

```
typedef int(* wifi_config_get_listen_interval_fp_t) (uint8_t *interval)
```

3.9.3.17 wifi_config_get_mac_address_fp_t

```
typedef int(* wifi_config_get_mac_address_fp_t) (wifi_mode_t interface, uint8_t *address)
```

3.9.3.18 wifi_config_get_mac_tx_data_rate_fp_t

```
typedef int(* wifi_config_get_mac_tx_data_rate_fp_t) (wifi_mac_data_rate_t *data_rate)
```

3.9.3.19 wifi_config_get_opmode_fp_t

```
typedef int(* wifi_config_get_opmode_fp_t) (uint8_t *mode)
```

3.9.3.20 wifi_config_get_skip_dtim_fp_t

```
typedef int(* wifi_config_get_skip_dtim_fp_t) (uint8_t *value)
```

3.9.3.21 wifi_config_get_ssid_fp_t

```
typedef int(* wifi_config_get_ssid_fp_t) (uint8_t *ssid, uint8_t *ssid_length)
```

3.9.3.22 wifi_config_get_sta_mac_address_from_flash_fp_t

```
typedef int(* wifi_config_get_sta_mac_address_from_flash_fp_t) (uint8_t *bssid)
```

3.9.3.23 wifi_config_set_bandwidth_fp_t

```
typedef int(* wifi_config_set_bandwidth_fp_t) (wifi_mode_t interface, wifi_bandwidth_t bandwidth)
```

3.9.3.24 wifi_config_set_bssid_fp_t

```
typedef int(* wifi_config_set_bssid_fp_t) (uint8_t *bssid)
```

3.9.3.25 wifi_config_set_channel_fp_t

```
typedef int(* wifi_config_set_channel_fp_t) (wifi_mode_t interface, uint8_t channel)
```

3.9.3.26 wifi_config_set_dtim_interval_fp_t

```
typedef int(* wifi_config_set_dtim_interval_fp_t) (uint8_t interval)
```

3.9.3.27 wifi_config_set_listen_interval_fp_t

```
typedef int(* wifi_config_set_listen_interval_fp_t) (uint8_t interval)
```

3.9.3.28 wifi_config_set_mac_address_fp_t

```
typedef int(* wifi_config_set_mac_address_fp_t) (wifi_mode_t interface, uint8_t *address)
```

3.9.3.29 wifi_config_set_mac_tx_data_rate_fp_t

```
typedef int(* wifi_config_set_mac_tx_data_rate_fp_t) (wifi_mac_data_rate_t data_rate)
```

3.9.3.30 wifi_config_set_opmode_fp_t

```
typedef int(* wifi_config_set_opmode_fp_t) (uint8_t mode)
```

3.9.3.31 wifi_config_set_skip_dtim_fp_t

```
typedef int(* wifi_config_set_skip_dtim_fp_t) (uint8_t value, bool save_flash)
```

3.9.3.32 wifi_config_set_ssid_fp_t

```
typedef int(* wifi_config_set_ssid_fp_t) (wifi_mode_t interface, uint8_t *ssid, uint8_t ssid_↩
length)
```

3.9.3.33 wifi_connection_connect_fp_t

```
typedef int(* wifi_connection_connect_fp_t) (wifi_config_t *config)
```

3.9.3.34 wifi_connection_connect_from_ac_index_fp_t

```
typedef int(* wifi_connection_connect_from_ac_index_fp_t) (uint8_t index)
```

3.9.3.35 wifi_connection_connect_from_ac_list_fp_t

```
typedef int(* wifi_connection_connect_from_ac_list_fp_t) (wifi_config_t *config)
```

3.9.3.36 wifi_connection_disconnect_ap_fp_t

```
typedef int(* wifi_connection_disconnect_ap_fp_t) (void)
```

3.9.3.37 wifi_connection_disconnect_sta_fp_t

```
typedef int(* wifi_connection_disconnect_sta_fp_t) (uint8_t *address)
```

3.9.3.38 wifi_connection_get_rssi_fp_t

```
typedef int(* wifi_connection_get_rssi_fp_t) (int8_t *rssi)
```

3.9.3.39 wifi_connection_register_event_handler_fp_t

```
typedef int(* wifi_connection_register_event_handler_fp_t) (wifi_event_t event, wifi_event_handler_t
handler)
```

3.9.3.40 wifi_connection_scan_start_fp_t

```
typedef int(* wifi_connection_scan_start_fp_t) (uint8_t *ssid, uint8_t ssid_length, uint8_↩
t *bssid, uint8_t scan_mode, uint8_t scan_option)
```

3.9.3.41 wifi_connection_unregister_event_handler_fp_t

```
typedef int(* wifi_connection_unregister_event_handler_fp_t) (wifi_event_t event, wifi_event_handler_t
handler)
```

3.9.3.42 wifi_convert_auth_mode_fp_t

```
typedef int(* wifi_convert_auth_mode_fp_t) (scan_info_t *pinfo, int privacy)
```

3.9.3.43 wifi_deinit_fp_t

```
typedef int(* wifi_deinit_fp_t) (void)
```

3.9.3.44 wifi_event_handler_t

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

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

Parameters

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

Returns

The return value is reserved and it is ignored.

3.9.3.45 wifi_fast_connect_get_mode_fp_t

```
typedef int(* wifi_fast_connect_get_mode_fp_t) (uint8_t ap_index, uint8_t *mode)
```

3.9.3.46 wifi_fast_connect_set_mode_fp_t

```
typedef int(* wifi_fast_connect_set_mode_fp_t) (uint8_t ap_index, uint8_t mode)
```

3.9.3.47 wifi_fast_connect_start_fp_t

```
typedef int(* wifi_fast_connect_start_fp_t) (uint8_t ap_index)
```

3.9.3.48 wifi_get_config_fp_t

```
typedef int(* wifi_get_config_fp_t) (wifi_mode_t interface, wifi_config_t *conf)
```

3.9.3.49 wifi_init_complete_cb_t

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

Initialization of complete callback function.

Invoked when Wi-Fi initialization is complete.

Parameters

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

3.9.3.50 wifi_init_fp_t

```
typedef int(* wifi_init_fp_t) (const wifi_init_config_t *config, wifi_init_complete_cb_t init↩
_cb)
```

3.9.3.51 wifi_result_t

```
typedef int32_t wifi_result_t
```

3.9.3.52 wifi_scan_get_ap_list_fp_t

```
typedef int(* wifi_scan_get_ap_list_fp_t) (wifi_scan_list_t *scan_list)
```

3.9.3.53 wifi_scan_get_ap_num_fp_t

```
typedef int(* wifi_scan_get_ap_num_fp_t) (uint16_t *number)
```

3.9.3.54 wifi_scan_get_ap_records_fp_t

```
typedef int(* wifi_scan_get_ap_records_fp_t) (uint16_t *number, wifi_scan_info_t *ap_records)
```

3.9.3.55 wifi_scan_start_fp_t

```
typedef int(* wifi_scan_start_fp_t) (const wifi_scan_config_t *config, bool block)
```

3.9.3.56 wifi_scan_stop_fp_t

```
typedef int(* wifi_scan_stop_fp_t) (void)
```

3.9.3.57 wifi_set_config_fp_t

```
typedef int(* wifi_set_config_fp_t) (wifi_mode_t interface, wifi_config_t *conf)
```

3.9.3.58 wifi_sta_get_ap_info_fp_t

```
typedef int(* wifi_sta_get_ap_info_fp_t) (wifi_ap_record_t *ap_info)
```

3.9.3.59 wifi_start_fp_t

```
typedef int(* wifi_start_fp_t) (void)
```

3.9.3.60 wifi_stop_fp_t

```
typedef int(* wifi_stop_fp_t) (void)
```

3.9.4 Function Documentation

3.9.4.1 wifi_auto_connect_clear_ap_info()

```
int wifi_auto_connect_clear_ap_info (
            uint8_t index )
```

Clear the AP information which index in the.

Attention

1. API returns false if try to clear AP information which something error

Parameters

in	*index*	The index of AP position • Range is 0 to 2

Returns

0 : success
other : failed

3.9.4.2 wifi_auto_connect_get_ap_info()

```
int wifi_auto_connect_get_ap_info (
            uint8_t index,
            wifi_auto_connect_info_t * info )
```

Get the AP information.

Attention

1. API returns false if try to get AP information which something error

Parameters

in	*index*	The index of AP position • Range is 0 to 2
out	*mode*	Get the AP information

Returns

0 : success
other : failed

3.9.4.3 wifi_auto_connect_get_ap_num()

```
int wifi_auto_connect_get_ap_num (
            uint8_t * num )
```

Get the maximum number of AP information.

Attention

1. API returns false if try to get maximum auto connect numbers which something error

Parameters

out	*num*	Get the maximum number of AP information

Returns

0 : success
other : failed

3.9.4.4 wifi_auto_connect_get_mode()

```
int wifi_auto_connect_get_mode (
            uint8_t * mode )
```

Get the auto connect mode.

Attention

1. API returns false if try to get auto connect mode which something error

Parameters

out	*mode*	Get the auto connect mode

Returns

0 : success
other : failed

3.9.4.5 wifi_auto_connect_get_saved_ap_num()

```
int wifi_auto_connect_get_saved_ap_num (
            uint8_t * num )
```

Get the current number of AP save in flash.

Attention

1. API returns false if try to get current auto connect numbers which something error

Parameters

out	*num*	The current number of AP information will be saved in flash.

Returns

0 : success
other : failed

3.9.4.6 wifi_auto_connect_init()

```
int wifi_auto_connect_init (
            void  )
```

Initialize function of auto connect.

Attention

1. API returns false if try to initial auto connect which something error

Returns

0 : success
other : failed

3.9.4.7 wifi_auto_connect_reset()

```
int wifi_auto_connect_reset (
            void  )
```

Reset all of auto/fast connect configuration.

Attention

1. API returns false if try to reset auto connect configuration which something error

Returns

0 : success
other : failed

3.9.4.8 wifi_auto_connect_set_ap_num()

```
int wifi_auto_connect_set_ap_num (
            uint8_t num )
```

Set the maximum number of AP information.

Attention

1. API returns false if try to set maximum auto connect numbers which something error

Parameters

in	*num*	The maximum number of AP information will be saved in flash. • Range is 1 to 3

Returns

0 : success
other : failed

3.9.4.9 wifi_auto_connect_set_mode()

```
int wifi_auto_connect_set_mode (
            uint8_t mode )
```

Set the auto connect mode.

Attention

1. API returns false if try to set auto connect mode which something error

Parameters

in	*mode*	Configure the current wifi working mode,The options are • WIFI_AUTO_CONNECT_ENABLE • WIFI_AUTO_CONNECT_DISABLE

Returns

0 : success
other : failed

3.9.4.10 wifi_auto_connect_start()

```
int wifi_auto_connect_start (
            void  )
```

Start auto connect mechanism.

Attention

1. API returns false if try to start auto connect function which something error

Returns

0 : success
other : failed

3.9.4.11 wifi_auto_connect_update_ch()

```
int wifi_auto_connect_update_ch (
            uint8_t ac_index,
            uint8_t channel )
```

Update the channel which AP index in auto connect list.

Attention

1. API returns false if update channel which something error

Parameters

in	*index*	The index of AP position • Range is 0 to 2
in	*channel*	The channel of AP's used. • Range is 1 to 14

Returns

0 : success
other : failed

3.9.4.12 wifi_config_get_bandwidth()

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

Get the bandwidth of OPL1000 specified interface.

Attention

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

Parameters

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

Returns

0 : success
other : failed

3.9.4.13 wifi_config_get_bssid()

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

get bssid after scan

Parameters

out	*bssid*	the string of bssid

Returns

0 : success
other : failed

3.9.4.14 wifi_config_get_channel()

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

Get the primary/secondary channel of OPL1000.

Attention

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

Parameters

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

Returns

0 : success
other : failed

3.9.4.15 wifi_config_get_dtim_interval()

```
int wifi_config_get_dtim_interval (
            uint8_t * interval )
```

3.9.4.16 wifi_config_get_listen_interval()

```
int wifi_config_get_listen_interval (
            uint8_t * interval )
```

3.9.4.17 wifi_config_get_mac_address()

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

Get mac of specified interface.

Parameters

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

Returns

0 : success
other : failed

3.9.4.18 wifi_config_get_mac_tx_data_rate()

```
int wifi_config_get_mac_tx_data_rate (
            wifi_mac_data_rate_t * data_rate )
```

Get the Mac tx data rate in current wifi setting of OPL1000.

Parameters

out	*data_rate*	Get the Mac tx data rate • WIFI_MAC_DATA_RATE_ARA • WIFI_MAC_DATA_RATE_1M • WIFI_MAC_DATA_RATE_2M • WIFI_MAC_DATA_RATE_5_5M • WIFI_MAC_DATA_RATE_11M

Returns

0 : success
other : failed

3.9.4.19 wifi_config_get_opmode()

```
int wifi_config_get_opmode (
            uint8_t * mode )
```

3.9.4.20 wifi_config_get_skip_dtim()

```
int wifi_config_get_skip_dtim (
            uint8_t * value )
```

Get the Skip DTIM value in current wifi setting of OPL1000.

Parameters

out	*value*	Get the Skip DTIM value in current wifi setting

Returns

0 : success
other : failed

3.9.4.21 wifi_config_get_ssid()

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

Get ssid value of AP.

Parameters

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

Returns

0 : success
other : failed

3.9.4.22 wifi_config_get_sta_mac_address_from_flash()

```
int wifi_config_get_sta_mac_address_from_flash (
            uint8_t * bssid )
```

Get mac address of station from flash.

Parameters

out	*address*	Get the MAC address of station from flash, The address is similar to this structure: xx:xx:xx:xx:xx:xx

Returns

0 : success
other : failed

3.9.4.23 wifi_config_set_bandwidth()

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

Set the bandwidth of OPL1000 specified interface.

Parameters

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

Returns

0 : success
other : failed

3.9.4.24 wifi_config_set_bssid()

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

config OPL1000 Wi-Fi bssid.

Parameters

in	*bssid*	the string of bssid

Returns

0 : success
other : failed

3.9.4.25 wifi_config_set_channel()

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

Set primary/secondary channel of OPL1000.

Attention

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

Parameters

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

Returns

0 : success
other : failed

3.9.4.26 wifi_config_set_dtim_interval()

```
int wifi_config_set_dtim_interval (
            uint8_t interval )
```

3.9.4.27 wifi_config_set_listen_interval()

```
int wifi_config_set_listen_interval (
            uint8_t interval )
```

3.9.4.28 wifi_config_set_mac_address()

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

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

Attention

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

Parameters

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

Returns

0 : success
other : failed

3.9.4.29 wifi_config_set_mac_tx_data_rate()

```
int wifi_config_set_mac_tx_data_rate (
            wifi_mac_data_rate_t data_rate )
```

Set the Mac tx data rate setting of OPL1000.

Parameters

in	*data_rate*	Set the Mac tx data rate • WIFI_MAC_DATA_RATE_ARA • WIFI_MAC_DATA_RATE_1M • WIFI_MAC_DATA_RATE_2M • WIFI_MAC_DATA_RATE_5_5M • WIFI_MAC_DATA_RATE_11M

Returns

0 : success
other : failed

3.9.4.30 wifi_config_set_opmode()

```
int wifi_config_set_opmode (
            uint8_t mode )
```

3.9.4.31 wifi_config_set_skip_dtim()

```
int wifi_config_set_skip_dtim (
            uint8_t value,
            bool save_flash )
```

Set the Skip DTIM value of OPL1000.

Parameters

in	*value*	Set the Skip DTIM value
in	*save_flash*	Enable/Disable to write in flash. • 0 : Not write in flash. (Only effect in runtime) • 1 : Write in flash and effect the value.

Attention

1. This API will set the skip DTIM value to share memory and stored in flash, please use wifi_config_get_skip_dtim() to check it.
2. The setting will be effect after next connect. We recommend re-connect AP after setting to make sure the value is correct with negotiate between AP.

Returns

0 : success
other : failed

3.9.4.32 wifi_config_set_ssid()

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

Set the ssid value of the current device.

Parameters

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

Returns

0 : success
other : failed

3.9.4.33 wifi_connection_connect()

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

Connect OPL1000 Wi-Fi station to certain AP.

Attention

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

Parameters

in	*config*	Establish connection parameters

Returns

0 : success
other : failed

3.9.4.34 wifi_connection_connect_from_ac_index()

```
int wifi_connection_connect_from_ac_index (
            uint8_t index )
```

Connect OPL1000 Wi-Fi station to certain AP by auto connect index.

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.
3. Then index should be 0 to begin.

Parameters

in	*index*	The index of AP in auto connect list

Returns

0 : success
1 : The index of AP is null
other : failed

3.9.4.35 wifi_connection_connect_from_ac_list()

```
int wifi_connection_connect_from_ac_list (
            wifi_config_t * config )
```

Connect OPL1000 Wi-Fi station to certain AP in auto connect list.

Attention

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

Parameters

in	*config*	Establish connection parameters

Returns

0 : success
1 : Not found in list
other : failed

3.9.4.36 wifi_connection_disconnect_ap()

```
int wifi_connection_disconnect_ap (
            void  )
```

Disconnect the link between OPL1000 and connected AP.

Returns

0 : success
other : failed

3.9.4.37 wifi_connection_disconnect_sta()

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

Disconnect the link between the current device and the station.

Parameters

in	*address*	station address

Returns

0 : success
other : failed

3.9.4.38 wifi_connection_get_rssi()

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

get signal strength of AP

Attention

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

Parameters

out	*rssi*	rssi value

Returns

0 : success
other : failed

3.9.4.39 wifi_connection_register_event_handler()

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

register wifi call back handler

Parameters

in	*event*	The type of the registered event. Options are • WIFI_EVENT_INIT_COMPLETE • 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
in	*handler*	registered event handler

Returns

0 : success
other : failed

3.9.4.40 wifi_connection_scan_start()

```
int wifi_connection_scan_start (
            uint8_t * ssid,
            uint8_t ssid_length,
            uint8_t * bssid,
            uint8_t scan_mode,
            uint8_t scan_option )
```

3.9.4.41 wifi_connection_unregister_event_handler()

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

unregister wifi call back handler

Parameters

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

Returns

0 : success
other : failed

3.9.4.42 wifi_convert_auth_mode()

```
int wifi_convert_auth_mode (
            scan_info_t * pinfo,
            int privacy )
```

3.9.4.43 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

3.9.4.44 wifi_fast_connect_get_mode()

```
int wifi_fast_connect_get_mode (
            uint8_t ap_index,
            uint8_t * mode )
```

Get the fast connect mode.

Attention

1. API returns false if try to get fast connect mode which something error

Parameters

in	*index*	The index of AP position • Range is 0 to 2
out	*mode*	Get the fast connect mode

Returns

0 : success
other : failed

3.9.4.45 wifi_fast_connect_set_mode()

```
int wifi_fast_connect_set_mode (
            uint8_t ap_index,
            uint8_t mode )
```

Set the fast connect mode.

Attention

1. API returns false if try to set fast connect mode which something error

Parameters

in	*index*	The index of AP position • Range is 0 to 2
in	*mode*	The fast connect mode

Returns

0 : success
other : failed

3.9.4.46 wifi_fast_connect_start()

```
int wifi_fast_connect_start (
            uint8_t ap_index )
```

Start fast connect mechanism.

Attention

1. API returns false if try to start fast connect function which something error

Parameters

in	*index*	The index of AP position • Range is 0 to 2

Returns

0 : success
other : failed

3.9.4.47 wifi_get_config()

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

Get configuration of specified interface.

Parameters

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

Returns

0 : success
other : failed

3.9.4.48 wifi_init()

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

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

Attention

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

Parameters

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

Returns

0 : success
other : failed

3.9.4.49 wifi_scan_get_ap_list()

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

Get list of APs that found in last scan operation.

Attention

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

Parameters

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

Returns

0 : success
other : failed

3.9.4.50 wifi_scan_get_ap_num()

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

Get the number of scanned APs.

Parameters

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

Attention

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

Returns

the scan result of AP number

3.9.4.51 wifi_scan_get_ap_records()

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

Get AP list found in last scan operation.

Parameters

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

Returns

0 : success
other : failed

3.9.4.52 wifi_scan_start()

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

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

Parameters

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

Returns

0 : success
other : failed

3.9.4.53 wifi_scan_stop()

```
int wifi_scan_stop (
            void  )
```

Stop scanning process.

Attention

This API shall be called after wifi_scan_start()

Returns

0 : success
other : failed

3.9.4.54 wifi_set_config()

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

Set configuration of OPL1000 STA.

Attention

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

Parameters

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

Returns

0 : success
other : failed

3.9.4.55 wifi_sta_get_ap_info()

```
int wifi_sta_get_ap_info (
            wifi_ap_record_t * ap_info )
```

Get information of AP which OPL1000 station is associated with.

Parameters

out	*ap_info*	get AP information from list

Returns

0 : success
other : failed

3.9.4.56 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

3.9.4.57 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

3.9.5 Variable Documentation

3.9.5.1 wifi_auto_connect_clear_ap_info_api

```
wifi_auto_connect_clear_ap_info_fp_t wifi_auto_connect_clear_ap_info_api
```

3.9.5.2 wifi_auto_connect_get_ap_info_api

```
wifi_auto_connect_get_ap_info_fp_t wifi_auto_connect_get_ap_info_api
```

3.9.5.3 wifi_auto_connect_get_ap_num_api

```
wifi_auto_connect_get_ap_num_fp_t wifi_auto_connect_get_ap_num_api
```

3.9.5.4 wifi_auto_connect_get_mode_api

```
wifi_auto_connect_get_mode_fp_t wifi_auto_connect_get_mode_api
```

3.9.5.5 wifi_auto_connect_get_saved_ap_num_api

```
wifi_auto_connect_get_saved_ap_num_fp_t wifi_auto_connect_get_saved_ap_num_api
```

3.9.5.6 wifi_auto_connect_init_api

```
wifi_auto_connect_init_fp_t wifi_auto_connect_init_api
```

3.9.5.7 wifi_auto_connect_reset_api

```
wifi_auto_connect_reset_fp_t wifi_auto_connect_reset_api
```

3.9.5.8 wifi_auto_connect_set_ap_num_api

wifi_auto_connect_set_ap_num_fp_t wifi_auto_connect_set_ap_num_api

3.9.5.9 wifi_auto_connect_set_mode_api

wifi_auto_connect_set_mode_fp_t wifi_auto_connect_set_mode_api

3.9.5.10 wifi_auto_connect_start_api

wifi_auto_connect_start_fp_t wifi_auto_connect_start_api

3.9.5.11 wifi_auto_connect_update_ch_api

wifi_auto_connect_update_ch_fp_t wifi_auto_connect_update_ch_api

3.9.5.12 wifi_config_get_bandwidth_api

wifi_config_get_bandwidth_fp_t wifi_config_get_bandwidth_api

3.9.5.13 wifi_config_get_bssid_api

wifi_config_get_bssid_fp_t wifi_config_get_bssid_api

3.9.5.14 wifi_config_get_channel_api

wifi_config_get_channel_fp_t wifi_config_get_channel_api

3.9.5.15 wifi_config_get_dtim_interval_api

wifi_config_get_dtim_interval_fp_t wifi_config_get_dtim_interval_api

3.9.5.16 wifi_config_get_listen_interval_api

wifi_config_get_listen_interval_fp_t wifi_config_get_listen_interval_api

3.9.5.17 wifi_config_get_mac_address_api

wifi_config_get_mac_address_fp_t wifi_config_get_mac_address_api

3.9.5.18 wifi_config_get_mac_tx_data_rate_api

wifi_config_get_mac_tx_data_rate_fp_t wifi_config_get_mac_tx_data_rate_api

3.9.5.19 wifi_config_get_opmode_api

wifi_config_get_opmode_fp_t wifi_config_get_opmode_api

3.9.5.20 wifi_config_get_skip_dtim_api

wifi_config_get_skip_dtim_fp_t wifi_config_get_skip_dtim_api

3.9.5.21 wifi_config_get_ssid_api

wifi_config_get_ssid_fp_t wifi_config_get_ssid_api

3.9.5.22 wifi_config_get_sta_mac_address_from_flash_api

wifi_config_get_sta_mac_address_from_flash_fp_t wifi_config_get_sta_mac_address_from_flash_api

3.9.5.23 wifi_config_set_bandwidth_api

wifi_config_set_bandwidth_fp_t wifi_config_set_bandwidth_api

3.9.5.24 wifi_config_set_bssid_api

wifi_config_set_bssid_fp_t wifi_config_set_bssid_api

3.9.5.25 wifi_config_set_channel_api

wifi_config_set_channel_fp_t wifi_config_set_channel_api

3.9.5.26 wifi_config_set_dtim_interval_api

wifi_config_set_dtim_interval_fp_t wifi_config_set_dtim_interval_api

3.9.5.27 wifi_config_set_listen_interval_api

wifi_config_set_listen_interval_fp_t wifi_config_set_listen_interval_api

3.9.5.28 wifi_config_set_mac_address_api

wifi_config_set_mac_address_fp_t wifi_config_set_mac_address_api

3.9.5.29 wifi_config_set_mac_tx_data_rate_api

wifi_config_set_mac_tx_data_rate_fp_t wifi_config_set_mac_tx_data_rate_api

3.9.5.30 wifi_config_set_opmode_api

wifi_config_set_opmode_fp_t wifi_config_set_opmode_api

3.9.5.31 wifi_config_set_skip_dtim_api

wifi_config_set_skip_dtim_fp_t wifi_config_set_skip_dtim_api

3.9.5.32 wifi_config_set_ssid_api

wifi_config_set_ssid_fp_t wifi_config_set_ssid_api

3.9.5.33 wifi_connection_connect_api

wifi_connection_connect_fp_t wifi_connection_connect_api

3.9.5.34 wifi_connection_connect_from_ac_index_api

wifi_connection_connect_from_ac_index_fp_t wifi_connection_connect_from_ac_index_api

3.9.5.35 wifi_connection_connect_from_ac_list_api

wifi_connection_connect_from_ac_list_fp_t wifi_connection_connect_from_ac_list_api

3.9.5.36 wifi_connection_disconnect_ap_api

wifi_connection_disconnect_ap_fp_t wifi_connection_disconnect_ap_api

3.9.5.37 wifi_connection_disconnect_sta_api

wifi_connection_disconnect_sta_fp_t wifi_connection_disconnect_sta_api

3.9.5.38 wifi_connection_get_rssi_api

wifi_connection_get_rssi_fp_t wifi_connection_get_rssi_api

3.9.5.39 wifi_connection_register_event_handler_api

wifi_connection_register_event_handler_fp_t wifi_connection_register_event_handler_api

3.9.5.40 wifi_connection_scan_start_api

wifi_connection_scan_start_fp_t wifi_connection_scan_start_api

3.9.5.41 wifi_connection_unregister_event_handler_api

wifi_connection_unregister_event_handler_fp_t wifi_connection_unregister_event_handler_api

3.9.5.42 wifi_convert_auth_mode_api

wifi_convert_auth_mode_fp_t wifi_convert_auth_mode_api

3.9.5.43 wifi_deinit_api

wifi_deinit_fp_t wifi_deinit_api

3.9.5.44 wifi_fast_connect_get_mode_api

wifi_fast_connect_get_mode_fp_t wifi_fast_connect_get_mode_api

3.9.5.45 wifi_fast_connect_set_mode_api

wifi_fast_connect_set_mode_fp_t wifi_fast_connect_set_mode_api

3.9.5.46 wifi_fast_connect_start_api

wifi_fast_connect_start_fp_t wifi_fast_connect_start_api

3.9.5.47 wifi_get_config_api

wifi_get_config_fp_t wifi_get_config_api

3.9.5.48 wifi_init_api

wifi_init_fp_t wifi_init_api

3.9.5.49 wifi_scan_get_ap_list_api

wifi_scan_get_ap_list_fp_t wifi_scan_get_ap_list_api

3.9.5.50 wifi_scan_get_ap_num_api

wifi_scan_get_ap_num_fp_t wifi_scan_get_ap_num_api

3.9.5.51 wifi_scan_get_ap_records_api

wifi_scan_get_ap_records_fp_t wifi_scan_get_ap_records_api

3.9.5.52 wifi_scan_start_api

wifi_scan_start_fp_t wifi_scan_start_api

3.9.5.53 wifi_scan_stop_api

wifi_scan_stop_fp_t wifi_scan_stop_api

3.9.5.54 wifi_set_config_api

wifi_set_config_fp_t wifi_set_config_api

3.9.5.55 wifi_sta_get_ap_info_api

wifi_sta_get_ap_info_fp_t wifi_sta_get_ap_info_api

3.9.5.56 wifi_start_api

wifi_start_fp_t wifi_start_api

3.9.5.57 wifi_stop_api

wifi_stop_fp_t wifi_stop_api

3.10 Enumeration

Enumerations

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

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

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

  This enumeration defines the supported events generated by the Wi-Fi driver. The event will be sent to the upper layer handler registered in wifi_register_event_handler().
- enum wifi_mac_data_rate_t {
  WIFI_MAC_DATA_RATE_ARA = 0, WIFI_MAC_DATA_RATE_1M, WIFI_MAC_DATA_RATE_2M,
  WIFI_MAC_DATA_RATE_5_5M,
  WIFI_MAC_DATA_RATE_11M }

  This enumeration defines wifi mac tx data rates..
- 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_UNACC
  WIFI_REASON_CODE_INVALID_INFO_ELEM = 13, WIFI_REASON_CODE_MIC_FAILURE, WIFI_REASON_CODE_4_WAY
  WIFI_REASON_CODE_GROUP_KEY_UPDATE_TIMEOUT,
  WIFI_REASON_CODE_DIFFERENT_INFO_ELEM, WIFI_REASON_CODE_GROUP_CIPHER_INVALID_VALID,
  WIFI_REASON_CODE_PAIRWISE_CIPHER_INVALID, WIFI_REASON_CODE_AKMP_INVALID,
  WIFI_REASON_CODE_UNSUPPORTED_RSNE_VERSION, WIFI_REASON_CODE_INVALID_RSNE_CAPABILITIES,
  WIFI_REASON_CODE_IEEE_802_1X_AUTH_FAILED, WIFI_REASON_CODE_CIPHER_REJECTED,
  WIFI_REASON_CODE_AUTO_CONNECT_FAILED = 200, WIFI_REASON_CODE_CONNECT_NOT_FOUND,
  WIFI_REASON_CODE_CONNECT_TIMEOUT }

  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, WIFI_SCAN_TYPE_MIX
  }

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

3.10.1 Detailed Description

3.10.2 Enumeration Type Documentation

3.10.2.1 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

3.10.2.2 wifi_bandwidth_t

```
enum wifi_bandwidth_t
```

Enumerator

WIFI_BW_HT20	Bandwidth is HT20
WIFI_BW_HT40	Bandwidth is HT40

3.10.2.3 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

3.10.2.4 wifi_event_t

`enum wifi_event_t`

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

Enumerator

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

3.10.2.5 wifi_mac_data_rate_t

`enum wifi_mac_data_rate_t`

This enumeration defines wifi mac tx data rates..

Enumerator

WIFI_MAC_DATA_RATE_ARA	Auto Rate Adaptation
WIFI_MAC_DATA_RATE_1M	Fix Mac Tx data rate in 1 Mbps
WIFI_MAC_DATA_RATE_2M	Fix Mac Tx data rate in 2 Mbps
WIFI_MAC_DATA_RATE_5_5M	Fix Mac Tx data rate in 5.5 Mbps
WIFI_MAC_DATA_RATE_11M	Fix Mac Tx data rate in 11 Mbps

3.10.2.6 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	

3.10.2.7 wifi_reason_code_t

```
enum wifi_reason_code_t
```

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

Enumerator

WIFI_REASON_CODE_SUCCESS	0 Reserved.
WIFI_REASON_CODE_FIND_AP_FAIL	1 (Internal) No AP found.
WIFI_REASON_CODE_PREV_AUTH_INVALID	2 Previous authentication is no longer valid.
WIFI_REASON_CODE_DEAUTH_LEAVING_BSS	3 Deauthenticated because sending STA is leaving (or has left) IBSS or ES.
WIFI_REASON_CODE_DISASSOC_INACTIVITY	4 Disassociated due to inactivity.
WIFI_REASON_CODE_DISASSOC_AP_OVERL↩ OAD	5 Disassociated because AP is unable to handle all currently associated STAs.
WIFI_REASON_CODE_CLASS_2_ERR	6 Class 2 frame received from nonauthenticated STA.
WIFI_REASON_CODE_CLASS_3_ERR	7 Class 3 frame received from nonauthenticated STA.
WIFI_REASON_CODE_DISASSOC_LEAVING_BSS	8 Disassociated because sending STA is leaving (or has left) BSS.
WIFI_REASON_CODE_ASSOC_BEFORE_AUTH	9 STA requesting (re)association is not authenticated with responding STA.
WIFI_REASON_CODE_DISASSOC_PWR_CAP_↩ UNACCEPTABLE	10 Disassociated because the information in the Power Capability element is unacceptable.
WIFI_REASON_CODE_DISASSOC_SUP_CHS_U↩ NACCEPTABLE	11 Disassociated because the information in the Supported Channels element is unacceptable.
WIFI_REASON_CODE_INVALID_INFO_ELEM	13 Invalid information element.
WIFI_REASON_CODE_MIC_FAILURE	14 Message integrity code (MIC) failure.
WIFI_REASON_CODE_4_WAY_HANDSHAKE_TI↩ MEOUT	15 4-Way Handshake time out.
WIFI_REASON_CODE_GROUP_KEY_UPDATE_↩ TIMEOUT	16 Group Key Handshake time out.
WIFI_REASON_CODE_DIFFERENT_INFO_ELEM	17 Information element in 4-Way Handshake different from (Re)Association Request/Probe Response/Beacon frame.
WIFI_REASON_CODE_GROUP_CIPHER_INVALI↩ D_VALID	18 Invalid group cipher.
WIFI_REASON_CODE_PAIRWISE_CIPHER_INV↩ ALID	19 Invalid pairwise cipher.
WIFI_REASON_CODE_AKMP_INVALID	20 Invalid AKMP.
WIFI_REASON_CODE_UNSUPPORTED_RSNE_↩ VERSION	21 Unsupported RSN information element version.
WIFI_REASON_CODE_INVALID_RSNE_CAPABI↩ LITIES	22 Invalid RSN information element capabilities.
WIFI_REASON_CODE_IEEE_802_1X_AUTH_FAI↩ LED	23 IEEE 802.1X authentication failed.
WIFI_REASON_CODE_CIPHER_REJECTED	24 Cipher suite rejected because of the security policy.
WIFI_REASON_CODE_AUTO_CONNECT_FAILED	200 Auto connect failed.
WIFI_REASON_CODE_CONNECT_NOT_FOUND	201 The target AP is not found.
WIFI_REASON_CODE_CONNECT_TIMEOUT	202 Connect to AP timeout.

3.10.2.8 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

3.10.2.9 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
WIFI_SCAN_TYPE_MIX	Active + Passive scan

3.10.2.10 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 4

Data Structure Documentation

4.1 _wpa_ie_data Struct Reference

`#include <controller_wifi_com.h>`

Data Fields

- int capabilities
- int group_cipher
- int key_mgmt
- int mgmt_group_cipher
- size_t num_pmkid
- int pairwise_cipher
- const u8 * pmkid
- int proto

4.1.1 Field Documentation

4.1.1.1 capabilities

`int capabilities`

4.1.1.2 group_cipher

`int group_cipher`

4.1.1.3 key_mgmt

```
int key_mgmt
```

4.1.1.4 mgmt_group_cipher

```
int mgmt_group_cipher
```

4.1.1.5 num_pmkid

```
size_t num_pmkid
```

4.1.1.6 pairwise_cipher

```
int pairwise_cipher
```

4.1.1.7 pmkid

```
const u8* pmkid
```

4.1.1.8 proto

```
int proto
```

4.2 asso_data Struct Reference

```
#include <controller_wifi_com.h>
```

Data Fields

- unsigned int eap_workaround
- int eapol_flags
- int group_cipher
- int key_mgmt
- int leap
- int mgmt_group_cipher
- int non_leap
- int pairwise_cipher
- char * passphrase
- int proto
- u8 psk [32]
- int psk_set

4.2.1 Field Documentation

4.2.1.1 eap_workaround

```
unsigned int eap_workaround
```

4.2.1.2 eapol_flags

```
int eapol_flags
```

4.2.1.3 group_cipher

```
int group_cipher
```

4.2.1.4 key_mgmt

```
int key_mgmt
```

4.2.1.5 leap

```
int leap
```

4.2.1.6 mgmt_group_cipher

```
int mgmt_group_cipher
```

4.2.1.7 non_leap

```
int non_leap
```

4.2.1.8 pairwise_cipher

```
int pairwise_cipher
```

4.2.1.9 passphrase

```
char* passphrase
```

4.2.1.10 proto

```
int proto
```

4.2.1.11 psk

```
u8 psk[32]
```

4.2.1.12 psk_set

```
int psk_set
```

4.3 auto_conn_info_t Struct Reference

```
#include <controller_wifi_com.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]
- u8 hid_ssid_len
- u64 latest_beacon_rx_time
- s8 passphrase [MAX_LEN_OF_PASSPHRASE]
- u8 psk [32]
- u8 rsn_ie [256]
- s8 rssi
- s8 ssid [IEEE80211_MAX_SSID_LEN+1]
- u8 ssid_len
- u8 supported_rates [IEEE80211_MAX_SUPPORTED_RATES]
- wpa_ie_data_t wpa_data
- u8 wpa_ie [257]

4.3.1 Field Documentation

4.3.1.1 ap_channel

```
u8 ap_channel
```

4.3.1.2 beacon_interval

```
u16 beacon_interval
```

4.3.1.3 bssid

```
u8 bssid[MAC_ADDR_LEN]
```

4.3.1.4 capabilities

```
u16 capabilities
```

4.3.1.5 dtim_prod

```
u8 dtim_prod
```

4.3.1.6 fast_connect

```
u8 fast_connect
```

4.3.1.7 free_ocpy

```
bool free_ocpy
```

4.3.1.8 hid_ssid

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

4.3.1.9 hid_ssid_len

```
u8 hid_ssid_len
```

4.3.1.10 latest_beacon_rx_time

```
u64 latest_beacon_rx_time
```

4.3.1.11 passphrase

```
s8 passphrase[MAX_LEN_OF_PASSPHRASE]
```

4.3.1.12 psk

```
u8 psk[32]
```

4.3.1.13 rsn_ie

```
u8 rsn_ie[256]
```

4.3.1.14 rssi

```
s8 rssi
```

4.3.1.15 ssid

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

4.3.1.16 ssid_len

```
u8 ssid_len
```

4.3.1.17 supported_rates

```
u8 supported_rates[IEEE80211_MAX_SUPPORTED_RATES]
```

4.3.1.18 wpa_data

```
wpa_ie_data_t wpa_data
```

4.3.1.19 wpa_ie

```
u8 wpa_ie[257]
```

4.4 auto_connect_cfg_t Struct Reference

```
#include <controller_wifi_com.h>
```

Data Fields

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

4.4.1 Field Documentation

4.4.1.1 flag

```
bool flag
```

4.4.1.2 front

```
s8 front
```

4.4.1.3 max_save_num

```
u8 max_save_num
```

4.4.1.4 pFCInfo

```
auto_conn_info_t* pFCInfo
```

4.4.1.5 rear

```
s8 rear
```

4.4.1.6 retryCount

```
u8 retryCount
```

4.4.1.7 targetIdx

```
u8 targetIdx
```

4.4.1.8 uFCApNum

```
u32 uFCApNum
```

4.5 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

4.5.1 Detailed Description

Send information to event by event_msg_t.

4.5.2 Field Documentation

4.5.2.1 event

```
uint32_t event
```

event type

4.5.2.2 length

```
uint32_t length
```

Packet length

4.5.2.3 param

```
uint8_t* param
```

event parament

4.6 hap_control_t Struct Reference

```
#include <controller_wifi_com.h>
```

Data Fields

- auto_conn_info_t * hap_ap_info
- u16 hap_bitvector
- u8 hap_en
- u8 hap_final_index
- u8 hap_index
- char hap_ssid [IEEE80211_MAX_SSID_LEN+1]

4.6.1 Field Documentation

4.6.1.1 hap_ap_info

```
auto_conn_info_t* hap_ap_info
```

4.6.1.2 hap_bitvector

```
u16 hap_bitvector
```

4.6.1.3 hap_en

```
u8 hap_en
```

4.6.1.4 hap_final_index

```
u8 hap_final_index
```

4.6.1.5 hap_index

```
u8 hap_index
```

4.6.1.6 hap_ssid

```
char hap_ssid[IEEE80211_MAX_SSID_LEN+1]
```

4.7 LE_BT_ADDR_T Struct Reference

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

Data Fields

- BD_ADDR addr
- UINT8 type

4.7.1 Field Documentation

4.7.1.1 addr

```
BD_ADDR addr
```

address

4.7.1.2 type

```
UINT8 type
```

address type

4.8 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

4.8.1 Field Documentation

4.8.1.1 conn_hdl

`UINT16 conn_hdl`

connection handle

4.8.1.2 conn_interval

`UINT16 conn_interval`

connection interval

4.8.1.3 conn_latency

`UINT16 conn_latency`

connection latency

4.8.1.4 dev_id

`UINT16 dev_id`

device ID

4.8.1.5 peer_addr

```
BD_ADDR peer_addr
```

perr address

4.8.1.6 peer_addr_type

```
UINT8 peer_addr_type
```

peer address type

4.8.1.7 role

```
UINT8 role
```

master or slave

4.8.1.8 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.8.1.9 supervison_timeout

```
UINT16 supervison_timeout
```

supervision timeout

4.9 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

4.9.1 Field Documentation

4.9.1.1 addr

```
BD_ADDR addr
```

address

4.9.1.2 addr_type

```
UINT8 addr_type
```

address type

4.9.1.3 data

```
UINT8 data[1]
```

4.9.1.4 event_type

```
UINT8 event_type
```

4.9.1.5 len

```
UINT8 len
```

4.9.1.6 rssi

```
INT8 rssi
```

RSSI

4.10 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

4.10.1 Field Documentation

4.10.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.10.1.2 itv_max

```
UINT16 itv_max
```

maxinum connection interval

4.10.1.3 itv_min

```
UINT16 itv_min
```

mininum connection interval

4.10.1.4 latency

```
UINT16 latency
```

slave latency

4.10.1.5 sv_tmo

```
UINT32 sv_tmo
```

supervision timeout

4.11 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

4.11.1 Field Documentation

4.11.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.11.1.2 interval

```
UINT16 interval
```

connection interval

4.11.1.3 latency

```
UINT16 latency
```

slave letency

4.11.1.4 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.11.1.5 supervision_timeout

```
UINT32 supervision_timeout
```

supervision timeout

4.12 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

4.12.1 Field Documentation

4.12.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.12.1.2 max_rx_octets

```
UINT16 max_rx_octets
```

connMaxRxOctets

4.12.1.3 max_rx_time

```
UINT16 max_rx_time
```

connMaxRxTime

4.12.1.4 max_tx_octets

```
UINT16 max_tx_octets
```

connMaxTxOctets

4.12.1.5 max_tx_time

```
UINT16 max_tx_time
```

connMaxTxTime

4.13 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

4.13.1 Field Documentation

4.13.1.1 direct_addr

```
BD_ADDR direct_addr
```

direct address

4.13.1.2 direct_addr_type

```
UINT8 direct_addr_type
```

direct address type

4.13.1.3 peer_addr

```
BD_ADDR peer_addr
```

peer address

4.13.1.4 peer_addr_type

```
UINT8 peer_addr_type
```

peer address type

4.13.1.5 rssi

```
INT8 rssi
```

RSSI

4.14 LE_CM_MSG_DISCONNECT_COMPLETE_IND_T Struct Reference

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

Data Fields

- UINT16 conn_hdl
- UINT8 reason
- UINT16 status

4.14.1 Field Documentation

4.14.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.14.1.2 reason

```
UINT8 reason
```

disconnect reason

4.14.1.3 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.15 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

4.15.1 Field Documentation

4.15.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.15.1.2 devid

```
UINT16 devid
```

device ID

4.15.1.3 enabled

```
UINT8 enabled
```

4.15.1.4 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.16 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

4.16.1 Field Documentation

4.16.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.16.1.2 devid

```
UINT16 devid
```

device ID

4.16.1.3 enabled

```
BOOL enabled
```

enable or disable

4.16.1.4 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.17 LE_CM_MSG_INIT_COMPLETE_CFM_T Struct Reference

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

Data Fields

- UINT16 status

4.17.1 Field Documentation

4.17.1.1 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.18 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]

4.18.1 Field Documentation

4.18.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.18.1.2 devid

```
UINT16 devid
```

device ID

4.18.1.3 ediv

```
UINT16 ediv
```

4.18.1.4 rand

```
UINT8 rand[8]
```

4.19 LE_CM_MSG_READ_ADV_TX_POWER_CFM_T Struct Reference

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

Data Fields

- INT8 pwr_level
- UINT16 status

4.19.1 Field Documentation

4.19.1.1 pwr_level

```
INT8 pwr_level
```

power level

4.19.1.2 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.20 LE_CM_MSG_READ_BD_ADDR_CFM_T Struct Reference

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

Data Fields

- BD_ADDR bd_addr
- UINT16 status

4.20.1 Field Documentation

4.20.1.1 bd_addr

```
BD_ADDR bd_addr
```

4.20.1.2 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.21 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

4.21.1 Field Documentation

4.21.1.1 ch_map

`UINT8 ch_map[5]`

channel map

4.21.1.2 conn_hdl

`UINT16 conn_hdl`

connection handle

4.21.1.3 status

`UINT16 status`

refer to LE_ERR_STATE in ble_err.h

4.22 LE_CM_MSG_READ_PHY_CFM_T Struct Reference

`#include <ble_cm_if.h>`

Data Fields

- UINT16 conn_hdl
- UINT8 rx_phy
- UINT16 status
- UINT8 tx_phy

4.22.1 Field Documentation

4.22.1.1 conn_hdl

```
UINT16 conn_hdl
```

4.22.1.2 rx_phy

```
UINT8 rx_phy
```

4.22.1.3 status

```
UINT16 status
```

4.22.1.4 tx_phy

```
UINT8 tx_phy
```

4.23 LE_CM_MSG_READ_RESOLVING_LIST_SIZE_CFM_T Struct Reference

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

Data Fields

- UINT8 size
- UINT16 status

4.23.1 Field Documentation

4.23.1.1 size

```
UINT8 size
```

resolving list size

4.23.1.2 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.24 LE_CM_MSG_READ_RSSI_CFM_T Struct Reference

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

Data Fields

- UINT16 conn_hdl
- INT8 rssi
- UINT16 status

4.24.1 Field Documentation

4.24.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.24.1.2 rssi

```
INT8 rssi
```

RSSI

4.24.1.3 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.25 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

4.25.1 Field Documentation

4.25.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.25.1.2 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.25.1.3 tx_power

```
INT8 tx_power
```

tx power

4.26 LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM_T Struct Reference

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

Data Fields

- UINT8 size
- UINT16 status

4.26.1 Field Documentation

4.26.1.1 size

```
UINT8 size
```

white list size

4.26.1.2 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.27 LE_CM_MSG_SET_DATA_LENGTH_CFM_T Struct Reference

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

Data Fields

- UINT16 conn_hdl
- UINT16 status

4.27.1 Field Documentation

4.27.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.27.1.2 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.28 LE_CM_MSG_SET_DISCONNECT_CFM_T Struct Reference

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

Data Fields

- UINT16 handle
- UINT16 status

4.28.1 Field Documentation

4.28.1.1 handle

```
UINT16 handle
```

connection handle

4.28.1.2 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.29 LE_CM_MSG_SET_PHY_CFM_T Struct Reference

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

Data Fields

- UINT16 conn_hdl
- UINT16 status

4.29.1 Field Documentation

4.29.1.1 conn_hdl

```
UINT16 conn_hdl
```

4.29.1.2 status

```
UINT16 status
```

4.30 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

4.30.1 Field Documentation

4.30.1.1 conn_hdl

`UINT16 conn_hdl`

connection handle

4.30.1.2 identifier

`UINT16 identifier`

4.30.1.3 interval_max

`UINT16 interval_max`

maxinum connection interval

4.30.1.4 interval_min

`UINT16 interval_min`

mininum connection interval

4.30.1.5 slave_latency

`UINT16 slave_latency`

slave latency

4.30.1.6 timeout_multiplier

```
UINT32 timeout_multiplier
```

4.31 LE_CM_REQ_STATUS_T Struct Reference

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

Data Fields

- UINT16 status

4.31.1 Field Documentation

4.31.1.1 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.32 LE_CONN_PARA_T Struct Reference

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

Data Fields

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

4.32.1 Field Documentation

4.32.1.1 itv_max

```
UINT16 itv_max
```

maxinum connection interval

4.32.1.2 itv_min

```
UINT16 itv_min
```

mininum connection interval

4.32.1.3 latency

```
UINT16 latency
```

slave latency

4.32.1.4 sv_timeout

```
UINT16 sv_timeout
```

supervision timeout

4.33 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

4.33.1 Field Documentation

4.33.1.1 channel_map

```
UINT8 channel_map
```

advertising channel map

4.33.1.2 filter_policy

```
UINT8 filter_policy
```

advertising filter policy

4.33.1.3 interval_max

```
UINT16 interval_max
```

maxinum advertising interval

4.33.1.4 interval_min

```
UINT16 interval_min
```

mininum advertising interval

4.33.1.5 own_addr_type

```
UINT8 own_addr_type
```

owner address type

4.33.1.6 peer_addr

```
BD_ADDR peer_addr
```

peer address

4.33.1.7 peer_addr_type

```
UINT8 peer_addr_type
```

peer address type

4.33.1.8 type

```
UINT8 type
```

advertising type

4.34 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

4.34.1 Field Documentation

4.34.1.1 interval_max

```
UINT16 interval_max
```

maxinum connection interval

4.34.1.2 interval_min

```
UINT16 interval_min
```

mininum connection interval

4.34.1.3 latency

```
UINT16 latency
```

slave latency

4.34.1.4 supervision_timeout

```
UINT16 supervision_timeout
```

supervision timeout for the LE Link

4.35 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

4.35.1 Field Documentation

4.35.1.1 filter_policy

```
UINT8 filter_policy
```

scan filter policy

4.35.1.2 interval

```
UINT16 interval
```

scan interval

4.35.1.3 own_addr_type

```
UINT8 own_addr_type
```

owner address type

4.35.1.4 type

```
UINT8 type
```

scan type

4.35.1.5 window

```
UINT16 window
```

scan window

4.36 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

4.36.1 Field Documentation

4.36.1.1 format

```
UINT8 format
```

UUID type

4.36.1.2 handle

```
UINT16 handle
```

handle

4.36.1.3 len

```
UINT16 len
```

value length

4.36.1.4 maxLen

```
UINT16 maxLen
```

maxinum value length

4.36.1.5 permit

```
UINT16 permit
```

permit

4.36.1.6 pUuid

```
UINT16* const pUuid
```

UUID

4.36.1.7 pVal

```
UINT8* const pVal
```

value

4.37 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

4.37.1 Field Documentation

4.37.1.1 conn_hdl

`UINT16 conn_hdl`

connection handle

4.37.1.2 devid

`UINT16 devid`

device index

4.37.1.3 handle

`UINT16 handle`

attribute handle

4.37.1.4 offset

`UINT16 offset`

attribute handle value

4.38 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

4.38.1 Field Documentation

4.38.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.38.1.2 devid

```
UINT16 devid
```

device ID

4.38.1.3 flag

```
UINT8 flag
```

refer to LE_GATT_FLAG_* in ble_gatt_if.h

4.38.1.4 handle

```
UINT16 handle
```

attribute handle

4.38.1.5 len

```
UINT16 len
```

length written

4.38.1.6 offset

```
UINT16 offset
```

attribute handle value

4.38.1.7 pVal

```
UINT8* pVal
```

value written

4.39 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]

4.39.1 Field Documentation

4.39.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.39.1.2 devid

```
UINT16 devid
```

device ID

4.39.1.3 format

```
UINT8 format
```

UUID type

4.39.1.4 handle

```
UINT16 handle
```

characteristic descriptor handle

4.39.1.5 uuid

```
UINT16 uuid[8]
```

UUID

4.40 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

4.40.1 Field Documentation

4.40.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.40.1.2 devid

```
UINT16 devid
```

device ID

4.40.1.3 format

```
UINT8 format
```

UUID type

4.40.1.4 handle

```
UINT16 handle
```

characteristic declaration handle

4.40.1.5 property

```
UINT8 property
```

property

4.40.1.6 uuid

```
UINT16 uuid[8]
```

UUID

4.40.1.7 val_hdl

```
UINT16 val_hdl
```

characteristic value handle

4.41 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

4.41.1 Field Documentation

4.41.1.1 att_err

`UINT8 att_err`

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

4.41.1.2 conn_hdl

`UINT16 conn_hdl`

connection handle

4.41.1.3 devid

`UINT16 devid`

device ID

4.41.1.4 handle

`UINT16 handle`

characteristic value handle

4.41.1.5 len

`UINT16 len`

value length

4.41.1.6 offset

`UINT16 offset`

value position offset

4.41.1.7 val

`UINT8* val`

value

4.42 LE_GATT_MSG_CONFIRMATION_CFM_T Struct Reference

`#include <ble_gatt_if.h>`

Data Fields

- UINT16 conn_hdl
- UINT16 devid
- UINT16 handle

4.42.1 Field Documentation

4.42.1.1 conn_hdl

`UINT16 conn_hdl`

connection handle

4.42.1.2 devid

`UINT16 devid`

device ID

4.42.1.3 handle

`UINT16 handle`

attribute handle

4.43 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

4.43.1 Field Documentation

4.43.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.43.1.2 current_rx_mtu

```
UINT16 current_rx_mtu
```

current receive MTU

4.43.1.3 devid

```
UINT16 devid
```

device ID

4.44 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

4.44.1 Field Documentation

4.44.1.1 client_rx_mtu

```
UINT16 client_rx_mtu
```

client receive MTU

4.44.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.44.1.3 devid

```
UINT16 devid
```

device ID

4.45 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

4.45.1 Field Documentation

4.45.1.1 att_err

```
UINT8 att_err
```

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

4.45.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.45.1.3 devid

```
UINT16 devid
```

device ID

4.45.1.4 err_hdl

```
UINT16 err_hdl
```

TBD

4.45.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.46 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

4.46.1 Field Documentation

4.46.1.1 att_err

```
UINT8 att_err
```

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

4.46.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.46.1.3 devid

```
UINT16 devid
```

device ID

4.46.1.4 handle

```
UINT16 handle
```

characteristic descriptor handle

4.46.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.47 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

4.47.1 Field Documentation

4.47.1.1 att_err

```
UINT8 att_err
```

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

4.47.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.47.1.3 devid

```
UINT16 devid
```

device ID

4.47.1.4 handle

```
UINT16 handle
```

4.47.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.48 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

4.48.1 Field Documentation

4.48.1.1 att_err

```
UINT8 att_err
```

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

4.48.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.48.1.3 devid

```
UINT16 devid
```

device ID

4.48.1.4 handle

```
UINT16 handle
```

characteristic descriptor handle

4.48.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.49 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

4.49.1 Field Documentation

4.49.1.1 att_err

```
UINT8 att_err
```

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

4.49.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.49.1.3 devid

```
UINT16 devid
```

device ID

4.49.1.4 handle

```
UINT16 handle
```

include service start handle

4.49.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.50 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

4.50.1 Field Documentation

4.50.1.1 att_err

```
UINT8 att_err
```

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

4.50.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.50.1.3 devid

```
UINT16 devid
```

device ID

4.50.1.4 handle

```
UINT16 handle
```

service start handle

4.50.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.51 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]

4.51.1 Field Documentation

4.51.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.51.1.2 devid

```
UINT16 devid
```

device ID

4.51.1.3 end_hdl

```
UINT16 end_hdl
```

end handle

4.51.1.4 format

```
UINT8 format
```

UUID type

4.51.1.5 handle

```
UINT16 handle
```

include servie handle

4.51.1.6 start_hdl

```
UINT16 start_hdl
```

start handle

4.51.1.7 uuid

```
UINT16 uuid[8]
```

UUID

4.52 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

4.52.1 Field Documentation

4.52.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.52.1.2 devid

```
UINT16 devid
```

device ID

4.52.1.3 handle

```
UINT16 handle
```

attribute handle

4.52.1.4 len

```
UINT16 len
```

value length

4.52.1.5 val

```
UINT8* val
```

value

4.53 LE_GATT_MSG_NOTIFY_CFM_T Struct Reference

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

Data Fields

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

4.53.1 Field Documentation

4.53.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.53.1.2 devid

```
UINT16 devid
```

device ID

4.53.1.3 handle

```
UINT16 handle
```

attribute handle

4.53.1.4 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.54 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

4.54.1 Field Documentation

4.54.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.54.1.2 devid

```
UINT16 devid
```

device ID

4.54.1.3 handle

```
UINT16 handle
```

attribute handle

4.54.1.4 len

```
UINT16 len
```

value length

4.54.1.5 val

```
UINT8* val
```

value

4.55 LE_GATT_MSG_OPERATION_TIMEOUT_T Struct Reference

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

Data Fields

- UINT8 att_op
- UINT16 conn_hdl
- UINT16 devid

4.55.1 Field Documentation

4.55.1.1 att_op

```
UINT8 att_op
```

refer to LE_ATT_OP_* in ble_att_if.h

4.55.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.55.1.3 devid

```
UINT16 devid
```

device ID

4.56 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

4.56.1 Field Documentation

4.56.1.1 att_err

```
UINT8 att_err
```

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

4.56.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.56.1.3 devid

```
UINT16 devid
```

device ID

4.56.1.4 handle

```
UINT16 handle
```

attribute handle

4.56.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.57 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

4.57.1 Field Documentation

4.57.1.1 att_err

```
UINT8 att_err
```

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

4.57.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.57.1.3 devid

```
UINT16 devid
```

device ID

4.57.1.4 handle

```
UINT16 handle
```

characteristic value handle

4.57.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.58 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

4.58.1 Field Documentation

4.58.1.1 att_err

```
UINT8 att_err
```

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

4.58.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.58.1.3 devid

```
UINT16 devid
```

device ID

4.58.1.4 handle

```
UINT16 handle
```

characteristic value handle

4.58.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.59 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

4.59.1 Field Documentation

4.59.1.1 att_err

```
UINT8 att_err
```

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

4.59.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.59.1.3 devid

```
UINT16 devid
```

device ID

4.59.1.4 handle

```
UINT16 handle
```

characteristic value handle

4.59.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.60 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

4.60.1 Field Documentation

4.60.1.1 att_err

```
UINT8 att_err
```

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

4.60.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.60.1.3 devid

```
UINT16 devid
```

device ID

4.60.1.4 err_hdl

```
UINT16 err_hdl
```

TBD

4.60.1.5 len

```
UINT16 len
```

value length

4.60.1.6 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.60.1.7 val

```
UINT8* val
```

value

4.61 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]

4.61.1 Field Documentation

4.61.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.61.1.2 devid

```
UINT16 devid
```

device ID

4.61.1.3 end_hdl

```
UINT16 end_hdl
```

end handle

4.61.1.4 format

```
UINT8 format
```

UUID type

4.61.1.5 start_hdl

```
UINT16 start_hdl
```

start handle

4.61.1.6 uuid

```
UINT16 uuid[8]
```

UUID

4.62 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

4.62.1 Field Documentation

4.62.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.62.1.2 devid

```
UINT16 devid
```

device ID

4.62.1.3 handle

```
UINT16 handle
```

attribute handle

4.62.1.4 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.63 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

4.63.1 Field Documentation

4.63.1.1 att_err

```
UINT8 att_err
```

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

4.63.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.63.1.3 devid

```
UINT16 devid
```

device ID

4.63.1.4 handle

```
UINT16 handle
```

characteristic value handle

4.63.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.64 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

4.64.1 Field Documentation

4.64.1.1 att_err

```
UINT8 att_err
```

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

4.64.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.64.1.3 devid

```
UINT16 devid
```

device ID

4.64.1.4 handle

```
UINT16 handle
```

attribute handle

4.64.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.65 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

4.65.1 Field Documentation

4.65.1.1 att_err

```
UINT8 att_err
```

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

4.65.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.65.1.3 devid

```
UINT16 devid
```

device ID

4.65.1.4 handle

```
UINT16 handle
```

characteristic value handle

4.65.1.5 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.66 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

4.66.1 Field Documentation

4.66.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.66.1.2 devid

```
UINT16 devid
```

device ID

4.66.1.3 handle

```
UINT16 handle
```

attribute handle

4.66.1.4 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.67 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

4.67.1 Field Documentation

4.67.1.1 endHdl

```
UINT16 endHdl
```

end handle

4.67.1.2 pAttr

```
LE_GATT_ATTR_T* pAttr
```

pointer attribute table

4.67.1.3 startHdl

```
UINT16 startHdl
```

start handle

4.67.1.4 svc_id

```
UINT16 svc_id
```

service ID

4.68 LE_SMP_MSG_ENCRYPTION_CHANGE_IND_T Struct Reference

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

Data Fields

- UINT16 conn_hdl
- BOOL enable

4.68.1 Field Documentation

4.68.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.68.1.2 enable

```
BOOL enable
```

enable or disable

4.69 LE_SMP_MSG_ENCRYPTION_REFRESH_IND_T Struct Reference

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

Data Fields

- UINT16 conn_hdl
- UINT16 status

4.69.1 Field Documentation

4.69.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.69.1.2 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.70 LE_SMP_MSG_OOB_DATA_REQUEST_IND_T Struct Reference

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

Data Fields

- UINT16 conn_hdl

4.70.1 Field Documentation

4.70.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.71 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

4.71.1 Field Documentation

4.71.1.1 action

```
UINT8 action
```

refer to LE_SM_IO_CAP_∗ in ble_smp_if.h

4.71.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.71.1.3 lost_bond

```
BOOL lost_bond
```

remote lost bond

4.71.1.4 sc

```
UINT8 sc
```

secure connection

4.72 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

4.72.1 Field Documentation

4.72.1.1 authenticated

`UINT8 authenticated`

authenticated

4.72.1.2 bonded

`UINT8 bonded`

bonded

4.72.1.3 conn_hdl

`UINT16 conn_hdl`

connection handle

4.72.1.4 peer_id_addr

`LE_BT_ADDR_T peer_id_addr`

peer device address

4.72.1.5 sc

`UINT8 sc`

secure connection

4.72.1.6 status

```
UINT16 status
```

refer to LE_ERR_STATE in ble_err.h

4.73 LE_SMP_MSG_PASSKEY_DISPLAY_IND_T Struct Reference

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

Data Fields

- UINT16 conn_hdl
- UINT32 passkey

4.73.1 Field Documentation

4.73.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.73.1.2 passkey

```
UINT32 passkey
```

passkey

4.74 LE_SMP_MSG_PASSKEY_INPUT_IND_T Struct Reference

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

Data Fields

- UINT16 conn_hdl

4.74.1 Field Documentation

4.74.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.75 LE_SMP_MSG_SC_OOB_DATA_REQUEST_IND_T Struct Reference

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

Data Fields

- UINT16 conn_hdl

4.75.1 Field Documentation

4.75.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.76 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

4.76.1 Field Documentation

4.76.1.1 bondable

```
UINT8 bondable
```

bonding

4.76.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.76.1.3 keypress

```
UINT8 keypress
```

keypress status

4.76.1.4 mitm

```
UINT8 mitm
```

MITM

4.76.1.5 sc

```
UINT8 sc
```

secure connection

4.77 LE_SMP_MSG_USER_CONFIRM_IND_T Struct Reference

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

Data Fields

- UINT32 confirm_num
- UINT16 conn_hdl

4.77.1 Field Documentation

4.77.1.1 confirm_num

```
UINT32 confirm_num
```

confirm number

4.77.1.2 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.78 LE_SMP_SC_OOB_DATA_T Struct Reference

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

Data Fields

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

4.78.1 Field Documentation

4.78.1.1 confirm

```
UINT8 confirm[16]
```

confirm data

4.78.1.2 rand

```
UINT8 rand[16]
```

random data

4.79 LE_SYS_MSG_BUF_OVERFLOW_T Struct Reference

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

Data Fields

- UINT16 conn_hdl

4.79.1 Field Documentation

4.79.1.1 conn_hdl

```
UINT16 conn_hdl
```

connection handle

4.80 mw_blewifi_cbs_store_t Struct Reference

```
#include <controller_wifi_com.h>
```

Data Fields

- uint8_t manufacture_name [STA_INFO_MAX_MANUF_NAME_SIZE]

4.80.1 Field Documentation

4.80.1.1 manufacture_name

```
uint8_t manufacture_name[STA_INFO_MAX_MANUF_NAME_SIZE]
```

4.81 mw_wifi_auto_connect_ap_info_t Struct Reference

```
#include <controller_wifi_com.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]
- u8 hid_ssid_len
- u64 latest_beacon_rx_time
- s8 passphrase [64]
- u8 psk [32]
- u8 rsn_ie [256]
- s8 rssi
- s8 ssid [IEEE80211_MAX_SSID_LEN+1]
- u8 ssid_len
- u8 supported_rates [IEEE80211_MAX_SUPPORTED_RATES]
- wpa_ie_data_t wpa_data
- u8 wpa_ie [257]

4.81.1 Field Documentation

4.81.1.1 ap_channel

```
u8 ap_channel
```

4.81.1.2 beacon_interval

```
u16 beacon_interval
```

4.81.1.3 bssid

```
u8 bssid[MAC_ADDR_LEN]
```

4.81.1.4 capabilities

```
u16 capabilities
```

4.81.1.5 dtim_prod

```
u8 dtim_prod
```

4.81.1.6 fast_connect

```
u8 fast_connect
```

4.81.1.7 free_ocpy

```
bool free_ocpy
```

4.81.1.8 hid_ssid

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

4.81.1.9 hid_ssid_len

```
u8 hid_ssid_len
```

4.81.1.10 latest_beacon_rx_time

```
u64 latest_beacon_rx_time
```

4.81.1.11 passphrase

```
s8 passphrase[64]
```

4.81.1.12 psk

```
u8 psk[32]
```

4.81.1.13 rsn_ie

```
u8 rsn_ie[256]
```

4.81.1.14 rssi

```
s8 rssi
```

4.81.1.15 ssid

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

4.81.1.16 ssid_len

```
u8 ssid_len
```

4.81.1.17 supported_rates

```
u8 supported_rates[IEEE80211_MAX_SUPPORTED_RATES]
```

4.81.1.18 wpa_data

```
wpa_ie_data_t wpa_data
```

4.81.1.19 wpa_ie

```
u8 wpa_ie[257]
```

4.82 mw_wifi_sta_info_t Struct Reference

```
#include <controller_wifi_com.h>
```

Data Fields

- uint8_t au8Dot11MACAddress [MAC_ADDR_LEN]
- uint8_t u8SkipDtimPeriods

4.82.1 Field Documentation

4.82.1.1 au8Dot11MACAddress

```
uint8_t au8Dot11MACAddress[MAC_ADDR_LEN]
```

4.82.1.2 u8SkipDtimPeriods

```
uint8_t u8SkipDtimPeriods
```

4.83 MwFimAutoConnectCFG_t Struct Reference

```
#include <controller_wifi_com.h>
```

Data Fields

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

4.83.1 Field Documentation

4.83.1.1 flag

```
bool flag
```

4.83.1.2 front

```
s8 front
```

4.83.1.3 max_save_num

```
u8 max_save_num
```

4.83.1.4 rear

```
s8 rear
```

4.83.1.5 targetIdx

```
u8 targetIdx
```

4.84 rx_eapol_data Struct Reference

```
#include <controller_wifi_com.h>
```

Data Fields

- u8 frame_buffer [384]
- unsigned int frame_length

4.84.1 Field Documentation

4.84.1.1 frame_buffer

```
u8 frame_buffer[384]
```

4.84.1.2 frame_length

```
unsigned int frame_length
```

4.85 S_WIFI_MLME_SCAN_CFG Struct Reference

```
#include <controller_wifi_com.h>
```

Data Fields

- scan_report_t * ptScanReport
- E_WIFI_MLME_SCAN_TYPE tScanType
- uint32_t u32ActiveScanDur
- uint32_t u32PassiveScanDur
- uint8_t u8aBssid [MAC_ADDR_LEN]
- uint8_t u8aSsid [IEEE80211_MAX_SSID_LEN+1]
- uint8_t u8Channel
- uint8_t u8MaxScanApNum
- uint8_t u8ResendCnt

4.85.1 Detailed Description

The parameter of MLME_CMD_SCAN

4.85.2 Field Documentation

4.85.2.1 ptScanReport

```
scan_report_t* ptScanReport
```

The scan report which filled by MSQ, report to APS

4.85.2.2 tScanType

```
E_WIFI_MLME_SCAN_TYPE tScanType
```

scan type. active, passive, or mix mode

4.85.2.3 u32ActiveScanDur

```
uint32_t u32ActiveScanDur
```

Scan duration per scan counter in channel. units: millisecond

4.85.2.4 u32PassiveScanDur

```
uint32_t u32PassiveScanDur
```

Scan duration per channel. units: millisecond

4.85.2.5 u8aBssid

```
uint8_t u8aBssid[MAC_ADDR_LEN]
```

Not supported yet. MAC address of AP

4.85.2.6 u8aSsid

```
uint8_t u8aSsid[IEEE80211_MAX_SSID_LEN+1]
```

Not supported yet. SSID of AP

4.85.2.7 u8Channel

```
uint8_t u8Channel
```

Only specific channel or scan all channels

4.85.2.8 u8MaxScanApNum

```
uint8_t u8MaxScanApNum
```

Max scan AP number. When scanned AP number over this value, MSQ will drop the AP with smallest RSSI value

4.85.2.9 u8ResendCnt

```
uint8_t u8ResendCnt
```

Send probe req counter per channel when active scan. After send probe req, it will wait active scan time, and then send next probe req. The total time will be increased by a factor of this value

4.86 scan_info_t Struct Reference

```
#include <controller_wifi_com.h>
```

Data Fields

- uint8_t ap_channel
- uint16_t beacon_interval
- uint8_t bssid [MAC_ADDR_LEN]
- uint16_t capabilities
- uint8_t dtim_prod
- unsigned char free_ocpy
- uint64_t latest_beacon_rx_time
- u8 rsn_ie [256]
- int8_t rssi
- char ssid [IEEE80211_MAX_SSID_LEN+1]
- uint8_t ssid_len
- uint8_t supported_rates [IEEE80211_MAX_SUPPORTED_RATES]
- wpa_ie_data_t wpa_data
- u8 wpa_ie [257]

4.86.1 Field Documentation

4.86.1.1 ap_channel

```
uint8_t ap_channel
```

4.86.1.2 beacon_interval

```
uint16_t beacon_interval
```

4.86.1.3 bssid

```
uint8_t bssid[MAC_ADDR_LEN]
```

4.86.1.4 capabilities

```
uint16_t capabilities
```

4.86.1.5 dtim_prod

```
uint8_t dtim_prod
```

4.86.1.6 free_ocpy

```
unsigned char free_ocpy
```

4.86.1.7 latest_beacon_rx_time

```
uint64_t latest_beacon_rx_time
```

4.86.1.8 rsn_ie

```
u8 rsn_ie[256]
```

4.86.1.9 rssi

```
int8_t rssi
```

4.86.1.10 ssid

```
char ssid[IEEE80211_MAX_SSID_LEN+1]
```

4.86.1.11 ssid_len

```
uint8_t ssid_len
```

4.86.1.12 supported_rates

```
uint8_t supported_rates[IEEE80211_MAX_SUPPORTED_RATES]
```

4.86.1.13 wpa_data

```
wpa_ie_data_t wpa_data
```

4.86.1.14 wpa_ie

```
u8 wpa_ie[257]
```

4.87 scan_report_t Struct Reference

```
#include <controller_wifi_com.h>
```

Data Fields

- scan_info_t * pScanInfo
- u32 uScanApNum

4.87.1 Field Documentation

4.87.1.1 pScanInfo

```
scan_info_t* pScanInfo
```

4.87.1.2 uScanApNum

```
u32 uScanApNum
```

4.88 T_RfCmd Struct Reference

```
#include <controller_wifi.h>
```

Data Fields

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

4.88.1 Field Documentation

4.88.1.1 iArgc

```
int iArgc
```

4.88.1.2 saArgv

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

4.88.1.3 u32Type

```
uint32_t u32Type
```

4.89 T_RfDefEvt Struct Reference

```
#include <controller_wifi.h>
```

Data Fields

- uint32_t u32Type
- uint8_t u8aData [59]
- uint8_t u8Status

4.89.1 Field Documentation

4.89.1.1 u32Type

```
uint32_t u32Type
```

4.89.1.2 u8aData

```
uint8_t u8aData[59]
```

4.89.1.3 u8Status

```
uint8_t u8Status
```

4.90 T_RfEvt Struct Reference

```
#include <controller_wifi.h>
```

Data Fields

- int8_t i8Rssi
- 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 u8Phy
- uint8_t u8Pkt
- uint8_t u8Reserved
- uint8_t u8Status
- uint8_t u8Unicast

4.90.1 Field Documentation

4.90.1.1 i8Rssi

```
int8_t i8Rssi
```

4.90.1.2 pParam

```
void* pParam
```

4.90.1.3 u16RfMode

```
uint16_t u16RfMode
```

4.90.1.4 u16RxCnt

```
uint16_t u16RxCnt
```

4.90.1.5 u16RxCrcOkCnt

```
uint16_t u16RxCrcOkCnt
```

4.90.1.6 u32Freq

```
uint32_t u32Freq
```

4.90.1.7 u32Mode

```
uint32_t u32Mode
```

4.90.1.8 u32RfChannel

```
uint32_t u32RfChannel
```

4.90.1.9 u32Type

```
uint32_t u32Type
```

4.90.1.10 u8Freq

```
uint8_t u8Freq
```

4.90.1.11 u8IpcEnable

```
uint8_t u8IpcEnable
```

4.90.1.12 u8Len

```
uint8_t u8Len
```

4.90.1.13 u8Phy

```
uint8_t u8Phy
```

4.90.1.14 u8Pkt

```
uint8_t u8Pkt
```

4.90.1.15 u8Reserved

```
uint8_t u8Reserved
```

4.90.1.16 u8Status

```
uint8_t u8Status
```

4.90.1.17 u8Unicast

```
uint8_t u8Unicast
```

4.91 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

4.91.1 Detailed Description

Range of active scan times per channel.

4.91.2 Field Documentation

4.91.2.1 max

`uint32_t max`

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

4.91.2.2 min

`uint32_t min`

minimum active scan time per channel, units: millisecond

4.92 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

4.92.1 Detailed Description

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

4.92.2 Field Documentation

4.92.2.1 auth_mode

```
wifi_auth_mode_t auth_mode
```

The authentication mode.

4.92.2.2 beacon_interval

```
uint16_t beacon_interval
```

Beacon interval, 100 ∼ 60000 ms, default 100 ms

4.92.2.3 channel

```
uint8_t channel
```

The channel of Soft-AP.

4.92.2.4 encrypt_type

```
wifi_cipher_type_t encrypt_type
```

The encryption mode.

4.92.2.5 max_connection

```
uint8_t max_connection
```

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

4.92.2.6 password

```
uint8_t password[WIFI_LENGTH_PASSPHRASE]
```

The password of the Soft-AP.

4.92.2.7 password_length

```
uint8_t password_length
```

The length of the password.

4.92.2.8 ssid

```
uint8_t ssid[WIFI_MAX_LENGTH_OF_SSID]
```

The SSID of the Soft-AP.

4.92.2.9 ssid_hidden

```
uint8_t ssid_hidden
```

Broadcast SSID or not, default 0, broadcast the SSID

4.92.2.10 ssid_length

```
uint8_t ssid_length
```

The length of the SSID.

4.93 wifi_auto_connect_info_t Struct Reference

This structure is the Wi-Fi auto connect for save in the flash (FIM).

```
#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
- char hid_ssid [WIFI_MAX_LENGTH_OF_SSID]
- int8_t rssi
- char ssid [WIFI_MAX_LENGTH_OF_SSID]
- uint8_t supported_rates [WIFI_MAX_SUPPORTED_RATES]

4.93.1 Detailed Description

This structure is the Wi-Fi auto connect for save in the flash (FIM).

4.93.2 Field Documentation

4.93.2.1 ap_channel

```
uint8_t ap_channel
```

4.93.2.2 beacon_interval

```
uint16_t beacon_interval
```

4.93.2.3 bssid

```
uint8_t bssid[WIFI_MAC_ADDRESS_LENGTH]
```

4.93.2.4 capabilities

```
uint16_t capabilities
```

4.93.2.5 dtim_prod

```
uint8_t dtim_prod
```

4.93.2.6 fast_connect

```
uint8_t fast_connect
```

4.93.2.7 hid_ssid

```
char hid_ssid[WIFI_MAX_LENGTH_OF_SSID]
```

4.93.2.8 rssi

```
int8_t rssi
```

4.93.2.9 ssid

```
char ssid[WIFI_MAX_LENGTH_OF_SSID]
```

4.93.2.10 supported_rates

```
uint8_t supported_rates[WIFI_MAX_SUPPORTED_RATES]
```

4.94 wifi_cmd_t Struct Reference

```
#include <controller_wifi.h>
```

Data Fields

- u32 arg1:8
- u32 arg2:16
- u32 cmd_type:8
- void * prvData

4.94.1 Field Documentation

4.94.1.1 arg1

```
u32 arg1
```

4.94.1.2 arg2

```
u32 arg2
```

4.94.1.3 cmd_type

```
u32 cmd_type
```

4.94.1.4 prvData

```
void* prvData
```

4.95 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

4.95.1 Detailed Description

Wi-Fi configuration for initialization.

4.95.2 Field Documentation

4.95.2.1 ap_config

```
wifi_ap_config_t ap_config
```

The configurations for the AP. It should be set when the wifi_mode_t is WIFI_MODE_AP .

4.95.2.2 sta_config

```
wifi_sta_config_t sta_config
```

The configurations for the STA. It should be set when the wifi_mode_t is WIFI_MODE_STA.

4.96 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

4.96.1 Detailed Description

wifi_event_info_t

4.96.2 Field Documentation

4.96.2.1 connected

```
wifi_event_sta_connected_t connected
```

station connected to AP

4.96.2.2 disconnected

`wifi_event_sta_disconnected_t disconnected`

station disconnected to AP

4.96.2.3 got_ip

`wifi_event_sta_got_ip_t got_ip`

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

4.96.2.4 scan_done

`wifi_event_sta_scan_done_t scan_done`

station scan (APs) done

4.97 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

4.97.1 Detailed Description

wifi_event_sta_connected_t

4.97.2 Field Documentation

4.97.2.1 authmode

`wifi_auth_mode_t authmode`

4.97.2.2 bssid

```
uint8_t bssid[6]
```

BSSID of connected AP

4.97.2.3 channel

```
uint8_t channel
```

channel of connected AP

4.97.2.4 ssid

```
uint8_t ssid[32]
```

SSID of connected AP

4.97.2.5 ssid_len

```
uint8_t ssid_len
```

SSID length of connected AP

4.98 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

4.98.1 Detailed Description

wifi_event_sta_disconnected_t

4.98.2 Field Documentation

4.98.2.1 bssid

```
uint8_t bssid[6]
```

BSSID of disconnected AP

4.98.2.2 reason

```
uint8_t reason
```

reason of disconnection

4.98.2.3 ssid

```
uint8_t ssid[32]
```

SSID of disconnected AP

4.98.2.4 ssid_len

```
uint8_t ssid_len
```

SSID length of disconnected AP

4.99 wifi_event_sta_got_ip_t Struct Reference

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

Data Fields

- bool ip_changed

4.99.1 Field Documentation

4.99.1.1 ip_changed

```
bool ip_changed
```

4.100 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

4.100.1 Detailed Description

wifi_event_sta_scan_done_t

4.100.2 Field Documentation

4.100.2.1 number

```
uint8_t number
```

4.100.2.2 scan_id

```
uint8_t scan_id
```

4.100.2.3 status

```
uint32_t status
```

status of scanning APs

4.101 wifi_evt_t Struct Reference

```
#include <controller_wifi.h>
```

Data Fields

- uint32_t evt_type
- void * prvData

4.101.1 Field Documentation

4.101.1.1 evt_type

```
uint32_t evt_type
```

4.101.1.2 prvData

```
void* prvData
```

4.102 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

4.102.1 Detailed Description

Structure describing parameters for a Wi-Fi fast scan.

4.102.2 Field Documentation

4.102.2.1 authmode

```
wifi_auth_mode_t authmode
```

The weakest authmode to accept in the fast scan mode

4.102.2.2 rssi

```
int8_t rssi
```

The minimum rssi to accept in the fast scan mode

4.103 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

4.103.1 Detailed Description

WiFi stack configuration parameters.

4.103.2 Field Documentation

4.103.2.1 event_handler

```
wifi_event_notify_cb_t event_handler
```

WiFi event handler

4.103.2.2 magic

```
int magic
```

WiFi init magic number, it should be the last field

4.104 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

4.104.1 Detailed Description

Parameters for an SSID scan.

4.104.2 Field Documentation

4.104.2.1 bssid

```
uint8_t* bssid
```

MAC address of AP

4.104.2.2 channel

```
uint8_t channel
```

channel, scan the specific channel

4.104.2.3 scan_time

```
wifi_scan_time_t scan_time
```

scan time per channel

4.104.2.4 scan_type

```
wifi_scan_type_t scan_type
```

scan type, active or passive

4.104.2.5 show_hidden

```
bool show_hidden
```

enable to scan AP whose SSID is hidden

4.104.2.6 ssid

```
uint8_t* ssid
```

SSID of AP

4.105 wifi_scan_info_t Struct Reference

This structure defines the inforamtion of scanned APs.

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

Data Fields

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

4.105.1 Detailed Description

This structure defines the inforamtion of scanned APs.

4.105.2 Field Documentation

4.105.2.1 auth_mode

```
wifi_auth_mode_t auth_mode
```

Please refer to the definition of wifi_auth_mode_t.

4.105.2.2 beacon_interval

```
uint16_t beacon_interval
```

Indicates the beacon interval.

4.105.2.3 bssid

```
uint8_t bssid[WIFI_MAC_ADDRESS_LENGTH]
```

AP's MAC address.

4.105.2.4 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.

4.105.2.5 channel

```
uint8_t channel
```

The channel used.

4.105.2.6 dtim_period

```
uint8_t dtim_period
```

The DTIM Period indicates the number of beacon intervals between successive DTIMs. If all TIMs are DTIMs, the DTIM Period field has the value 1.

4.105.2.7 group_cipher

```
wifi_cipher_type_t group_cipher
```

group cipher of AP

4.105.2.8 pairwise_cipher

```
wifi_cipher_type_t pairwise_cipher
```

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

4.105.2.9 rssi

```
int rssi
```

Records the RSSI value when probe response is received.

4.105.2.10 ssid

```
uint8_t ssid[WIFI_MAX_LENGTH_OF_SSID]
```

Stores the predefined SSID.

4.105.2.11 ssid_length

```
uint8_t ssid_length
```

Length of the SSID.

4.106 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

4.106.1 Detailed Description

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

4.106.2 Field Documentation

4.106.2.1 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

4.106.2.2 num

```
int num
```

number of AP in the list

4.107 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

4.107.1 Detailed Description

Aggregate of active & passive scan time per channel.

4.107.2 Field Documentation

4.107.2.1 active

`wifi_active_scan_time_t active`

active scan time per channel, units: millisecond.

4.107.2.2 passive

`uint32_t passive`

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

4.108 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

4.108.1 Detailed Description

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

4.108.2 Field Documentation

4.108.2.1 bssid

```
uint8_t bssid[WIFI_MAC_ADDRESS_LENGTH]
```

The MAC address of the target AP.

4.108.2.2 bssid_present

```
uint8_t bssid_present
```

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

4.108.2.3 password

```
uint8_t password[WIFI_LENGTH_PASSPHRASE]
```

The password of the target AP.

4.108.2.4 password_length

```
uint8_t password_length
```

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

4.108.2.5 scan_method

```
wifi_scan_method_t scan_method
```

do all channel scan or fast scan

4.108.2.6 sort_method

```
wifi_sort_method_t sort_method
```

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

4.108.2.7 ssid

```
uint8_t ssid[WIFI_MAX_LENGTH_OF_SSID]
```

The SSID of the target AP.

4.108.2.8 ssid_length

```
uint8_t ssid_length
```

The length of the SSID.

4.108.2.9 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.

4.109 wifi_wpa_ie_data_t Struct Reference

This structure is the Wi-Fi auto connect with wpa information for save in the flash (FIM).

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

Data Fields

- int capabilities
- int group_cipher
- int key_mgmt
- int mgmt_group_cipher
- uint32_t num_pmkid
- int pairwise_cipher
- const uint8_t * pmkid
- int proto

4.109.1 Detailed Description

This structure is the Wi-Fi auto connect with wpa information for save in the flash (FIM).

4.109.2 Field Documentation

4.109.2.1 capabilities

```
int capabilities
```

4.109.2.2 group_cipher

```
int group_cipher
```

4.109.2.3 key_mgmt

```
int key_mgmt
```

4.109.2.4 mgmt_group_cipher

```
int mgmt_group_cipher
```

4.109.2.5 num_pmkid

```
uint32_t num_pmkid
```

4.109.2.6 pairwise_cipher

```
int pairwise_cipher
```

4.109.2.7 pmkid

```
const uint8_t* pmkid
```

4.109.2.8 proto

```
int proto
```

Index

_wpa_ie_data, 151
    capabilities, 151
    group_cipher, 151
    key_mgmt, 151
    mgmt_group_cipher, 152
    num_pmkid, 152
    pairwise_cipher, 152
    pmkid, 152
    proto, 152

action
    LE_SMP_MSG_PAIRING_ACTION_IND_T, 220
active
    wifi_scan_time_t, 258
addr
    LE_BT_ADDR_T, 161
    LE_CM_MSG_ADVERTISE_REPORT_IND_↩
        T, 164
addr_type
    LE_CM_MSG_ADVERTISE_REPORT_IND_↩
        T, 164
ap_channel
    auto_conn_info_t, 155
    mw_wifi_auto_connect_ap_info_t, 227
    scan_info_t, 234
    wifi_auto_connect_info_t, 244
ap_config
    wifi_config_t, 247
ap_record
    wifi_scan_list_t, 257
arg1
    wifi_cmd_t, 246
arg2
    wifi_cmd_t, 246
asso_data, 152
    eap_workaround, 153
    eapol_flags, 153
    group_cipher, 153
    key_mgmt, 153
    leap, 153
    mgmt_group_cipher, 153
    non_leap, 154
    pairwise_cipher, 154
    passphrase, 154
    proto, 154
    psk, 154
    psk_set, 154
att_err
    LE_GATT_MSG_CHARACTERISTIC_VAL_IND↩
        _T, 192
    LE_GATT_MSG_EXECUTE_WRITE_RELIABL↩
        E_CFM_T, 195
    LE_GATT_MSG_FIND_ALL_CHAR_DESC_CF↩
        M_T, 196
    LE_GATT_MSG_FIND_ALL_PRIMARY_SERVI↩
        CE_CFM_T, 197
    LE_GATT_MSG_FIND_CHARACTERISTIC_CF↩
        M_T, 198
    LE_GATT_MSG_FIND_INCLUDED_SERVICE_↩
        CFM_T, 199
    LE_GATT_MSG_FIND_PRIMARY_SERVICE_B↩
        Y_UUID_CFM_T, 200
    LE_GATT_MSG_PREPARE_WRITE_RELIABL↩
        E_CFM_T, 206
    LE_GATT_MSG_READ_CHAR_VAL_BY_UUID↩
        _CFM_T, 207
    LE_GATT_MSG_READ_CHARACTERISTIC_V↩
        ALUE_CFM_T, 208
    LE_GATT_MSG_READ_LONG_CHAR_VAL_C↩
        FM_T, 209
    LE_GATT_MSG_READ_MULTIPLE_CHAR_VA↩
        L_CFM_T, 210
    LE_GATT_MSG_WRITE_CHAR_VAL_RELIAB↩
        LE_CFM_T, 214
    LE_GATT_MSG_WRITE_CHAR_VALUE_CFM↩
        _T, 215
    LE_GATT_MSG_WRITE_LONG_CHAR_VALU↩
        E_CFM_T, 216
att_op
    LE_GATT_MSG_OPERATION_TIMEOUT_T, 205
au8Dot11MACAddress
    mw_wifi_sta_info_t, 230
auth_mode
    wifi_ap_config_t, 242
    wifi_scan_info_t, 255
authenticated
    LE_SMP_MSG_PAIRING_COMPLETE_IND_T,
        221
authmode
    wifi_event_sta_connected_t, 248
    wifi_fast_scan_threshold_t, 252
auto_conn_info_t, 154
    ap_channel, 155
    beacon_interval, 155
    bssid, 155
    capabilities, 155
    dtim_prod, 155
    fast_connect, 156
    free_ocpy, 156

hid_ssid, 156
hid_ssid_len, 156
latest_beacon_rx_time, 156
passphrase, 156
psk, 156
rsn_ie, 156
rssi, 157
ssid, 157
ssid_len, 157
supported_rates, 157
wpa_data, 157
wpa_ie, 157
auto_connect_cfg_t, 157
flag, 158
front, 158
max_save_num, 158
pFCInfo, 158
rear, 158
retryCount, 158
targetIdx, 159
uFCApNum, 159

BLE ALL APIs, 7
LeSmpGetBondIdFromAddr, 7
BLE CM APIs, 8
LE_CM_MSG_ADD_TO_RESOLVING_LIST_C↩ FM_T, 9
LE_CM_MSG_ADD_TO_WHITE_LIST_CFM_T, 9
LE_CM_MSG_CANCEL_CONNECTION_CFM_T, 9
LE_CM_MSG_CLEAR_RESOLVING_LIST_CF↩ M_T, 10
LE_CM_MSG_CLEAR_WHITE_LIST_CFM_T, 10
LE_CM_MSG_CREATE_CONNECTION_CFM_T, 10
LE_CM_MSG_ENTER_ADVERTISING_CFM_T, 10
LE_CM_MSG_ENTER_SCANNING_CFM_T, 10
LE_CM_MSG_EXIT_ADVERTISING_CFM_T, 10
LE_CM_MSG_EXIT_SCANNING_CFM_T, 10
LE_CM_MSG_PHY_UPDATE_COMPLETE_IN↩ D_T, 10
LE_CM_MSG_REMOVE_FROM_RESOLVING_↩ LIST_CFM_T, 11
LE_CM_MSG_REMOVE_FROM_WHITE_LIST↩ _CFM_T, 11
LE_CM_MSG_SET_ADVERTISING_DATA_CF↩ M_T, 11
LE_CM_MSG_SET_ADVERTISING_PARAMS_↩ CFM_T, 11
LE_CM_MSG_SET_CHANNEL_MAP_CFM_T, 11
LE_CM_MSG_SET_DEFAULT_PHY_CFM_T, 11
LE_CM_MSG_SET_RANDOM_ADDRESS_CF↩ M_T, 11
LE_CM_MSG_SET_RPA_TIMEOUT_CFM_T, 11
LE_CM_MSG_SET_SCAN_PARAMS_CFM_T, 12
LE_CM_MSG_SET_SCAN_RSP_DATA_CFM_T, 12
LeCmInit, 13

BLE GAP APIs, 15
GAP_ADTYPE_128BIT_COMPLETE, 17
GAP_ADTYPE_128BIT_MORE, 17
GAP_ADTYPE_16BIT_COMPLETE, 17
GAP_ADTYPE_16BIT_MORE, 18
GAP_ADTYPE_32BIT_COMPLETE, 18
GAP_ADTYPE_32BIT_MORE, 18
GAP_ADTYPE_3D_INFO_DATA, 18
GAP_ADTYPE_ADV_INTERVAL, 18
GAP_ADTYPE_APPEARANCE, 18
GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPO↩ RTED, 18
GAP_ADTYPE_FLAGS_GENERAL, 19
GAP_ADTYPE_FLAGS_LIMITED, 19
GAP_ADTYPE_FLAGS, 18
GAP_ADTYPE_LE_BD_ADDR, 19
GAP_ADTYPE_LE_ROLE, 19
GAP_ADTYPE_LOCAL_NAME_COMPLETE, 19
GAP_ADTYPE_LOCAL_NAME_SHORT, 19
GAP_ADTYPE_MANUFACTURER_SPECIFIC, 19
GAP_ADTYPE_OOB_CLASS_OF_DEVICE, 19
GAP_ADTYPE_OOB_SIMPLE_PAIRING_HAS↩ HC, 20
GAP_ADTYPE_OOB_SIMPLE_PAIRING_RAN↩ DR, 20
GAP_ADTYPE_POWER_LEVEL, 20
GAP_ADTYPE_PUBLIC_TARGET_ADDR, 20
GAP_ADTYPE_RANDOM_TARGET_ADDR, 20
GAP_ADTYPE_SERVICE_DATA_128BIT, 20
GAP_ADTYPE_SERVICE_DATA_32BIT, 20
GAP_ADTYPE_SERVICE_DATA, 20
GAP_ADTYPE_SERVICES_LIST_128BIT, 21
GAP_ADTYPE_SERVICES_LIST_16BIT, 21
GAP_ADTYPE_SIGNED_DATA, 21
GAP_ADTYPE_SIMPLE_PAIRING_HASHC_256, 21
GAP_ADTYPE_SIMPLE_PAIRING_RANDR_256, 21
GAP_ADTYPE_SLAVE_CONN_INTERVAL_RA↩ NGE, 21
GAP_ADTYPE_SM_OOB_FLAG, 21
GAP_ADTYPE_SM_TK, 21
GAP_PUBLIC_ADDR, 22
GAP_RAND_ADDR_NRPA, 22
GAP_RAND_ADDR_RPA, 22
GAP_RAND_ADDR_STATIC, 22
GAP_SCAN_TYPE_ACTIVE, 22
GAP_SCAN_TYPE_PASSIVE, 22
GAP_TX_PWR_CURR_VAL, 22
GAP_TX_PWR_MAX_VAL, 22
GAPBOND_IO_CAP_DISPLAY_ONLY, 23
GAPBOND_IO_CAP_DISPLAY_YES_NO, 23
GAPBOND_IO_CAP_KEYBOARD_DISPLAY, 23
GAPBOND_IO_CAP_KEYBOARD_ONLY, 23
GAPBOND_IO_CAP_NO_INPUT_NO_OUTPUT, 23
GAPBOND_PAIRING_MODE_INITIATE, 23
GAPBOND_PAIRING_MODE_NO_PAIRING, 23

GAPBOND_PAIRING_MODE_WAIT_FOR_REQ, 23
LE_GAP_ADV_MAX_SIZE, 24
LeGapAddToResolvingList, 24
LeGapAddToWhiteList, 24
LeGapAdvertisingEnable, 25
LeGapCentralConnectReq, 25
LeGapCentralSetDataChannel, 25
LeGapClearResolvingList, 27
LeGapClearWhiteList, 27
LeGapConnParaRequestRsp, 27
LeGapConnUpdateRequest, 28
LeGapConnUpdateResponse, 28
LeGapConnectCancelReq, 27
LeGapDisconnectReq, 29
LeGapGenRandAddr, 29
LeGapGetBtAddr, 29
LeGapReadAdvChannelTxPower, 29
LeGapReadChannelMap, 30
LeGapReadPhy, 30
LeGapReadResolvingListSize, 30
LeGapReadRssi, 30
LeGapReadTxPower, 31
LeGapReadWhiteListSize, 31
LeGapRemoveFromWhiteList, 31
LeGapScanningReq, 32
LeGapSetAdvData, 32
LeGapSetAdvParameter, 33
LeGapSetConnParameter, 33
LeGapSetDataChannelPduLen, 33
LeGapSetDefaultPhy, 34
LeGapSetPhy, 34
LeGapSetRandAddr, 34
LeGapSetRpaTimeout, 35
LeGapSetStaticAddr, 35
LeSetScanParameter, 35
LeSetScanRspData, 36
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_UUINT16, 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, 173
beacon_interval
auto_conn_info_t, 155
mw_wifi_auto_connect_ap_info_t, 227
scan_info_t, 234
wifi_ap_config_t, 242
wifi_auto_connect_info_t, 244
wifi_scan_info_t, 255
bondable
LE_SMP_MSG_SLAVE_SECURITY_REQUES↩
T_IND_T, 223
bonded
LE_SMP_MSG_PAIRING_COMPLETE_IND_T,
221
bssid
auto_conn_info_t, 155
mw_wifi_auto_connect_ap_info_t, 227

scan_info_t, 234
wifi_auto_connect_info_t, 244
wifi_event_sta_connected_t, 248
wifi_event_sta_disconnected_t, 249
wifi_scan_config_t, 254
wifi_scan_info_t, 255
wifi_sta_config_t, 259
bssid_present
wifi_sta_config_t, 259

CHAR_AGGREGATE_DESCRIPTOR
BLE GATT APIs, 41
CHAR_CLIENT_CONFIG_DESCRIPTOR
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
_wpa_ie_data, 151
auto_conn_info_t, 155
mw_wifi_auto_connect_ap_info_t, 227
scan_info_t, 234
wifi_auto_connect_info_t, 244
wifi_wpa_ie_data_t, 260
capability_info
wifi_scan_info_t, 256
ch_map
LE_CM_MSG_READ_CHANNEL_MAP_CFM_T, 174
channel
wifi_ap_config_t, 242
wifi_event_sta_connected_t, 249
wifi_scan_config_t, 254
wifi_scan_info_t, 256
channel_map
LE_GAP_ADVERTISING_PARAM_T, 182
client_rx_mtu
LE_GATT_MSG_EXCHANGE_MTU_IND_T, 194
cmd_type
wifi_cmd_t, 246
confirm
LE_SMP_SC_OOB_DATA_T, 225
confirm_num
LE_SMP_MSG_USER_CONFIRM_IND_T, 224

conn_hdl
LE_CM_CONNECTION_COMPLETE_IND_T, 162
LE_CM_MSG_CONN_PARA_REQ_T, 165
LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T, 166
LE_CM_MSG_DATA_LEN_CHANGE_IND_T, 167
LE_CM_MSG_DISCONNECT_COMPLETE_IND_T, 169
LE_CM_MSG_ENCRYPTION_CHANGE_IND_T, 169
LE_CM_MSG_ENCRYPTION_REFRESH_IND_T, 170
LE_CM_MSG_LTK_REQ_IND_T, 172
LE_CM_MSG_READ_CHANNEL_MAP_CFM_T, 174
LE_CM_MSG_READ_PHY_CFM_T, 175
LE_CM_MSG_READ_RSSI_CFM_T, 176
LE_CM_MSG_READ_TX_POWER_CFM_T, 177
LE_CM_MSG_SET_DATA_LENGTH_CFM_T, 178
LE_CM_MSG_SET_PHY_CFM_T, 179
LE_CM_MSG_SIGNAL_UPDATE_REQ_T, 180
LE_GATT_MSG_ACCESS_READ_IND_T, 187
LE_GATT_MSG_ACCESS_WRITE_IND_T, 188
LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND_T, 189
LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND_T, 190
LE_GATT_MSG_CHARACTERISTIC_VAL_IND_T, 192
LE_GATT_MSG_CONFIRMATION_CFM_T, 193
LE_GATT_MSG_EXCHANGE_MTU_CFM_T, 194
LE_GATT_MSG_EXCHANGE_MTU_IND_T, 194
LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM_T, 195
LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM_T, 196
LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_CFM_T, 197
LE_GATT_MSG_FIND_CHARACTERISTIC_CFM_T, 198
LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM_T, 199
LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM_T, 200
LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T, 201
LE_GATT_MSG_INDICATE_IND_T, 202
LE_GATT_MSG_NOTIFY_CFM_T, 203
LE_GATT_MSG_NOTIFY_IND_T, 204
LE_GATT_MSG_OPERATION_TIMEOUT_T, 205
LE_GATT_MSG_PREPARE_WRITE_RELIABLE_CFM_T, 206
LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_CFM_T, 207
LE_GATT_MSG_READ_CHARACTERISTIC_VALUE_CFM_T, 208
LE_GATT_MSG_READ_LONG_CHAR_VAL_C↩

FM_T, 209
LE_GATT_MSG_READ_MULTIPLE_CHAR_VA↩
L_CFM_T, 210
LE_GATT_MSG_SERVICE_INFO_IND_T, 212
LE_GATT_MSG_SIGNED_WRITE_CFM_T, 213
LE_GATT_MSG_WRITE_CHAR_VAL_RELIAB↩
LE_CFM_T, 214
LE_GATT_MSG_WRITE_CHAR_VALUE_CFM↩
_T, 215
LE_GATT_MSG_WRITE_LONG_CHAR_VALU↩
E_CFM_T, 216
LE_GATT_MSG_WRITE_NO_RSP_CFM_T, 217
LE_SMP_MSG_ENCRYPTION_CHANGE_IND↩
_T, 218
LE_SMP_MSG_ENCRYPTION_REFRESH_IND↩
_T, 219
LE_SMP_MSG_OOB_DATA_REQUEST_IND_T,
219
LE_SMP_MSG_PAIRING_ACTION_IND_T, 220
LE_SMP_MSG_PAIRING_COMPLETE_IND_T,
221
LE_SMP_MSG_PASSKEY_DISPLAY_IND_T, 222
LE_SMP_MSG_PASSKEY_INPUT_IND_T, 222
LE_SMP_MSG_SC_OOB_DATA_REQUEST_I↩
ND_T, 223
LE_SMP_MSG_SLAVE_SECURITY_REQUES↩
T_IND_T, 224
LE_SMP_MSG_USER_CONFIRM_IND_T, 225
LE_SYS_MSG_BUF_OVERFLOW_T, 226
conn_interval
LE_CM_CONNECTION_COMPLETE_IND_T, 162
conn_latency
LE_CM_CONNECTION_COMPLETE_IND_T, 162
connected
wifi_event_info_t, 247
current_rx_mtu
LE_GATT_MSG_EXCHANGE_MTU_CFM_T, 194

data
LE_CM_MSG_ADVERTISE_REPORT_IND_↩
T, 164
dev_id
LE_CM_CONNECTION_COMPLETE_IND_T, 162
devid
LE_CM_MSG_ENCRYPTION_CHANGE_IND_T,
170
LE_CM_MSG_ENCRYPTION_REFRESH_IND_T,
170
LE_CM_MSG_LTK_REQ_IND_T, 172
LE_GATT_MSG_ACCESS_READ_IND_T, 187
LE_GATT_MSG_ACCESS_WRITE_IND_T, 188
LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_↩
IND_T, 189
LE_GATT_MSG_CHARACTERISTIC_DECL_IN↩
FO_IND_T, 190
LE_GATT_MSG_CHARACTERISTIC_VAL_IND↩
_T, 192
LE_GATT_MSG_CONFIRMATION_CFM_T, 193
LE_GATT_MSG_EXCHANGE_MTU_CFM_T, 194
LE_GATT_MSG_EXCHANGE_MTU_IND_T, 195
LE_GATT_MSG_EXECUTE_WRITE_RELIABL↩
E_CFM_T, 195
LE_GATT_MSG_FIND_ALL_CHAR_DESC_CF↩
M_T, 196
LE_GATT_MSG_FIND_ALL_PRIMARY_SERVI↩
CE_CFM_T, 197
LE_GATT_MSG_FIND_CHARACTERISTIC_CF↩
M_T, 198
LE_GATT_MSG_FIND_INCLUDED_SERVICE_↩
CFM_T, 199
LE_GATT_MSG_FIND_PRIMARY_SERVICE_B↩
Y_UUID_CFM_T, 200
LE_GATT_MSG_INCLUDE_SERVICE_INFO_I↩
ND_T, 201
LE_GATT_MSG_INDICATE_IND_T, 203
LE_GATT_MSG_NOTIFY_CFM_T, 204
LE_GATT_MSG_NOTIFY_IND_T, 204
LE_GATT_MSG_OPERATION_TIMEOUT_T, 206
LE_GATT_MSG_PREPARE_WRITE_RELIABL↩
E_CFM_T, 206
LE_GATT_MSG_READ_CHAR_VAL_BY_UUID↩
_CFM_T, 207
LE_GATT_MSG_READ_CHARACTERISTIC_V↩
ALUE_CFM_T, 208
LE_GATT_MSG_READ_LONG_CHAR_VAL_C↩
FM_T, 209
LE_GATT_MSG_READ_MULTIPLE_CHAR_VA↩
L_CFM_T, 210
LE_GATT_MSG_SERVICE_INFO_IND_T, 212
LE_GATT_MSG_SIGNED_WRITE_CFM_T, 213
LE_GATT_MSG_WRITE_CHAR_VAL_RELIAB↩
LE_CFM_T, 214
LE_GATT_MSG_WRITE_CHAR_VALUE_CFM↩
_T, 215
LE_GATT_MSG_WRITE_LONG_CHAR_VALU↩
E_CFM_T, 216
LE_GATT_MSG_WRITE_NO_RSP_CFM_T, 217
direct_addr
LE_CM_MSG_DIRECT_ADV_REPORT_IND_T,
168
direct_addr_type
LE_CM_MSG_DIRECT_ADV_REPORT_IND_T,
168
disconnected
wifi_event_info_t, 247
dtim_period
wifi_scan_info_t, 256
dtim_prod
auto_conn_info_t, 155
mw_wifi_auto_connect_ap_info_t, 227
scan_info_t, 234
wifi_auto_connect_info_t, 244

eap_workaround
asso_data, 153
eapol_flags
asso_data, 153
ediv

LE_CM_MSG_LTK_REQ_IND_T, 172
enable
LE_SMP_MSG_ENCRYPTION_CHANGE_IND_T, 218
enabled
LE_CM_MSG_ENCRYPTION_CHANGE_IND_T, 170
LE_CM_MSG_ENCRYPTION_REFRESH_IND_T, 171
encrypt_type
wifi_ap_config_t, 242
end_hdl
LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T, 201
LE_GATT_MSG_SERVICE_INFO_IND_T, 212
endHdl
LE_GATT_SERVICE_T, 217
Enumeration, 145
wifi_auth_mode_t, 145
wifi_bandwidth_t, 146
wifi_cipher_type_t, 146
wifi_event_t, 146
wifi_mac_data_rate_t, 147
wifi_mode_t, 147
wifi_reason_code_t, 148
wifi_scan_method_t, 149
wifi_scan_type_t, 149
wifi_sort_method_t, 149
err_hdl
LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM_T, 195
LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL_CFM_T, 211
event
event_msg_t, 159
event_handler
wifi_init_config_t, 253
event_msg_t, 159
event, 159
length, 159
param, 160
event_type
LE_CM_MSG_ADVERTISE_REPORT_IND_T, 164
evt_type
wifi_evt_t, 252

fast_connect
auto_conn_info_t, 156
mw_wifi_auto_connect_ap_info_t, 228
wifi_auto_connect_info_t, 245
filter_policy
LE_GAP_ADVERTISING_PARAM_T, 182
LE_GAP_SCAN_PARAM_T, 185
flag
auto_connect_cfg_t, 158
LE_GATT_MSG_ACCESS_WRITE_IND_T, 188
MwFimAutoConnectCFG_t, 230
format
LE_GATT_ATTR_T, 186
LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND_T, 189
LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND_T, 190
LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T, 202
LE_GATT_MSG_SERVICE_INFO_IND_T, 212
frame_buffer
rx_eapol_data, 231
frame_length
rx_eapol_data, 231
free_ocpy
auto_conn_info_t, 156
mw_wifi_auto_connect_ap_info_t, 228
scan_info_t, 235
front
auto_connect_cfg_t, 158
MwFimAutoConnectCFG_t, 230

GAP_ADTYPE_128BIT_COMPLETE
BLE GAP APIs, 17
GAP_ADTYPE_128BIT_MORE
BLE GAP APIs, 17
GAP_ADTYPE_16BIT_COMPLETE
BLE GAP APIs, 17
GAP_ADTYPE_16BIT_MORE
BLE GAP APIs, 18
GAP_ADTYPE_32BIT_COMPLETE
BLE GAP APIs, 18
GAP_ADTYPE_32BIT_MORE
BLE GAP APIs, 18
GAP_ADTYPE_3D_INFO_DATA
BLE GAP APIs, 18
GAP_ADTYPE_ADV_INTERVAL
BLE GAP APIs, 18
GAP_ADTYPE_APPEARANCE
BLE GAP APIs, 18
GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED
BLE GAP APIs, 18
GAP_ADTYPE_FLAGS_GENERAL
BLE GAP APIs, 19
GAP_ADTYPE_FLAGS_LIMITED
BLE GAP APIs, 19
GAP_ADTYPE_FLAGS
BLE GAP APIs, 18
GAP_ADTYPE_LE_BD_ADDR
BLE GAP APIs, 19
GAP_ADTYPE_LE_ROLE
BLE GAP APIs, 19
GAP_ADTYPE_LOCAL_NAME_COMPLETE
BLE GAP APIs, 19
GAP_ADTYPE_LOCAL_NAME_SHORT
BLE GAP APIs, 19
GAP_ADTYPE_MANUFACTURER_SPECIFIC
BLE GAP APIs, 19
GAP_ADTYPE_OOB_CLASS_OF_DEVICE
BLE GAP APIs, 19
GAP_ADTYPE_OOB_SIMPLE_PAIRING_HASHC

BLE GAP APIs, 20
GAP_ADTYPE_OOB_SIMPLE_PAIRING_RANDR
BLE GAP APIs, 20
GAP_ADTYPE_POWER_LEVEL
BLE GAP APIs, 20
GAP_ADTYPE_PUBLIC_TARGET_ADDR
BLE GAP APIs, 20
GAP_ADTYPE_RANDOM_TARGET_ADDR
BLE GAP APIs, 20
GAP_ADTYPE_SERVICE_DATA_128BIT
BLE GAP APIs, 20
GAP_ADTYPE_SERVICE_DATA_32BIT
BLE GAP APIs, 20
GAP_ADTYPE_SERVICE_DATA
BLE GAP APIs, 20
GAP_ADTYPE_SERVICES_LIST_128BIT
BLE GAP APIs, 21
GAP_ADTYPE_SERVICES_LIST_16BIT
BLE GAP APIs, 21
GAP_ADTYPE_SIGNED_DATA
BLE GAP APIs, 21
GAP_ADTYPE_SIMPLE_PAIRING_HASHC_256
BLE GAP APIs, 21
GAP_ADTYPE_SIMPLE_PAIRING_RANDR_256
BLE GAP APIs, 21
GAP_ADTYPE_SLAVE_CONN_INTERVAL_RANGE
BLE GAP APIs, 21
GAP_ADTYPE_SM_OOB_FLAG
BLE GAP APIs, 21
GAP_ADTYPE_SM_TK
BLE GAP APIs, 21
GAP_PUBLIC_ADDR
BLE GAP APIs, 22
GAP_RAND_ADDR_NRPA
BLE GAP APIs, 22
GAP_RAND_ADDR_RPA
BLE GAP APIs, 22
GAP_RAND_ADDR_STATIC
BLE GAP APIs, 22
GAP_SCAN_TYPE_ACTIVE
BLE GAP APIs, 22
GAP_SCAN_TYPE_PASSIVE
BLE GAP APIs, 22
GAP_TX_PWR_CURR_VAL
BLE GAP APIs, 22
GAP_TX_PWR_MAX_VAL
BLE GAP APIs, 22
GAPBOND_IO_CAP_DISPLAY_ONLY
BLE GAP APIs, 23
GAPBOND_IO_CAP_DISPLAY_YES_NO
BLE GAP APIs, 23
GAPBOND_IO_CAP_KEYBOARD_DISPLAY
BLE GAP APIs, 23
GAPBOND_IO_CAP_KEYBOARD_ONLY
BLE GAP APIs, 23
GAPBOND_IO_CAP_NO_INPUT_NO_OUTPUT
BLE GAP APIs, 23
GAPBOND_PAIRING_MODE_INITIATE
BLE GAP APIs, 23
GAPBOND_PAIRING_MODE_NO_PAIRING
BLE GAP APIs, 23
GAPBOND_PAIRING_MODE_WAIT_FOR_REQ
BLE GAP APIs, 23
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, 248
group_cipher
_wpa_ie_data, 151
asso_data, 153
wifi_scan_info_t, 256
wifi_wpa_ie_data_t, 261

handle
LE_CM_MSG_SET_DISCONNECT_CFM_T, 179
LE_GATT_ATTR_T, 186
LE_GATT_MSG_ACCESS_READ_IND_T, 187
LE_GATT_MSG_ACCESS_WRITE_IND_T, 188
LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND_T, 189
LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND_T, 191
LE_GATT_MSG_CHARACTERISTIC_VAL_IND_T, 192
LE_GATT_MSG_CONFIRMATION_CFM_T, 193
LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM_T, 196
LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_CFM_T, 197
LE_GATT_MSG_FIND_CHARACTERISTIC_CFM_T, 198
LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM_T, 199
LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM_T, 200
LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T, 202
LE_GATT_MSG_INDICATE_IND_T, 203
LE_GATT_MSG_NOTIFY_CFM_T, 204
LE_GATT_MSG_NOTIFY_IND_T, 205
LE_GATT_MSG_PREPARE_WRITE_RELIABLE_CFM_T, 206
LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_CFM_T, 207
LE_GATT_MSG_READ_CHARACTERISTIC_VALUE_CFM_T, 208
LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM_T, 209
LE_GATT_MSG_SIGNED_WRITE_CFM_T, 213
LE_GATT_MSG_WRITE_CHAR_VAL_RELIABLE_CFM_T, 214
LE_GATT_MSG_WRITE_CHAR_VALUE_CFM_T, 215
LE_GATT_MSG_WRITE_LONG_CHAR_VALUE_CFM_T, 216
LE_GATT_MSG_WRITE_NO_RSP_CFM_T, 217
hap_ap_info
hap_control_t, 160
hap_bitvector
hap_control_t, 160
hap_control_t, 160
hap_ap_info, 160
hap_bitvector, 160
hap_en, 160
hap_final_index, 160
hap_index, 161
hap_ssid, 161
hap_en
hap_control_t, 160
hap_final_index
hap_control_t, 160
hap_index
hap_control_t, 161
hap_ssid
hap_control_t, 161
hid_ssid
auto_conn_info_t, 156
mw_wifi_auto_connect_ap_info_t, 228
wifi_auto_connect_info_t, 245
hid_ssid_len
auto_conn_info_t, 156
mw_wifi_auto_connect_ap_info_t, 228

i8Rssi
T_RfEvt, 238
iArgc
T_RfCmd, 237
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, 180
interval
LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T, 166
LE_GAP_SCAN_PARAM_T, 185
interval_max
LE_CM_MSG_SIGNAL_UPDATE_REQ_T, 180
LE_GAP_ADVERTISING_PARAM_T, 183
LE_GAP_CONN_PARAM_T, 184
interval_min
LE_CM_MSG_SIGNAL_UPDATE_REQ_T, 180
LE_GAP_ADVERTISING_PARAM_T, 183
LE_GAP_CONN_PARAM_T, 184
ip_changed
wifi_event_sta_got_ip_t, 250
itv_max
LE_CM_MSG_CONN_PARA_REQ_T, 165
LE_CONN_PARA_T, 181
itv_min
LE_CM_MSG_CONN_PARA_REQ_T, 165
LE_CONN_PARA_T, 181

key_mgmt
_wpa_ie_data, 151
asso_data, 153
wifi_wpa_ie_data_t, 261
keypress

LE_SMP_MSG_SLAVE_SECURITY_REQUES↩
T_IND_T, 224

LE_ATT_MSG_BASE
BLE MSG APIs, 72
LE_ATT_UUID_SIZE
BLE GATT APIs, 45
LE_BT_ADDR_T, 161
addr, 161
type, 161
LE_CM_CONNECTION_COMPLETE_IND_T, 162
conn_hdl, 162
conn_interval, 162
conn_latency, 162
dev_id, 162
peer_addr, 162
peer_addr_type, 163
role, 163
status, 163
supervison_timeout, 163
LE_CM_MSG_ADD_TO_RESOLVING_LIST_CFM_T
BLE CM APIs, 9
LE_CM_MSG_ADD_TO_WHITE_LIST_CFM_T
BLE CM APIs, 9
LE_CM_MSG_ADVERTISE_REPORT_IND_T, 163
addr, 164
addr_type, 164
data, 164
event_type, 164
len, 164
rssi, 164
LE_CM_MSG_BASE
BLE MSG APIs, 72
LE_CM_MSG_CANCEL_CONNECTION_CFM_T
BLE CM APIs, 9
LE_CM_MSG_CLEAR_RESOLVING_LIST_CFM_T
BLE CM APIs, 10
LE_CM_MSG_CLEAR_WHITE_LIST_CFM_T
BLE CM APIs, 10
LE_CM_MSG_CONN_PARA_REQ_T, 164
conn_hdl, 165
itv_max, 165
itv_min, 165
latency, 165
sv_tmo, 165
LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T,
165
conn_hdl, 166
interval, 166
latency, 166
status, 166
supervision_timeout, 166
LE_CM_MSG_CREATE_CONNECTION_CFM_T
BLE CM APIs, 10
LE_CM_MSG_DATA_LEN_CHANGE_IND_T, 166
conn_hdl, 167
max_rx_octets, 167
max_rx_time, 167
max_tx_octets, 167
max_tx_time, 167
LE_CM_MSG_DIRECT_ADV_REPORT_IND_T, 167
direct_addr, 168
direct_addr_type, 168
peer_addr, 168
peer_addr_type, 168
rssi, 168
LE_CM_MSG_DISCONNECT_COMPLETE_IND_T,
168
conn_hdl, 169
reason, 169
status, 169
LE_CM_MSG_ENCRYPTION_CHANGE_IND_T, 169
conn_hdl, 169
devid, 170
enabled, 170
status, 170
LE_CM_MSG_ENCRYPTION_REFRESH_IND_T, 170
conn_hdl, 170
devid, 170
enabled, 171
status, 171
LE_CM_MSG_ENTER_ADVERTISING_CFM_T
BLE CM APIs, 10
LE_CM_MSG_ENTER_SCANNING_CFM_T
BLE CM APIs, 10
LE_CM_MSG_EXIT_ADVERTISING_CFM_T
BLE CM APIs, 10
LE_CM_MSG_EXIT_SCANNING_CFM_T
BLE CM APIs, 10
LE_CM_MSG_INIT_COMPLETE_CFM_T, 171
status, 171
LE_CM_MSG_LTK_REQ_IND_T, 171
conn_hdl, 172
devid, 172
ediv, 172
rand, 172
LE_CM_MSG_PHY_UPDATE_COMPLETE_IND_T
BLE CM APIs, 10
LE_CM_MSG_READ_ADV_TX_POWER_CFM_T, 172
pwr_level, 173
status, 173
LE_CM_MSG_READ_BD_ADDR_CFM_T, 173
bd_addr, 173
status, 173
LE_CM_MSG_READ_CHANNEL_MAP_CFM_T, 174
ch_map, 174
conn_hdl, 174
status, 174
LE_CM_MSG_READ_PHY_CFM_T, 174
conn_hdl, 175
rx_phy, 175
status, 175
tx_phy, 175
LE_CM_MSG_READ_RESOLVING_LIST_SIZE_CF↩
M_T, 175
size, 175
status, 175

LE_CM_MSG_READ_RSSI_CFM_T, 176
conn_hdl, 176
rssi, 176
status, 176
LE_CM_MSG_READ_TX_POWER_CFM_T, 176
conn_hdl, 177
status, 177
tx_power, 177
LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM_T, 177
size, 177
status, 178
LE_CM_MSG_REMOVE_FROM_RESOLVING_LIST_CFM_T
BLE CM APIs, 11
LE_CM_MSG_REMOVE_FROM_WHITE_LIST_CFM_T
BLE CM APIs, 11
LE_CM_MSG_SET_ADVERTISING_DATA_CFM_T
BLE CM APIs, 11
LE_CM_MSG_SET_ADVERTISING_PARAMS_CFM_T
BLE CM APIs, 11
LE_CM_MSG_SET_CHANNEL_MAP_CFM_T
BLE CM APIs, 11
LE_CM_MSG_SET_DATA_LENGTH_CFM_T, 178
conn_hdl, 178
status, 178
LE_CM_MSG_SET_DEFAULT_PHY_CFM_T
BLE CM APIs, 11
LE_CM_MSG_SET_DISCONNECT_CFM_T, 178
handle, 179
status, 179
LE_CM_MSG_SET_PHY_CFM_T, 179
conn_hdl, 179
status, 179
LE_CM_MSG_SET_RANDOM_ADDRESS_CFM_T
BLE CM APIs, 11
LE_CM_MSG_SET_RPA_TIMEOUT_CFM_T
BLE CM APIs, 11
LE_CM_MSG_SET_SCAN_PARAMS_CFM_T
BLE CM APIs, 12
LE_CM_MSG_SET_SCAN_RSP_DATA_CFM_T
BLE CM APIs, 12
LE_CM_MSG_SIGNAL_UPDATE_REQ_T, 180
conn_hdl, 180
identifier, 180
interval_max, 180
interval_min, 180
slave_latency, 180
timeout_multiplier, 180
LE_CM_REQ_STATUS_T, 181
status, 181
LE_CONN_PARA_T, 181
itv_max, 181
itv_min, 181
latency, 182
sv_timeout, 182
LE_GAP_ADV_MAX_SIZE
BLE GAP APIs, 24
LE_GAP_ADVERTISING_PARAM_T, 182
channel_map, 182
filter_policy, 182
interval_max, 183
interval_min, 183
own_addr_type, 183
peer_addr, 183
peer_addr_type, 183
type, 183
LE_GAP_CONN_PARAM_T, 183
interval_max, 184
interval_min, 184
latency, 184
supervision_timeout, 184
LE_GAP_SCAN_PARAM_T, 184
filter_policy, 185
interval, 185
own_addr_type, 185
type, 185
window, 185
LE_GATT_ATTR_T, 185
format, 186
handle, 186
len, 186
maxLen, 186
pUuid, 186
pVal, 186
permit, 186
LE_GATT_CHAR_PROP_AUTH
BLE GATT APIs, 46
LE_GATT_CHAR_PROP_BCAST
BLE GATT APIs, 46
LE_GATT_CHAR_PROP_EXT_PROP
BLE GATT APIs, 46
LE_GATT_CHAR_PROP_IND
BLE GATT APIs, 46
LE_GATT_CHAR_PROP_NTF
BLE GATT APIs, 46
LE_GATT_CHAR_PROP_RD
BLE GATT APIs, 46
LE_GATT_CHAR_PROP_WR_NO_RESP
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, 187
conn_hdl, 187
devid, 187
handle, 187
offset, 187
LE_GATT_MSG_ACCESS_WRITE_IND_T, 187
conn_hdl, 188
devid, 188
flag, 188
handle, 188
len, 188
offset, 188
pVal, 189
LE_GATT_MSG_BASE
BLE MSG APIs, 72
LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND_T, 189
conn_hdl, 189
devid, 189
format, 189
handle, 189
uuid, 190
LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND_T, 190
conn_hdl, 190
devid, 190
format, 190
handle, 191
property, 191
uuid, 191
val_hdl, 191
LE_GATT_MSG_CHARACTERISTIC_VAL_IND_T, 191
att_err, 192
conn_hdl, 192
devid, 192
handle, 192
len, 192
offset, 192
val, 192
LE_GATT_MSG_CONFIRMATION_CFM_T, 193
conn_hdl, 193
devid, 193
handle, 193
LE_GATT_MSG_EXCHANGE_MTU_CFM_T, 193
conn_hdl, 194
current_rx_mtu, 194
devid, 194
LE_GATT_MSG_EXCHANGE_MTU_IND_T, 194
client_rx_mtu, 194
conn_hdl, 194
devid, 195
LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM_T, 195
att_err, 195
conn_hdl, 195
devid, 195
err_hdl, 195
status, 196
LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM_T, 196
att_err, 196
conn_hdl, 196
devid, 196
handle, 196
status, 197
LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_CFM_T, 197
att_err, 197
conn_hdl, 197
devid, 197
handle, 197
status, 198
LE_GATT_MSG_FIND_CHARACTERISTIC_CFM_T, 198
att_err, 198
conn_hdl, 198
devid, 198
handle, 198
status, 199
LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM_T, 199
att_err, 199
conn_hdl, 199
devid, 199
handle, 199
status, 200
LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM_T, 200
att_err, 200
conn_hdl, 200
devid, 200
handle, 200
status, 201
LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T, 201
conn_hdl, 201
devid, 201
end_hdl, 201
format, 202
handle, 202
start_hdl, 202
uuid, 202
LE_GATT_MSG_INDICATE_IND_T, 202
conn_hdl, 202
devid, 203
handle, 203
len, 203
val, 203
LE_GATT_MSG_NOTIFY_CFM_T, 203
conn_hdl, 203
devid, 204
handle, 204
status, 204
LE_GATT_MSG_NOTIFY_IND_T, 204

conn_hdl, 204
devid, 204
handle, 205
len, 205
val, 205
LE_GATT_MSG_OPERATION_TIMEOUT_T, 205
att_op, 205
conn_hdl, 205
devid, 206
LE_GATT_MSG_PREPARE_WRITE_RELIABLE_CF↩ M_T, 206
att_err, 206
conn_hdl, 206
devid, 206
handle, 206
status, 207
LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_CF↩ M_T, 207
att_err, 207
conn_hdl, 207
devid, 207
handle, 207
status, 208
LE_GATT_MSG_READ_CHARACTERISTIC_VALU↩ E_CFM_T, 208
att_err, 208
conn_hdl, 208
devid, 208
handle, 208
status, 209
LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM_T, 209
att_err, 209
conn_hdl, 209
devid, 209
handle, 209
status, 210
LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL_C↩ FM_T, 210
att_err, 210
conn_hdl, 210
devid, 210
err_hdl, 211
len, 211
status, 211
val, 211
LE_GATT_MSG_SERVICE_INFO_IND_T, 211
conn_hdl, 212
devid, 212
end_hdl, 212
format, 212
start_hdl, 212
uuid, 212
LE_GATT_MSG_SIGNED_WRITE_CFM_T, 212
conn_hdl, 213
devid, 213
handle, 213
status, 213
LE_GATT_MSG_WRITE_CHAR_VAL_RELIABLE_C↩ FM_T, 213
att_err, 214
conn_hdl, 214
devid, 214
handle, 214
status, 214
LE_GATT_MSG_WRITE_CHAR_VALUE_CFM_T, 214
att_err, 215
conn_hdl, 215
devid, 215
handle, 215
status, 215
LE_GATT_MSG_WRITE_LONG_CHAR_VALUE_CF↩ M_T, 215
att_err, 216
conn_hdl, 216
devid, 216
handle, 216
status, 216
LE_GATT_MSG_WRITE_NO_RSP_CFM_T, 216
conn_hdl, 217
devid, 217
handle, 217
status, 217
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, 217
endHdl, 217
pAttr, 218
startHdl, 218
svc_id, 218
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, 218
conn_hdl, 218
enable, 218
LE_SMP_MSG_ENCRYPTION_REFRESH_IND_T, 219
conn_hdl, 219
status, 219
LE_SMP_MSG_OOB_DATA_REQUEST_IND_T, 219
conn_hdl, 219
LE_SMP_MSG_PAIRING_ACTION_IND_T, 220
action, 220
conn_hdl, 220
lost_bond, 220
sc, 220
LE_SMP_MSG_PAIRING_COMPLETE_IND_T, 221
authenticated, 221
bonded, 221
conn_hdl, 221
peer_id_addr, 221
sc, 221
status, 221
LE_SMP_MSG_PASSKEY_DISPLAY_IND_T, 222
conn_hdl, 222
passkey, 222
LE_SMP_MSG_PASSKEY_INPUT_IND_T, 222
conn_hdl, 222
LE_SMP_MSG_SC_OOB_DATA_REQUEST_IND_T, 223
conn_hdl, 223
LE_SMP_MSG_SLAVE_SECURITY_REQUEST_IND_T, 223
bondable, 223
conn_hdl, 224
keypress, 224
mitm, 224
sc, 224
LE_SMP_MSG_USER_CONFIRM_IND_T, 224
confirm_num, 224
conn_hdl, 225
LE_SMP_SC_OOB_DATA_T, 225
confirm, 225
rand, 225
LE_SYS_MSG_BASE
BLE MSG APIs, 73
LE_SYS_MSG_BUF_OVERFLOW_T, 225
conn_hdl, 226
latency
LE_CM_MSG_CONN_PARA_REQ_T, 165
LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T, 166
LE_CONN_PARA_T, 182
LE_GAP_CONN_PARAM_T, 184
latest_beacon_rx_time
auto_conn_info_t, 156
mw_wifi_auto_connect_ap_info_t, 228
scan_info_t, 235
LeCancelAllMessage
BLE MSG APIs, 76
LeCancelAllSubMessage
BLE MSG APIs, 77
LeCancelFirstMessage
BLE MSG APIs, 77
LeCancelFirstSubMessage
BLE MSG APIs, 77
LeCmInit
BLE CM APIs, 13
LeGapAddToResolvingList
BLE GAP APIs, 24
LeGapAddToWhiteList
BLE GAP APIs, 24
LeGapAdvertisingEnable
BLE GAP APIs, 25
LeGapCentralConnectReq
BLE GAP APIs, 25
LeGapCentralSetDataChannel
BLE GAP APIs, 25

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
LeGapReadPhy
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
LeGapSetDefaultPhy
BLE GAP APIs, 34
LeGapSetPhy
BLE GAP APIs, 34
LeGapSetRandAddr
BLE GAP APIs, 34
LeGapSetRpaTimeout
BLE GAP APIs, 35
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, 36
LeSmpGetBondIdFromAddr
BLE ALL APIs, 7
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
leap
asso_data, 153
len
LE_CM_MSG_ADVERTISE_REPORT_IND_↩
T, 164
LE_GATT_ATTR_T, 186
LE_GATT_MSG_ACCESS_WRITE_IND_T, 188
LE_GATT_MSG_CHARACTERISTIC_VAL_IND↩
_T, 192
LE_GATT_MSG_INDICATE_IND_T, 203
LE_GATT_MSG_NOTIFY_IND_T, 205
LE_GATT_MSG_READ_MULTIPLE_CHAR_VA↩
L_CFM_T, 211
length
event_msg_t, 159
lost_bond
LE_SMP_MSG_PAIRING_ACTION_IND_T, 220

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, 253
manufacture_name
mw_blewifi_cbs_store_t, 226
max
wifi_active_scan_time_t, 241
max_connection
wifi_ap_config_t, 242
max_rx_octets
LE_CM_MSG_DATA_LEN_CHANGE_IND_T, 167
max_rx_time
LE_CM_MSG_DATA_LEN_CHANGE_IND_T, 167
max_save_num
auto_connect_cfg_t, 158
MwFimAutoConnectCFG_t, 231
max_tx_octets
LE_CM_MSG_DATA_LEN_CHANGE_IND_T, 167
max_tx_time
LE_CM_MSG_DATA_LEN_CHANGE_IND_T, 167
maxLen
LE_GATT_ATTR_T, 186
mgmt_group_cipher
_wpa_ie_data, 152
asso_data, 153
wifi_wpa_ie_data_t, 261
min
wifi_active_scan_time_t, 241
mitm
LE_SMP_MSG_SLAVE_SECURITY_REQUES↩
T_IND_T, 224

MsgData
BLE MSG APIs, 75
MsgLock
BLE MSG APIs, 75
mw_blewifi_cbs_store_t, 226
manufacture_name, 226
mw_wifi_auto_connect_ap_info_t, 226
ap_channel, 227
beacon_interval, 227
bssid, 227
capabilities, 227
dtim_prod, 227
fast_connect, 228
free_ocpy, 228
hid_ssid, 228
hid_ssid_len, 228
latest_beacon_rx_time, 228
passphrase, 228
psk, 228
rsn_ie, 228
rssi, 229
ssid, 229
ssid_len, 229
supported_rates, 229
wpa_data, 229
wpa_ie, 229
mw_wifi_sta_info_t, 229
au8Dot11MACAddress, 230
u8SkipDtimPeriods, 230
MwFimAutoConnectCFG_t, 230
flag, 230
front, 230
max_save_num, 231
rear, 231
targetIdx, 231

non_leap
asso_data, 154
num
wifi_scan_list_t, 257
num_pmkid
_wpa_ie_data, 152
wifi_wpa_ie_data_t, 261
number
wifi_event_sta_scan_done_t, 251

offset
LE_GATT_MSG_ACCESS_READ_IND_T, 187
LE_GATT_MSG_ACCESS_WRITE_IND_T, 188
LE_GATT_MSG_CHARACTERISTIC_VAL_IND_T, 192
own_addr_type
LE_GAP_ADVERTISING_PARAM_T, 183
LE_GAP_SCAN_PARAM_T, 185

pAttr
LE_GATT_SERVICE_T, 218
pFCInfo
auto_connect_cfg_t, 158
pParam
T_RfEvt, 238
PRIMARY_SERVICE_DECL_UUID128
BLE GATT APIs, 50
PRIMARY_SERVICE_DECL_UUID16
BLE GATT APIs, 50
pScanInfo
scan_report_t, 236
pUuid
LE_GATT_ATTR_T, 186
pVal
LE_GATT_ATTR_T, 186
LE_GATT_MSG_ACCESS_WRITE_IND_T, 189
pairwise_cipher
_wpa_ie_data, 152
asso_data, 154
wifi_scan_info_t, 256
wifi_wpa_ie_data_t, 261
param
event_msg_t, 160
passive
wifi_scan_time_t, 258
passkey
LE_SMP_MSG_PASSKEY_DISPLAY_IND_T, 222
passphrase
asso_data, 154
auto_conn_info_t, 156
mw_wifi_auto_connect_ap_info_t, 228
password
wifi_ap_config_t, 243
wifi_sta_config_t, 259
password_length
wifi_ap_config_t, 243
wifi_sta_config_t, 259
peer_addr
LE_CM_CONNECTION_COMPLETE_IND_T, 162
LE_CM_MSG_DIRECT_ADV_REPORT_IND_T, 168
LE_GAP_ADVERTISING_PARAM_T, 183
peer_addr_type
LE_CM_CONNECTION_COMPLETE_IND_T, 163
LE_CM_MSG_DIRECT_ADV_REPORT_IND_T, 168
LE_GAP_ADVERTISING_PARAM_T, 183
peer_id_addr
LE_SMP_MSG_PAIRING_COMPLETE_IND_T, 221
permit
LE_GATT_ATTR_T, 186
pmkid
_wpa_ie_data, 152
wifi_wpa_ie_data_t, 261
property
LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND_T, 191
proto
_wpa_ie_data, 152
asso_data, 154

wifi_wpa_ie_data_t, 261
prvData
wifi_cmd_t, 246
wifi_evt_t, 252
psk
asso_data, 154
auto_conn_info_t, 156
mw_wifi_auto_connect_ap_info_t, 228
psk_set
asso_data, 154
ptScanReport
S_WIFI_MLME_SCAN_CFG, 232
pwr_level
LE_CM_MSG_READ_ADV_TX_POWER_CFM↩
_T, 173

rand
LE_CM_MSG_LTK_REQ_IND_T, 172
LE_SMP_SC_OOB_DATA_T, 225
rear
auto_connect_cfg_t, 158
MwFimAutoConnectCFG_t, 231
reason
LE_CM_MSG_DISCONNECT_COMPLETE_IN↩
D_T, 169
wifi_event_sta_disconnected_t, 250
retryCount
auto_connect_cfg_t, 158
role
LE_CM_CONNECTION_COMPLETE_IND_T, 163
rsn_ie
auto_conn_info_t, 156
mw_wifi_auto_connect_ap_info_t, 228
scan_info_t, 235
rssi
auto_conn_info_t, 157
LE_CM_MSG_ADVERTISE_REPORT_IND_↩
T, 164
LE_CM_MSG_DIRECT_ADV_REPORT_IND_T,
168
LE_CM_MSG_READ_RSSI_CFM_T, 176
mw_wifi_auto_connect_ap_info_t, 229
scan_info_t, 235
wifi_auto_connect_info_t, 245
wifi_fast_scan_threshold_t, 252
wifi_scan_info_t, 256
rx_eapol_data, 231
frame_buffer, 231
frame_length, 231
rx_phy
LE_CM_MSG_READ_PHY_CFM_T, 175

S_WIFI_MLME_SCAN_CFG, 232
ptScanReport, 232
tScanType, 232
u32ActiveScanDur, 232
u32PassiveScanDur, 232
u8Channel, 233
u8MaxScanApNum, 233
u8ResendCnt, 233
u8aBssid, 233
u8aSsid, 233
SECONDARY_SERVICE_DECL_UUID128
BLE GATT APIs, 50
SECONDARY_SERVICE_DECL_UUID16
BLE GATT APIs, 50
saArgv
T_RfCmd, 237
sc
LE_SMP_MSG_PAIRING_ACTION_IND_T, 220
LE_SMP_MSG_PAIRING_COMPLETE_IND_T,
221
LE_SMP_MSG_SLAVE_SECURITY_REQUES↩
T_IND_T, 224
scan_done
wifi_event_info_t, 248
scan_id
wifi_event_sta_scan_done_t, 251
scan_info_t, 233
ap_channel, 234
beacon_interval, 234
bssid, 234
capabilities, 234
dtim_prod, 234
free_ocpy, 235
latest_beacon_rx_time, 235
rsn_ie, 235
rssi, 235
ssid, 235
ssid_len, 235
supported_rates, 235
wpa_data, 235
wpa_ie, 236
scan_method
wifi_sta_config_t, 259
scan_report_t, 236
pScanInfo, 236
uScanApNum, 236
scan_time
wifi_scan_config_t, 254
scan_type
wifi_scan_config_t, 254
show_hidden
wifi_scan_config_t, 254
size
LE_CM_MSG_READ_RESOLVING_LIST_SIZE↩
_CFM_T, 175
LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM↩
_T, 177
slave_latency
LE_CM_MSG_SIGNAL_UPDATE_REQ_T, 180
sort_method
wifi_sta_config_t, 259
ssid
auto_conn_info_t, 157
mw_wifi_auto_connect_ap_info_t, 229
scan_info_t, 235

wifi_ap_config_t, 243
wifi_auto_connect_info_t, 245
wifi_event_sta_connected_t, 249
wifi_event_sta_disconnected_t, 250
wifi_scan_config_t, 254
wifi_scan_info_t, 256
wifi_sta_config_t, 259
ssid_hidden
wifi_ap_config_t, 243
ssid_len
auto_conn_info_t, 157
mw_wifi_auto_connect_ap_info_t, 229
scan_info_t, 235
wifi_event_sta_connected_t, 249
wifi_event_sta_disconnected_t, 250
ssid_length
wifi_ap_config_t, 243
wifi_scan_info_t, 257
wifi_sta_config_t, 260
sta_config
wifi_config_t, 247
start_hdl
LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T, 202
LE_GATT_MSG_SERVICE_INFO_IND_T, 212
startHdl
LE_GATT_SERVICE_T, 218
status
LE_CM_CONNECTION_COMPLETE_IND_T, 163
LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T, 166
LE_CM_MSG_DISCONNECT_COMPLETE_IND_T, 169
LE_CM_MSG_ENCRYPTION_CHANGE_IND_T, 170
LE_CM_MSG_ENCRYPTION_REFRESH_IND_T, 171
LE_CM_MSG_INIT_COMPLETE_CFM_T, 171
LE_CM_MSG_READ_ADV_TX_POWER_CFM_T, 173
LE_CM_MSG_READ_BD_ADDR_CFM_T, 173
LE_CM_MSG_READ_CHANNEL_MAP_CFM_T, 174
LE_CM_MSG_READ_PHY_CFM_T, 175
LE_CM_MSG_READ_RESOLVING_LIST_SIZE_CFM_T, 175
LE_CM_MSG_READ_RSSI_CFM_T, 176
LE_CM_MSG_READ_TX_POWER_CFM_T, 177
LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM_T, 178
LE_CM_MSG_SET_DATA_LENGTH_CFM_T, 178
LE_CM_MSG_SET_DISCONNECT_CFM_T, 179
LE_CM_MSG_SET_PHY_CFM_T, 179
LE_CM_REQ_STATUS_T, 181
LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM_T, 196
LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM_T, 197
LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_CFM_T, 198
LE_GATT_MSG_FIND_CHARACTERISTIC_CFM_T, 199
LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM_T, 200
LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM_T, 201
LE_GATT_MSG_NOTIFY_CFM_T, 204
LE_GATT_MSG_PREPARE_WRITE_RELIABLE_CFM_T, 207
LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_CFM_T, 208
LE_GATT_MSG_READ_CHARACTERISTIC_VALUE_CFM_T, 209
LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM_T, 210
LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL_CFM_T, 211
LE_GATT_MSG_SIGNED_WRITE_CFM_T, 213
LE_GATT_MSG_WRITE_CHAR_VAL_RELIABLE_CFM_T, 214
LE_GATT_MSG_WRITE_CHAR_VALUE_CFM_T, 215
LE_GATT_MSG_WRITE_LONG_CHAR_VALUE_CFM_T, 216
LE_GATT_MSG_WRITE_NO_RSP_CFM_T, 217
LE_SMP_MSG_ENCRYPTION_REFRESH_IND_T, 219
LE_SMP_MSG_PAIRING_COMPLETE_IND_T, 221
wifi_event_sta_scan_done_t, 251
supervision_timeout
LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T, 166
LE_GAP_CONN_PARAM_T, 184
supervison_timeout
LE_CM_CONNECTION_COMPLETE_IND_T, 163
supported_rates
auto_conn_info_t, 157
mw_wifi_auto_connect_ap_info_t, 229
scan_info_t, 235
wifi_auto_connect_info_t, 245
sv_timeout
LE_CONN_PARA_T, 182
sv_tmo
LE_CM_MSG_CONN_PARA_REQ_T, 165
svc_id
LE_GATT_SERVICE_T, 218

T_HOUR
BLE MSG APIs, 74
T_MIN
BLE MSG APIs, 74
T_RfCmd, 236
iArgc, 237
saArgv, 237

u32Type, 237
T_RfDefEvt, 237
u32Type, 237
u8Status, 238
u8aData, 237
T_RfEvt, 238
i8Rssi, 238
pParam, 238
u16RfMode, 239
u16RxCnt, 239
u16RxCrcOkCnt, 239
u32Freq, 239
u32Mode, 239
u32RfChannel, 239
u32Type, 239
u8Freq, 239
u8IpcEnable, 240
u8Len, 240
u8Phy, 240
u8Pkt, 240
u8Reserved, 240
u8Status, 240
u8Unicast, 240
T_SEC
BLE MSG APIs, 74
TASKHANDLER
BLE MSG APIs, 75
TASKPACK
BLE MSG APIs, 76
TASK
BLE MSG APIs, 75
tScanType
S_WIFI_MLME_SCAN_CFG, 232
targetIdx
auto_connect_cfg_t, 159
MwFimAutoConnectCFG_t, 231
Task
BLE MSG APIs, 75
threshold
wifi_sta_config_t, 260
timeout_multiplier
LE_CM_MSG_SIGNAL_UPDATE_REQ_T, 180
tx_phy
LE_CM_MSG_READ_PHY_CFM_T, 175
tx_power
LE_CM_MSG_READ_TX_POWER_CFM_T, 177
type
LE_BT_ADDR_T, 161
LE_GAP_ADVERTISING_PARAM_T, 183
LE_GAP_SCAN_PARAM_T, 185

u16RfMode
T_RfEvt, 239
u16RxCnt
T_RfEvt, 239
u16RxCrcOkCnt
T_RfEvt, 239
u32ActiveScanDur
S_WIFI_MLME_SCAN_CFG, 232
u32Freq
T_RfEvt, 239
u32Mode
T_RfEvt, 239
u32PassiveScanDur
S_WIFI_MLME_SCAN_CFG, 232
u32RfChannel
T_RfEvt, 239
u32Type
T_RfCmd, 237
T_RfDefEvt, 237
T_RfEvt, 239
u8Channel
S_WIFI_MLME_SCAN_CFG, 233
u8Freq
T_RfEvt, 239
u8IpcEnable
T_RfEvt, 240
u8Len
T_RfEvt, 240
u8MaxScanApNum
S_WIFI_MLME_SCAN_CFG, 233
u8Phy
T_RfEvt, 240
u8Pkt
T_RfEvt, 240
u8ResendCnt
S_WIFI_MLME_SCAN_CFG, 233
u8Reserved
T_RfEvt, 240
u8SkipDtimPeriods
mw_wifi_sta_info_t, 230
u8Status
T_RfDefEvt, 238
T_RfEvt, 240
u8Unicast
T_RfEvt, 240
u8aBssid
S_WIFI_MLME_SCAN_CFG, 233
u8aData
T_RfDefEvt, 237
u8aSsid
S_WIFI_MLME_SCAN_CFG, 233
uFCApNum
auto_connect_cfg_t, 159
uScanApNum
scan_report_t, 236
uuid
LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_↩
IND_T, 190
LE_GATT_MSG_CHARACTERISTIC_DECL_IN↩
FO_IND_T, 191
LE_GATT_MSG_INCLUDE_SERVICE_INFO_I↩
ND_T, 202
LE_GATT_MSG_SERVICE_INFO_IND_T, 212

val
LE_GATT_MSG_CHARACTERISTIC_VAL_IND↩
_T, 192

LE_GATT_MSG_INDICATE_IND_T, 203
LE_GATT_MSG_NOTIFY_IND_T, 205
LE_GATT_MSG_READ_MULTIPLE_CHAR_VA↩
L_CFM_T, 211
val_hdl
LE_GATT_MSG_CHARACTERISTIC_DECL_IN↩
FO_IND_T, 191

WIFI APIs, 91
WIFI_BEACON_INTERVAL_LENGTH, 92
WIFI_CAPABILITY_INFO_LENGTH, 92
WIFI_LENGTH_802_11, 93
WIFI_LENGTH_PASSPHRASE, 93
WIFI_MAC_ADDRESS_LENGTH, 93
WIFI_MAC_NUM_OF_CHANNELS, 93
WIFI_MAX_LENGTH_OF_SSID, 93
WIFI_MAX_SCAN_AP_NUM, 93
WIFI_MAX_SUPPORTED_RATES, 94
wifi_ap_record_t, 94
wifi_auto_connet_mode_e, 94
wifi_event_notify_cb_t, 94
wifi_event_process_handler, 95
wifi_install_default_event_handlers, 95
wifi_register_event_handler, 95
WIFI Common APIs, 97
wifi_event_cb_t, 97
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_READY_TIME, 104
wifi_auto_connect_clear_ap_info, 113
wifi_auto_connect_clear_ap_info_api, 138
wifi_auto_connect_clear_ap_info_fp_t, 104
wifi_auto_connect_get_ap_info, 114
wifi_auto_connect_get_ap_info_api, 138
wifi_auto_connect_get_ap_info_fp_t, 105
wifi_auto_connect_get_ap_num, 114
wifi_auto_connect_get_ap_num_api, 138
wifi_auto_connect_get_ap_num_fp_t, 105
wifi_auto_connect_get_mode, 115
wifi_auto_connect_get_mode_api, 138
wifi_auto_connect_get_mode_fp_t, 105
wifi_auto_connect_get_saved_ap_num, 115
wifi_auto_connect_get_saved_ap_num_api, 138
wifi_auto_connect_get_saved_ap_num_fp_t, 105
wifi_auto_connect_init, 115
wifi_auto_connect_init_api, 138
wifi_auto_connect_init_fp_t, 105
wifi_auto_connect_reset, 116
wifi_auto_connect_reset_api, 138
wifi_auto_connect_reset_fp_t, 105
wifi_auto_connect_set_ap_num, 116
wifi_auto_connect_set_ap_num_api, 138
wifi_auto_connect_set_ap_num_fp_t, 105
wifi_auto_connect_set_mode, 117
wifi_auto_connect_set_mode_api, 139
wifi_auto_connect_set_mode_fp_t, 106
wifi_auto_connect_start, 117
wifi_auto_connect_start_api, 139
wifi_auto_connect_start_fp_t, 106
wifi_auto_connect_update_ch, 118
wifi_auto_connect_update_ch_api, 139
wifi_auto_connect_update_ch_fp_t, 106
wifi_config_get_bandwidth, 118
wifi_config_get_bandwidth_api, 139
wifi_config_get_bandwidth_fp_t, 106
wifi_config_get_bssid, 119
wifi_config_get_bssid_api, 139
wifi_config_get_bssid_fp_t, 106
wifi_config_get_channel, 119
wifi_config_get_channel_api, 139
wifi_config_get_channel_fp_t, 106
wifi_config_get_dtim_interval, 120
wifi_config_get_dtim_interval_api, 139
wifi_config_get_dtim_interval_fp_t, 106
wifi_config_get_listen_interval, 120
wifi_config_get_listen_interval_api, 139
wifi_config_get_listen_interval_fp_t, 106
wifi_config_get_mac_address, 120
wifi_config_get_mac_address_api, 140
wifi_config_get_mac_address_fp_t, 107
wifi_config_get_mac_tx_data_rate, 121
wifi_config_get_mac_tx_data_rate_api, 140
wifi_config_get_mac_tx_data_rate_fp_t, 107
wifi_config_get_opmode, 121
wifi_config_get_opmode_api, 140
wifi_config_get_opmode_fp_t, 107
wifi_config_get_skip_dtim, 121
wifi_config_get_skip_dtim_api, 140
wifi_config_get_skip_dtim_fp_t, 107
wifi_config_get_ssid, 122
wifi_config_get_ssid_api, 140
wifi_config_get_ssid_fp_t, 107
wifi_config_get_sta_mac_address_from_flash, 122
wifi_config_get_sta_mac_address_from_flash_api,
140
wifi_config_get_sta_mac_address_from_flash_↩
fp_t, 107
wifi_config_set_bandwidth, 122
wifi_config_set_bandwidth_api, 140
wifi_config_set_bandwidth_fp_t, 107
wifi_config_set_bssid, 123
wifi_config_set_bssid_api, 140
wifi_config_set_bssid_fp_t, 107
wifi_config_set_channel, 123
wifi_config_set_channel_api, 141
wifi_config_set_channel_fp_t, 108
wifi_config_set_dtim_interval, 124
wifi_config_set_dtim_interval_api, 141
wifi_config_set_dtim_interval_fp_t, 108
wifi_config_set_listen_interval, 124
wifi_config_set_listen_interval_api, 141
wifi_config_set_listen_interval_fp_t, 108
wifi_config_set_mac_address, 124
wifi_config_set_mac_address_api, 141

wifi_config_set_mac_address_fp_t, 108
wifi_config_set_mac_tx_data_rate, 125
wifi_config_set_mac_tx_data_rate_api, 141
wifi_config_set_mac_tx_data_rate_fp_t, 108
wifi_config_set_opmode, 125
wifi_config_set_opmode_api, 141
wifi_config_set_opmode_fp_t, 108
wifi_config_set_skip_dtim, 125
wifi_config_set_skip_dtim_api, 141
wifi_config_set_skip_dtim_fp_t, 108
wifi_config_set_ssid, 126
wifi_config_set_ssid_api, 141
wifi_config_set_ssid_fp_t, 108
wifi_connection_connect, 126
wifi_connection_connect_api, 142
wifi_connection_connect_fp_t, 109
wifi_connection_connect_from_ac_index, 127
wifi_connection_connect_from_ac_index_api, 142
wifi_connection_connect_from_ac_index_fp_t, 109
wifi_connection_connect_from_ac_list, 127
wifi_connection_connect_from_ac_list_api, 142
wifi_connection_connect_from_ac_list_fp_t, 109
wifi_connection_disconnect_ap, 128
wifi_connection_disconnect_ap_api, 142
wifi_connection_disconnect_ap_fp_t, 109
wifi_connection_disconnect_sta, 128
wifi_connection_disconnect_sta_api, 142
wifi_connection_disconnect_sta_fp_t, 109
wifi_connection_get_rssi, 129
wifi_connection_get_rssi_api, 142
wifi_connection_get_rssi_fp_t, 109
wifi_connection_register_event_handler, 129
wifi_connection_register_event_handler_api, 142
wifi_connection_register_event_handler_fp_t, 109
wifi_connection_scan_start, 130
wifi_connection_scan_start_api, 142
wifi_connection_scan_start_fp_t, 110
wifi_connection_unregister_event_handler, 130
wifi_connection_unregister_event_handler_api, 143
wifi_connection_unregister_event_handler_fp_t, 110
wifi_convert_auth_mode, 131
wifi_convert_auth_mode_api, 143
wifi_convert_auth_mode_fp_t, 110
wifi_deinit, 131
wifi_deinit_api, 143
wifi_deinit_fp_t, 110
wifi_event_handler_t, 110
wifi_fast_connect_get_mode, 131
wifi_fast_connect_get_mode_api, 143
wifi_fast_connect_get_mode_fp_t, 111
wifi_fast_connect_set_mode, 132
wifi_fast_connect_set_mode_api, 143
wifi_fast_connect_set_mode_fp_t, 111
wifi_fast_connect_start, 132
wifi_fast_connect_start_api, 143
wifi_fast_connect_start_fp_t, 111
wifi_get_config, 133
wifi_get_config_api, 143
wifi_get_config_fp_t, 111
wifi_init, 133
wifi_init_api, 143
wifi_init_complete_cb_t, 111
wifi_init_fp_t, 111
wifi_result_t, 112
wifi_scan_get_ap_list, 134
wifi_scan_get_ap_list_api, 144
wifi_scan_get_ap_list_fp_t, 112
wifi_scan_get_ap_num, 134
wifi_scan_get_ap_num_api, 144
wifi_scan_get_ap_num_fp_t, 112
wifi_scan_get_ap_records, 135
wifi_scan_get_ap_records_api, 144
wifi_scan_get_ap_records_fp_t, 112
wifi_scan_start, 135
wifi_scan_start_api, 144
wifi_scan_start_fp_t, 112
wifi_scan_stop, 135
wifi_scan_stop_api, 144
wifi_scan_stop_fp_t, 112
wifi_set_config, 136
wifi_set_config_api, 144
wifi_set_config_fp_t, 112
wifi_sta_get_ap_info, 136
wifi_sta_get_ap_info_api, 144
wifi_sta_get_ap_info_fp_t, 113
wifi_start, 137
wifi_start_api, 144
wifi_start_fp_t, 113
wifi_stop, 137
wifi_stop_api, 144
wifi_stop_fp_t, 113
WIFI_BEACON_INTERVAL_LENGTH
WIFI APIs, 92
WIFI_CAPABILITY_INFO_LENGTH
WIFI APIs, 92
WIFI_LENGTH_802_11
WIFI APIs, 93
WIFI_LENGTH_PASSPHRASE
WIFI APIs, 93
WIFI_MAC_ADDRESS_LENGTH
WIFI APIs, 93
WIFI_MAC_NUM_OF_CHANNELS
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, 94
WIFI_READY_TIME
WIFI STA APIs, 104
wifi_active_scan_time_t, 241
max, 241
min, 241

wifi_ap_config_t, 241
auth_mode, 242
beacon_interval, 242
channel, 242
encrypt_type, 242
max_connection, 242
password, 243
password_length, 243
ssid, 243
ssid_hidden, 243
ssid_length, 243
wifi_ap_record_t
WIFI APIs, 94
wifi_auth_mode_t
Enumeration, 145
wifi_auto_connect_clear_ap_info
WIFI STA APIs, 113
wifi_auto_connect_clear_ap_info_api
WIFI STA APIs, 138
wifi_auto_connect_clear_ap_info_fp_t
WIFI STA APIs, 104
wifi_auto_connect_get_ap_info
WIFI STA APIs, 114
wifi_auto_connect_get_ap_info_api
WIFI STA APIs, 138
wifi_auto_connect_get_ap_info_fp_t
WIFI STA APIs, 105
wifi_auto_connect_get_ap_num
WIFI STA APIs, 114
wifi_auto_connect_get_ap_num_api
WIFI STA APIs, 138
wifi_auto_connect_get_ap_num_fp_t
WIFI STA APIs, 105
wifi_auto_connect_get_mode
WIFI STA APIs, 115
wifi_auto_connect_get_mode_api
WIFI STA APIs, 138
wifi_auto_connect_get_mode_fp_t
WIFI STA APIs, 105
wifi_auto_connect_get_saved_ap_num
WIFI STA APIs, 115
wifi_auto_connect_get_saved_ap_num_api
WIFI STA APIs, 138
wifi_auto_connect_get_saved_ap_num_fp_t
WIFI STA APIs, 105
wifi_auto_connect_info_t, 243
ap_channel, 244
beacon_interval, 244
bssid, 244
capabilities, 244
dtim_prod, 244
fast_connect, 245
hid_ssid, 245
rssi, 245
ssid, 245
supported_rates, 245
wifi_auto_connect_init
WIFI STA APIs, 115
wifi_auto_connect_init_api
WIFI STA APIs, 138
wifi_auto_connect_init_fp_t
WIFI STA APIs, 105
wifi_auto_connect_reset
WIFI STA APIs, 116
wifi_auto_connect_reset_api
WIFI STA APIs, 138
wifi_auto_connect_reset_fp_t
WIFI STA APIs, 105
wifi_auto_connect_set_ap_num
WIFI STA APIs, 116
wifi_auto_connect_set_ap_num_api
WIFI STA APIs, 138
wifi_auto_connect_set_ap_num_fp_t
WIFI STA APIs, 105
wifi_auto_connect_set_mode
WIFI STA APIs, 117
wifi_auto_connect_set_mode_api
WIFI STA APIs, 139
wifi_auto_connect_set_mode_fp_t
WIFI STA APIs, 106
wifi_auto_connect_start
WIFI STA APIs, 117
wifi_auto_connect_start_api
WIFI STA APIs, 139
wifi_auto_connect_start_fp_t
WIFI STA APIs, 106
wifi_auto_connect_update_ch
WIFI STA APIs, 118
wifi_auto_connect_update_ch_api
WIFI STA APIs, 139
wifi_auto_connect_update_ch_fp_t
WIFI STA APIs, 106
wifi_auto_connet_mode_e
WIFI APIs, 94
wifi_bandwidth_t
Enumeration, 146
wifi_cipher_type_t
Enumeration, 146
wifi_cmd_t, 245
arg1, 246
arg2, 246
cmd_type, 246
prvData, 246
wifi_config_get_bandwidth
WIFI STA APIs, 118
wifi_config_get_bandwidth_api
WIFI STA APIs, 139
wifi_config_get_bandwidth_fp_t
WIFI STA APIs, 106
wifi_config_get_bssid
WIFI STA APIs, 119
wifi_config_get_bssid_api
WIFI STA APIs, 139
wifi_config_get_bssid_fp_t
WIFI STA APIs, 106
wifi_config_get_channel

WIFI STA APIs, 119
wifi_config_get_channel_api
WIFI STA APIs, 139
wifi_config_get_channel_fp_t
WIFI STA APIs, 106
wifi_config_get_dtim_interval
WIFI STA APIs, 120
wifi_config_get_dtim_interval_api
WIFI STA APIs, 139
wifi_config_get_dtim_interval_fp_t
WIFI STA APIs, 106
wifi_config_get_listen_interval
WIFI STA APIs, 120
wifi_config_get_listen_interval_api
WIFI STA APIs, 139
wifi_config_get_listen_interval_fp_t
WIFI STA APIs, 106
wifi_config_get_mac_address
WIFI STA APIs, 120
wifi_config_get_mac_address_api
WIFI STA APIs, 140
wifi_config_get_mac_address_fp_t
WIFI STA APIs, 107
wifi_config_get_mac_tx_data_rate
WIFI STA APIs, 121
wifi_config_get_mac_tx_data_rate_api
WIFI STA APIs, 140
wifi_config_get_mac_tx_data_rate_fp_t
WIFI STA APIs, 107
wifi_config_get_opmode
WIFI STA APIs, 121
wifi_config_get_opmode_api
WIFI STA APIs, 140
wifi_config_get_opmode_fp_t
WIFI STA APIs, 107
wifi_config_get_skip_dtim
WIFI STA APIs, 121
wifi_config_get_skip_dtim_api
WIFI STA APIs, 140
wifi_config_get_skip_dtim_fp_t
WIFI STA APIs, 107
wifi_config_get_ssid
WIFI STA APIs, 122
wifi_config_get_ssid_api
WIFI STA APIs, 140
wifi_config_get_ssid_fp_t
WIFI STA APIs, 107
wifi_config_get_sta_mac_address_from_flash
WIFI STA APIs, 122
wifi_config_get_sta_mac_address_from_flash_api
WIFI STA APIs, 140
wifi_config_get_sta_mac_address_from_flash_fp_t
WIFI STA APIs, 107
wifi_config_set_bandwidth
WIFI STA APIs, 122
wifi_config_set_bandwidth_api
WIFI STA APIs, 140
wifi_config_set_bandwidth_fp_t
WIFI STA APIs, 107
wifi_config_set_bssid
WIFI STA APIs, 123
wifi_config_set_bssid_api
WIFI STA APIs, 140
wifi_config_set_bssid_fp_t
WIFI STA APIs, 107
wifi_config_set_channel
WIFI STA APIs, 123
wifi_config_set_channel_api
WIFI STA APIs, 141
wifi_config_set_channel_fp_t
WIFI STA APIs, 108
wifi_config_set_dtim_interval
WIFI STA APIs, 124
wifi_config_set_dtim_interval_api
WIFI STA APIs, 141
wifi_config_set_dtim_interval_fp_t
WIFI STA APIs, 108
wifi_config_set_listen_interval
WIFI STA APIs, 124
wifi_config_set_listen_interval_api
WIFI STA APIs, 141
wifi_config_set_listen_interval_fp_t
WIFI STA APIs, 108
wifi_config_set_mac_address
WIFI STA APIs, 124
wifi_config_set_mac_address_api
WIFI STA APIs, 141
wifi_config_set_mac_address_fp_t
WIFI STA APIs, 108
wifi_config_set_mac_tx_data_rate
WIFI STA APIs, 125
wifi_config_set_mac_tx_data_rate_api
WIFI STA APIs, 141
wifi_config_set_mac_tx_data_rate_fp_t
WIFI STA APIs, 108
wifi_config_set_opmode
WIFI STA APIs, 125
wifi_config_set_opmode_api
WIFI STA APIs, 141
wifi_config_set_opmode_fp_t
WIFI STA APIs, 108
wifi_config_set_skip_dtim
WIFI STA APIs, 125
wifi_config_set_skip_dtim_api
WIFI STA APIs, 141
wifi_config_set_skip_dtim_fp_t
WIFI STA APIs, 108
wifi_config_set_ssid
WIFI STA APIs, 126
wifi_config_set_ssid_api
WIFI STA APIs, 141
wifi_config_set_ssid_fp_t
WIFI STA APIs, 108
wifi_config_t, 246
ap_config, 247
sta_config, 247

wifi_connection_connect
WIFI STA APIs, 126
wifi_connection_connect_api
WIFI STA APIs, 142
wifi_connection_connect_fp_t
WIFI STA APIs, 109
wifi_connection_connect_from_ac_index
WIFI STA APIs, 127
wifi_connection_connect_from_ac_index_api
WIFI STA APIs, 142
wifi_connection_connect_from_ac_index_fp_t
WIFI STA APIs, 109
wifi_connection_connect_from_ac_list
WIFI STA APIs, 127
wifi_connection_connect_from_ac_list_api
WIFI STA APIs, 142
wifi_connection_connect_from_ac_list_fp_t
WIFI STA APIs, 109
wifi_connection_disconnect_ap
WIFI STA APIs, 128
wifi_connection_disconnect_ap_api
WIFI STA APIs, 142
wifi_connection_disconnect_ap_fp_t
WIFI STA APIs, 109
wifi_connection_disconnect_sta
WIFI STA APIs, 128
wifi_connection_disconnect_sta_api
WIFI STA APIs, 142
wifi_connection_disconnect_sta_fp_t
WIFI STA APIs, 109
wifi_connection_get_rssi
WIFI STA APIs, 129
wifi_connection_get_rssi_api
WIFI STA APIs, 142
wifi_connection_get_rssi_fp_t
WIFI STA APIs, 109
wifi_connection_register_event_handler
WIFI STA APIs, 129
wifi_connection_register_event_handler_api
WIFI STA APIs, 142
wifi_connection_register_event_handler_fp_t
WIFI STA APIs, 109
wifi_connection_scan_start
WIFI STA APIs, 130
wifi_connection_scan_start_api
WIFI STA APIs, 142
wifi_connection_scan_start_fp_t
WIFI STA APIs, 110
wifi_connection_unregister_event_handler
WIFI STA APIs, 130
wifi_connection_unregister_event_handler_api
WIFI STA APIs, 143
wifi_connection_unregister_event_handler_fp_t
WIFI STA APIs, 110
wifi_convert_auth_mode
WIFI STA APIs, 131
wifi_convert_auth_mode_api
WIFI STA APIs, 143
wifi_convert_auth_mode_fp_t
WIFI STA APIs, 110
wifi_deinit
WIFI STA APIs, 131
wifi_deinit_api
WIFI STA APIs, 143
wifi_deinit_fp_t
WIFI STA APIs, 110
wifi_event_cb_t
WIFI Common APIs, 97
wifi_event_handler_t
WIFI STA APIs, 110
wifi_event_info_t, 247
connected, 247
disconnected, 247
got_ip, 248
scan_done, 248
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, 94
wifi_event_process_handler
WIFI APIs, 95
WIFI Common APIs, 99
wifi_event_sta_connected_t, 248
authmode, 248
bssid, 248
channel, 249
ssid, 249
ssid_len, 249
wifi_event_sta_disconnected_t, 249
bssid, 249
reason, 250
ssid, 250
ssid_len, 250
wifi_event_sta_got_ip_t, 250
ip_changed, 250
wifi_event_sta_scan_done_t, 251
number, 251
scan_id, 251
status, 251
wifi_event_t
Enumeration, 146
wifi_evt_t, 251
evt_type, 252
prvData, 252
wifi_fast_connect_get_mode
WIFI STA APIs, 131
wifi_fast_connect_get_mode_api
WIFI STA APIs, 143
wifi_fast_connect_get_mode_fp_t
WIFI STA APIs, 111
wifi_fast_connect_set_mode
WIFI STA APIs, 132

wifi_fast_connect_set_mode_api
WIFI STA APIs, 143
wifi_fast_connect_set_mode_fp_t
WIFI STA APIs, 111
wifi_fast_connect_start
WIFI STA APIs, 132
wifi_fast_connect_start_api
WIFI STA APIs, 143
wifi_fast_connect_start_fp_t
WIFI STA APIs, 111
wifi_fast_scan_threshold_t, 252
authmode, 252
rssi, 252
wifi_get_config
WIFI STA APIs, 133
wifi_get_config_api
WIFI STA APIs, 143
wifi_get_config_fp_t
WIFI STA APIs, 111
wifi_init
WIFI STA APIs, 133
wifi_init_api
WIFI STA APIs, 143
wifi_init_complete_cb_t
WIFI STA APIs, 111
wifi_init_config_t, 253
event_handler, 253
magic, 253
wifi_init_fp_t
WIFI STA APIs, 111
wifi_install_default_event_handlers
WIFI APIs, 95
wifi_mac_data_rate_t
Enumeration, 147
wifi_mode_t
Enumeration, 147
wifi_reason_code_t
Enumeration, 148
wifi_register_event_handler
WIFI APIs, 95
wifi_result_t
WIFI STA APIs, 112
wifi_scan_config_t, 253
bssid, 254
channel, 254
scan_time, 254
scan_type, 254
show_hidden, 254
ssid, 254
wifi_scan_get_ap_list
WIFI STA APIs, 134
wifi_scan_get_ap_list_api
WIFI STA APIs, 144
wifi_scan_get_ap_list_fp_t
WIFI STA APIs, 112
wifi_scan_get_ap_num
WIFI STA APIs, 134
wifi_scan_get_ap_num_api
WIFI STA APIs, 144
wifi_scan_get_ap_num_fp_t
WIFI STA APIs, 112
wifi_scan_get_ap_records
WIFI STA APIs, 135
wifi_scan_get_ap_records_api
WIFI STA APIs, 144
wifi_scan_get_ap_records_fp_t
WIFI STA APIs, 112
wifi_scan_info_t, 255
auth_mode, 255
beacon_interval, 255
bssid, 255
capability_info, 256
channel, 256
dtim_period, 256
group_cipher, 256
pairwise_cipher, 256
rssi, 256
ssid, 256
ssid_length, 257
wifi_scan_list_t, 257
ap_record, 257
num, 257
wifi_scan_method_t
Enumeration, 149
wifi_scan_start
WIFI STA APIs, 135
wifi_scan_start_api
WIFI STA APIs, 144
wifi_scan_start_fp_t
WIFI STA APIs, 112
wifi_scan_stop
WIFI STA APIs, 135
wifi_scan_stop_api
WIFI STA APIs, 144
wifi_scan_stop_fp_t
WIFI STA APIs, 112
wifi_scan_time_t, 258
active, 258
passive, 258
wifi_scan_type_t
Enumeration, 149
wifi_set_config
WIFI STA APIs, 136
wifi_set_config_api
WIFI STA APIs, 144
wifi_set_config_fp_t
WIFI STA APIs, 112
wifi_sort_method_t
Enumeration, 149
wifi_sta_config_t, 258
bssid, 259
bssid_present, 259
password, 259
password_length, 259
scan_method, 259
sort_method, 259

ssid, 259
ssid_length, 260
threshold, 260
wifi_sta_get_ap_info
WIFI STA APIs, 136
wifi_sta_get_ap_info_api
WIFI STA APIs, 144
wifi_sta_get_ap_info_fp_t
WIFI STA APIs, 113
wifi_start
WIFI STA APIs, 137
wifi_start_api
WIFI STA APIs, 144
wifi_start_fp_t
WIFI STA APIs, 113
wifi_stop
WIFI STA APIs, 137
wifi_stop_api
WIFI STA APIs, 144
wifi_stop_fp_t
WIFI STA APIs, 113
wifi_wpa_ie_data_t, 260
capabilities, 260
group_cipher, 261
key_mgmt, 261
mgmt_group_cipher, 261
num_pmkid, 261
pairwise_cipher, 261
pmkid, 261
proto, 261
window
LE_GAP_SCAN_PARAM_T, 185
wpa_data
auto_conn_info_t, 157
mw_wifi_auto_connect_ap_info_t, 229
scan_info_t, 235
wpa_ie
auto_conn_info_t, 157
mw_wifi_auto_connect_ap_info_t, 229
scan_info_t, 236