

OPL1000_WIFI_BLE_API_GUIDE

v1.1.1.1

Generated by Doxygen 1.8.14

Contents

1	SDK PREVIEW	1
2	Module Index	3
2.1	Modules	3
3	Data Structure Index	5
3.1	Data Structures	5
4	Module Documentation	9
4.1	BLE ALL APIs	9
4.1.1	Detailed Description	9
4.1.2	Function Documentation	9
4.1.2.1	LeSmpGetBondIdFromAddr()	9
4.2	BLE CM APIs	10
4.2.1	Detailed Description	11
4.2.2	Typedef Documentation	11
4.2.2.1	LE_CM_MSG_ADD_TO_RESOLVING_LIST_CFM_T	11
4.2.2.2	LE_CM_MSG_ADD_TO_WHITE_LIST_CFM_T	11
4.2.2.3	LE_CM_MSG_CANCEL_CONNECTION_CFM_T	12
4.2.2.4	LE_CM_MSG_CLEAR_RESOLVING_LIST_CFM_T	12
4.2.2.5	LE_CM_MSG_CLEAR_WHITE_LIST_CFM_T	12
4.2.2.6	LE_CM_MSG_CREATE_CONNECTION_CFM_T	12
4.2.2.7	LE_CM_MSG_ENTER_ADVERTISING_CFM_T	12
4.2.2.8	LE_CM_MSG_ENTER_SCANNING_CFM_T	12
4.2.2.9	LE_CM_MSG_EXIT_ADVERTISING_CFM_T	12

4.2.2.10	LE_CM_MSG_EXIT_SCANNING_CFM_T	12
4.2.2.11	LE_CM_MSG_PHY_UPDATE_COMPLETE_IND_T	13
4.2.2.12	LE_CM_MSG_REMOVE_FROM_RESOLVING_LIST_CFM_T	13
4.2.2.13	LE_CM_MSG_REMOVE_FROM_WHITE_LIST_CFM_T	13
4.2.2.14	LE_CM_MSG_SET_ADVERTISING_DATA_CFM_T	13
4.2.2.15	LE_CM_MSG_SET_ADVERTISING_PARAMS_CFM_T	13
4.2.2.16	LE_CM_MSG_SET_CHANNEL_MAP_CFM_T	13
4.2.2.17	LE_CM_MSG_SET_DEFAULT_PHY_CFM_T	13
4.2.2.18	LE_CM_MSG_SET_RANDOM_ADDRESS_CFM_T	13
4.2.2.19	LE_CM_MSG_SET_RPA_TIMEOUT_CFM_T	14
4.2.2.20	LE_CM_MSG_SET_SCAN_PARAMS_CFM_T	14
4.2.2.21	LE_CM_MSG_SET_SCAN_RSP_DATA_CFM_T	14
4.2.3	Enumeration Type Documentation	14
4.2.3.1	anonymous enum	14
4.2.4	Function Documentation	15
4.2.4.1	LeCmInit()	15
4.3	BLE GAP APIs	17
4.3.1	Detailed Description	19
4.3.2	Macro Definition Documentation	19
4.3.2.1	GAP_ADTYPE_128BIT_COMPLETE	19
4.3.2.2	GAP_ADTYPE_128BIT_MORE	19
4.3.2.3	GAP_ADTYPE_16BIT_COMPLETE	20
4.3.2.4	GAP_ADTYPE_16BIT_MORE	20
4.3.2.5	GAP_ADTYPE_32BIT_COMPLETE	20
4.3.2.6	GAP_ADTYPE_32BIT_MORE	20
4.3.2.7	GAP_ADTYPE_3D_INFO_DATA	20
4.3.2.8	GAP_ADTYPE_ADV_INTERVAL	20
4.3.2.9	GAP_ADTYPE_APPEARANCE	20
4.3.2.10	GAP_ADTYPE_FLAGS	20
4.3.2.11	GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED	21

4.3.2.12	GAP_ADTYPE_FLAGS_GENERAL	21
4.3.2.13	GAP_ADTYPE_FLAGS_LIMITED	21
4.3.2.14	GAP_ADTYPE_LE_BD_ADDR	21
4.3.2.15	GAP_ADTYPE_LE_ROLE	21
4.3.2.16	GAP_ADTYPE_LOCAL_NAME_COMPLETE	21
4.3.2.17	GAP_ADTYPE_LOCAL_NAME_SHORT	21
4.3.2.18	GAP_ADTYPE_MANUFACTURER_SPECIFIC	21
4.3.2.19	GAP_ADTYPE_OOB_CLASS_OF_DEVICE	22
4.3.2.20	GAP_ADTYPE_OOB_SIMPLE_PAIRING_HASHC	22
4.3.2.21	GAP_ADTYPE_OOB_SIMPLE_PAIRING_RANDR	22
4.3.2.22	GAP_ADTYPE_POWER_LEVEL	22
4.3.2.23	GAP_ADTYPE_PUBLIC_TARGET_ADDR	22
4.3.2.24	GAP_ADTYPE_RANDOM_TARGET_ADDR	22
4.3.2.25	GAP_ADTYPE_SERVICE_DATA	22
4.3.2.26	GAP_ADTYPE_SERVICE_DATA_128BIT	22
4.3.2.27	GAP_ADTYPE_SERVICE_DATA_32BIT	23
4.3.2.28	GAP_ADTYPE_SERVICES_LIST_128BIT	23
4.3.2.29	GAP_ADTYPE_SERVICES_LIST_16BIT	23
4.3.2.30	GAP_ADTYPE_SIGNED_DATA	23
4.3.2.31	GAP_ADTYPE_SIMPLE_PAIRING_HASHC_256	23
4.3.2.32	GAP_ADTYPE_SIMPLE_PAIRING_RANDR_256	23
4.3.2.33	GAP_ADTYPE_SLAVE_CONN_INTERVAL_RANGE	23
4.3.2.34	GAP_ADTYPE_SM_OOB_FLAG	23
4.3.2.35	GAP_ADTYPE_SM_TK	24
4.3.2.36	GAP_PUBLIC_ADDR	24
4.3.2.37	GAP_RAND_ADDR_NRPA	24
4.3.2.38	GAP_RAND_ADDR_RPA	24
4.3.2.39	GAP_RAND_ADDR_STATIC	24
4.3.2.40	GAP_SCAN_TYPE_ACTIVE	24
4.3.2.41	GAP_SCAN_TYPE_PASSIVE	24

4.3.2.42	GAP_TX_PWR_CURR_VAL	24
4.3.2.43	GAP_TX_PWR_MAX_VAL	25
4.3.2.44	GAPBOND_IO_CAP_DISPLAY_ONLY	25
4.3.2.45	GAPBOND_IO_CAP_DISPLAY_YES_NO	25
4.3.2.46	GAPBOND_IO_CAP_KEYBOARD_DISPLAY	25
4.3.2.47	GAPBOND_IO_CAP_KEYBOARD_ONLY	25
4.3.2.48	GAPBOND_IO_CAP_NO_INPUT_NO_OUTPUT	25
4.3.2.49	GAPBOND_PAIRING_MODE_INITIATE	25
4.3.2.50	GAPBOND_PAIRING_MODE_NO_PAIRING	25
4.3.2.51	GAPBOND_PAIRING_MODE_WAIT_FOR_REQ	26
4.3.2.52	LE_GAP_ADV_MAX_SIZE	26
4.3.3	Function Documentation	26
4.3.3.1	LeGapAddToResolvingList()	26
4.3.3.2	LeGapAddToWhiteList()	26
4.3.3.3	LeGapAdvertisingEnable()	27
4.3.3.4	LeGapCentralConnectReq()	27
4.3.3.5	LeGapCentralSetDataChannel()	27
4.3.3.6	LeGapClearResolvingList()	29
4.3.3.7	LeGapClearWhiteList()	29
4.3.3.8	LeGapConnectCancelReq()	29
4.3.3.9	LeGapConnParaRequestRsp()	29
4.3.3.10	LeGapConnUpdateRequest()	30
4.3.3.11	LeGapConnUpdateResponse()	30
4.3.3.12	LeGapDisconnectReq()	31
4.3.3.13	LeGapGenRandAddr()	31
4.3.3.14	LeGapGetBtAddr()	31
4.3.3.15	LeGapReadAdvChannelTxPower()	32
4.3.3.16	LeGapReadChannelMap()	32
4.3.3.17	LeGapReadPhy()	32
4.3.3.18	LeGapReadResolvingListSize()	32

4.3.3.19	LeGapReadRssi()	32
4.3.3.20	LeGapReadTxPower()	33
4.3.3.21	LeGapReadWhiteListSize()	33
4.3.3.22	LeGapRemoveFromWhiteList()	33
4.3.3.23	LeGapScanningReq()	34
4.3.3.24	LeGapSetAdvData()	34
4.3.3.25	LeGapSetAdvParameter()	35
4.3.3.26	LeGapSetConnParameter()	35
4.3.3.27	LeGapSetDataChannelPduLen()	35
4.3.3.28	LeGapSetDefaultPhy()	36
4.3.3.29	LeGapSetPhy()	36
4.3.3.30	LeGapSetRandAddr()	36
4.3.3.31	LeGapSetRpaTimeout()	37
4.3.3.32	LeGapSetStaticAddr()	37
4.3.3.33	LeSetScanParameter()	37
4.3.3.34	LeSetScanRspData()	38
4.4	BLE GATT APIs	39
4.4.1	Detailed Description	43
4.4.2	Macro Definition Documentation	43
4.4.2.1	CHAR_AGGREGATE_DESCRIPTOR	43
4.4.2.2	CHAR_CLIENT_CONFIG_DESCRIPTOR	44
4.4.2.3	CHAR_DECL_UUID16_ATTR_VAL	44
4.4.2.4	CHAR_EXT_PROP_DESCRIPTOR	44
4.4.2.5	CHAR_PRESENT_FORMAT_DESCRIPTOR	44
4.4.2.6	CHAR_SERVER_CONFIG_DESCRIPTOR	44
4.4.2.7	CHAR_USER_DESC_DESCRIPTOR	44
4.4.2.8	CHARACTERISTIC_DECL_UUID128	45
4.4.2.9	CHARACTERISTIC_DECL_UUID16	45
4.4.2.10	CHARACTERISTIC_UUID128	45
4.4.2.11	CHARACTERISTIC_UUID16	45

4.4.2.12	GATT_CHAR_AGG_FORMAT_UUID	45
4.4.2.13	GATT_CHAR_EXT_PROPS_UUID	45
4.4.2.14	GATT_CHAR_FORMAT_UUID	46
4.4.2.15	GATT_CHAR_USER_DESC_UUID	46
4.4.2.16	GATT_CHARACTERISTIC_UUID	46
4.4.2.17	GATT_CLIENT_CHAR_CFG_UUID	46
4.4.2.18	GATT_EXT_REPORT_REF_UUID	46
4.4.2.19	GATT_INCLUDE_UUID	46
4.4.2.20	GATT_PRIMARY_SERVICE_UUID	46
4.4.2.21	GATT_REPORT_REF_UUID	46
4.4.2.22	GATT_SECONDARY_SERVICE_UUID	47
4.4.2.23	GATT_SERV_CHAR_CFG_UUID	47
4.4.2.24	GATT_VALID_RANGE_UUID	47
4.4.2.25	INCLUDE_DECL_UUID128	47
4.4.2.26	INCLUDE_DECL_UUID128_ATTR_VAL	47
4.4.2.27	INCLUDE_DECL_UUID16_ATTR_VAL	47
4.4.2.28	INCLUDE_DECL_UINT16	47
4.4.2.29	LE_ATT_UUID_SIZE	48
4.4.2.30	LE_GATT_CHAR_PROP_AUTH	48
4.4.2.31	LE_GATT_CHAR_PROP_BCAST	48
4.4.2.32	LE_GATT_CHAR_PROP_EXT_PROP	48
4.4.2.33	LE_GATT_CHAR_PROP_IND	48
4.4.2.34	LE_GATT_CHAR_PROP_NTF	48
4.4.2.35	LE_GATT_CHAR_PROP_RD	48
4.4.2.36	LE_GATT_CHAR_PROP_WR	49
4.4.2.37	LE_GATT_CHAR_PROP_WR_NO_RESP	49
4.4.2.38	LE_GATT_CLIENT_CFG_INDICATION	49
4.4.2.39	LE_GATT_CLIENT_CFG_NOTIFICATION	49
4.4.2.40	LE_GATT_EXT_PROP_RELIABLE_WR	49
4.4.2.41	LE_GATT_EXT_PROP_WR_AUX	49

4.4.2.42	LE_GATT_FLAG_PREPARE_WRITE	49
4.4.2.43	LE_GATT_FLAG_WRITE_CMD	49
4.4.2.44	LE_GATT_FLAG_WRITE_REQ	50
4.4.2.45	LE_GATT_PERM_AUTH_READABLE	50
4.4.2.46	LE_GATT_PERM_AUTH_WRITABLE	50
4.4.2.47	LE_GATT_PERM_NONE	50
4.4.2.48	LE_GATT_PERM_READ	50
4.4.2.49	LE_GATT_PERM_RELIABLE_WRITE	50
4.4.2.50	LE_GATT_PERM_WRITE_CMD	50
4.4.2.51	LE_GATT_PERM_WRITE_REQ	50
4.4.2.52	LE_GATT_PERMIT_AUTHEN_READ	51
4.4.2.53	LE_GATT_PERMIT_AUTHEN_WRITE	51
4.4.2.54	LE_GATT_PERMIT_AUTHOR_READ	51
4.4.2.55	LE_GATT_PERMIT_AUTHOR_WRITE	51
4.4.2.56	LE_GATT_PERMIT_ENCRYPT_READ	51
4.4.2.57	LE_GATT_PERMIT_ENCRYPT_WRITE	51
4.4.2.58	LE_GATT_PERMIT_READ	51
4.4.2.59	LE_GATT_PERMIT_READABLE	51
4.4.2.60	LE_GATT_PERMIT_SC_AUTHEN_READ	52
4.4.2.61	LE_GATT_PERMIT_SC_AUTHEN_WRITE	52
4.4.2.62	LE_GATT_PERMIT_WRITABLE	52
4.4.2.63	LE_GATT_PERMIT_WRITE	52
4.4.2.64	PRIMARY_SERVICE_DECL_UUID128	52
4.4.2.65	PRIMARY_SERVICE_DECL_UUID16	52
4.4.2.66	SECONDARY_SERVICE_DECL_UUID128	52
4.4.2.67	SECONDARY_SERVICE_DECL_UUID16	53
4.4.3	Enumeration Type Documentation	53
4.4.3.1	anonymous enum	53
4.4.4	Function Documentation	54
4.4.4.1	LeGattAccessReadRsp()	54

4.4.4.2	LeGattAccessWriteRsp()	54
4.4.4.3	LeGattChangeAttrVal()	55
4.4.4.4	LeGattCharValConfirmation()	55
4.4.4.5	LeGattCharValIndicate()	56
4.4.4.6	LeGattCharValNotify()	56
4.4.4.7	LeGattExchangeMtuReq()	57
4.4.4.8	LeGattExchangeMtuRsp()	57
4.4.4.9	LeGattExecuteWriteCharValReliable()	58
4.4.4.10	LeGattFindAllCharacteristic()	58
4.4.4.11	LeGattFindAllCharDescriptor()	58
4.4.4.12	LeGattFindAllPrimaryService()	59
4.4.4.13	LeGattFindCharacteristicByUuid()	59
4.4.4.14	LeGattFindIncludedService()	60
4.4.4.15	LeGattFindPrimaryServiceByUuid()	60
4.4.4.16	LeGattGetAttrHandle()	61
4.4.4.17	LeGattGetAttrVal()	61
4.4.4.18	LeGattGetAttrValLen()	61
4.4.4.19	LeGattGetAttrValMaxLen()	63
4.4.4.20	LeGattInit()	63
4.4.4.21	LeGattModifyAttrVal()	64
4.4.4.22	LeGattPrepareWriteCharValReliable()	64
4.4.4.23	LeGattReadCharValByUuid()	65
4.4.4.24	LeGattReadCharValue()	65
4.4.4.25	LeGattReadLongCharVal()	66
4.4.4.26	LeGattReadMultipleCharVal()	66
4.4.4.27	LeGattRegisterIncludeService()	66
4.4.4.28	LeGattRegisterService()	67
4.4.4.29	LeGattSignedWriteNoRsp()	67
4.4.4.30	LeGattStopCurrentProcedure()	68
4.4.4.31	LeGattWriteCharVal()	68

4.4.4.32	LeGattWriteCharValReliable()	69
4.4.4.33	LeGattWriteLongCharVal()	69
4.4.4.34	LeGattWriteNoRsp()	70
4.4.5	Variable Documentation	70
4.4.5.1	gcCharacteristicUuid	70
4.4.5.2	gcCharAggregateUuid	70
4.4.5.3	gcCharExtPropUuid	71
4.4.5.4	gcCharFormatUuid	71
4.4.5.5	gcCharUserDescUuid	71
4.4.5.6	gcClientCharConfigUuid	71
4.4.5.7	gcExtReportRefUuid	71
4.4.5.8	gcIncludeUuid	71
4.4.5.9	gcPrimaryServiceUuid	71
4.4.5.10	gcReportRefUuid	71
4.4.5.11	gcSecondaryServiceUuid	72
4.4.5.12	gcServerCharConfigUuid	72
4.4.5.13	gcValidRangeUuid	72
4.5	BLE MSG APIs	73
4.5.1	Detailed Description	74
4.5.2	Macro Definition Documentation	74
4.5.2.1	LE_ATT_MSG_BASE	74
4.5.2.2	LE_CM_MSG_BASE	74
4.5.2.3	LE_GATT_MSG_BASE	75
4.5.2.4	LE_HCI_MSG_BASE	75
4.5.2.5	LE_L2CAP_MSG_BASE	75
4.5.2.6	LE_SMP_MSG_BASE	75
4.5.2.7	LE_SYS_MSG_BASE	75
4.5.2.8	MESSAGE_ALLOCATE	75
4.5.2.9	MESSAGE_BULID	75
4.5.2.10	MESSAGE_DATA_BULID	76

4.5.2.11	MESSAGE_OFFSET	76
4.5.2.12	T_HOUR	76
4.5.2.13	T_MIN	76
4.5.2.14	T_SEC	76
4.5.3	Typedef Documentation	76
4.5.3.1	MESSAGE	76
4.5.3.2	MESSAGEID	77
4.5.3.3	MsgData	77
4.5.3.4	MsgLock	77
4.5.3.5	MSGLOCK	77
4.5.3.6	MSGSUBID	77
4.5.3.7	MSGTIMER	77
4.5.3.8	Task	77
4.5.3.9	TASK	77
4.5.3.10	TASKHANDLER	78
4.5.3.11	TASKPACK	78
4.5.4	Enumeration Type Documentation	78
4.5.4.1	anonymous enum	78
4.5.5	Function Documentation	78
4.5.5.1	LeCancelAllMessage()	78
4.5.5.2	LeCancelAllSubMessage()	79
4.5.5.3	LeCancelFirstMessage()	79
4.5.5.4	LeCancelFirstSubMessage()	80
4.5.5.5	LeGetSubMsgId()	80
4.5.5.6	LeHostCreateTask()	80
4.5.5.7	LeHostMessageLoop()	81
4.5.5.8	LeSendMessage()	81
4.5.5.9	LeSendMessageAfter()	81
4.5.5.10	LeSendMessageUnlock()	82
4.5.5.11	LeSendSubMessage()	82

4.5.5.12	LeSendSubMessageAfter()	83
4.5.5.13	LeSendSubMessageUnlock()	83
4.6	BLE SMP APIs	85
4.6.1	Detailed Description	86
4.6.2	Macro Definition Documentation	86
4.6.2.1	LE_MAX_BOND_COUNT	86
4.6.2.2	LE_SM_IO_CAP_DISP_ONLY	86
4.6.2.3	LE_SM_IO_CAP_DISP_YES_NO	86
4.6.2.4	LE_SM_IO_CAP_KEYBOARD_DISP	87
4.6.2.5	LE_SM_IO_CAP_KEYBOARD_ONLY	87
4.6.2.6	LE_SM_IO_CAP_NO_IO	87
4.6.2.7	LE_SM_PAIR_MITM_NO	87
4.6.2.8	LE_SM_PAIR_MITM_YES	87
4.6.2.9	LE_SM_PAIR_OOB_NO	87
4.6.2.10	LE_SM_PAIR_OOB_YES	87
4.6.2.11	LE_SM_PAIR_SC_NO	87
4.6.2.12	LE_SM_PAIR_SC_YES	88
4.6.3	Enumeration Type Documentation	88
4.6.3.1	anonymous enum	88
4.6.3.2	anonymous enum	88
4.6.4	Function Documentation	89
4.6.4.1	LeSmpInit()	89
4.6.4.2	LeSmpOobAuthDataRsp()	89
4.6.4.3	LeSmpOobPresent()	89
4.6.4.4	LeSmpPasskeyInput()	90
4.6.4.5	LeSmpScOobComputeConfirmVal()	90
4.6.4.6	LeSmpScOobDataRsp()	91
4.6.4.7	LeSmpSecurityReq()	91
4.6.4.8	LeSmpSecurityRsp()	91
4.6.4.9	LeSmpSetDefaultConfig()	92

4.6.4.10	LeSmpUserConfirmRsp()	92
4.7	WIFI APIs	93
4.7.1	Detailed Description	94
4.7.2	Macro Definition Documentation	94
4.7.2.1	WIFI_BEACON_INTERVAL_LENGTH	94
4.7.2.2	WIFI_CAPABILITY_INFO_LENGTH	95
4.7.2.3	WIFI_LENGTH_802_11	95
4.7.2.4	WIFI_LENGTH_PASSPHRASE	95
4.7.2.5	WIFI_MAC_ADDRESS_LENGTH	95
4.7.2.6	WIFI_MAX_LENGTH_OF_SSID	95
4.7.2.7	WIFI_MAX_SCAN_AP_NUM	95
4.7.2.8	WIFI_MAX_SUPPORTED_RATES	96
4.7.3	Typedef Documentation	96
4.7.3.1	wifi_ap_record_t	96
4.7.3.2	wifi_event_notify_cb_t	96
4.7.4	Enumeration Type Documentation	96
4.7.4.1	wifi_auto_connet_mode_e	96
4.7.5	Function Documentation	96
4.7.5.1	wifi_event_process_handler()	96
4.7.5.2	wifi_install_default_event_handlers()	97
4.7.5.3	wifi_register_event_handler()	97
4.8	WIFI Common APIs	99
4.8.1	Detailed Description	99
4.8.2	Typedef Documentation	99
4.8.2.1	wifi_event_cb_t	99
4.8.3	Function Documentation	99
4.8.3.1	wifi_event_loop_init()	99
4.8.3.2	wifi_event_loop_send()	100
4.8.3.3	wifi_event_loop_set_cb()	100
4.8.3.4	wifi_event_process_handler()	101

4.9	WIFI STA APIs	102
4.9.1	Detailed Description	106
4.9.2	Macro Definition Documentation	106
4.9.2.1	WIFI_READY_TIME	106
4.9.3	Typedef Documentation	106
4.9.3.1	wifi_auto_connect_clear_ap_info_fp_t	106
4.9.3.2	wifi_auto_connect_get_ap_info_fp_t	106
4.9.3.3	wifi_auto_connect_get_ap_num_fp_t	106
4.9.3.4	wifi_auto_connect_get_mode_fp_t	106
4.9.3.5	wifi_auto_connect_init_fp_t	106
4.9.3.6	wifi_auto_connect_reset_fp_t	107
4.9.3.7	wifi_auto_connect_set_ap_num_fp_t	107
4.9.3.8	wifi_auto_connect_set_mode_fp_t	107
4.9.3.9	wifi_auto_connect_start_fp_t	107
4.9.3.10	wifi_config_get_bandwidth_fp_t	107
4.9.3.11	wifi_config_get_bssid_fp_t	107
4.9.3.12	wifi_config_get_channel_fp_t	107
4.9.3.13	wifi_config_get_dtim_interval_fp_t	107
4.9.3.14	wifi_config_get_listen_interval_fp_t	108
4.9.3.15	wifi_config_get_mac_address_fp_t	108
4.9.3.16	wifi_config_get_opmode_fp_t	108
4.9.3.17	wifi_config_get_ssid_fp_t	108
4.9.3.18	wifi_config_set_bandwidth_fp_t	108
4.9.3.19	wifi_config_set_bssid_fp_t	108
4.9.3.20	wifi_config_set_channel_fp_t	108
4.9.3.21	wifi_config_set_dtim_interval_fp_t	108
4.9.3.22	wifi_config_set_listen_interval_fp_t	109
4.9.3.23	wifi_config_set_mac_address_fp_t	109
4.9.3.24	wifi_config_set_opmode_fp_t	109
4.9.3.25	wifi_config_set_ssid_fp_t	109

4.9.3.26	wifi_connection_connect_fp_t	109
4.9.3.27	wifi_connection_disconnect_ap_fp_t	109
4.9.3.28	wifi_connection_disconnect_sta_fp_t	109
4.9.3.29	wifi_connection_get_rssi_fp_t	110
4.9.3.30	wifi_connection_register_event_handler_fp_t	110
4.9.3.31	wifi_connection_scan_start_fp_t	110
4.9.3.32	wifi_connection_unregister_event_handler_fp_t	110
4.9.3.33	wifi_convert_auth_mode_fp_t	110
4.9.3.34	wifi_deinit_fp_t	110
4.9.3.35	wifi_event_handler_t	110
4.9.3.36	wifi_fast_connect_get_mode_fp_t	111
4.9.3.37	wifi_fast_connect_set_mode_fp_t	111
4.9.3.38	wifi_fast_connect_start_fp_t	111
4.9.3.39	wifi_get_config_fp_t	111
4.9.3.40	wifi_init_complete_cb_t	111
4.9.3.41	wifi_init_fp_t	112
4.9.3.42	wifi_result_t	112
4.9.3.43	wifi_scan_get_ap_list_fp_t	112
4.9.3.44	wifi_scan_get_ap_num_fp_t	112
4.9.3.45	wifi_scan_get_ap_records_fp_t	112
4.9.3.46	wifi_scan_start_fp_t	112
4.9.3.47	wifi_scan_stop_fp_t	113
4.9.3.48	wifi_set_config_fp_t	113
4.9.3.49	wifi_sta_get_ap_info_fp_t	113
4.9.3.50	wifi_start_fp_t	113
4.9.3.51	wifi_stop_fp_t	113
4.9.4	Function Documentation	113
4.9.4.1	wifi_auto_connect_clear_ap_info()	113
4.9.4.2	wifi_auto_connect_get_ap_info()	114
4.9.4.3	wifi_auto_connect_get_ap_num()	114

4.9.4.4	wifi_auto_connect_get_mode()	115
4.9.4.5	wifi_auto_connect_init()	115
4.9.4.6	wifi_auto_connect_reset()	116
4.9.4.7	wifi_auto_connect_set_ap_num()	116
4.9.4.8	wifi_auto_connect_set_mode()	116
4.9.4.9	wifi_auto_connect_start()	117
4.9.4.10	wifi_config_get_bandwidth()	117
4.9.4.11	wifi_config_get_bssid()	118
4.9.4.12	wifi_config_get_channel()	118
4.9.4.13	wifi_config_get_dtim_interval()	119
4.9.4.14	wifi_config_get_listen_interval()	119
4.9.4.15	wifi_config_get_mac_address()	119
4.9.4.16	wifi_config_get_opmode()	120
4.9.4.17	wifi_config_get_skip_dtim()	120
4.9.4.18	wifi_config_get_ssid()	120
4.9.4.19	wifi_config_set_bandwidth()	121
4.9.4.20	wifi_config_set_bssid()	121
4.9.4.21	wifi_config_set_channel()	121
4.9.4.22	wifi_config_set_dtim_interval()	122
4.9.4.23	wifi_config_set_listen_interval()	122
4.9.4.24	wifi_config_set_mac_address()	122
4.9.4.25	wifi_config_set_opmode()	123
4.9.4.26	wifi_config_set_skip_dtim()	123
4.9.4.27	wifi_config_set_ssid()	124
4.9.4.28	wifi_connection_connect()	124
4.9.4.29	wifi_connection_disconnect_ap()	125
4.9.4.30	wifi_connection_disconnect_sta()	125
4.9.4.31	wifi_connection_get_rssi()	125
4.9.4.32	wifi_connection_register_event_handler()	126
4.9.4.33	wifi_connection_scan_start()	126

4.9.4.34	wifi_connection_unregister_event_handler()	127
4.9.4.35	wifi_convert_auth_mode()	127
4.9.4.36	wifi_deinit()	127
4.9.4.37	wifi_fast_connect_get_mode()	127
4.9.4.38	wifi_fast_connect_set_mode()	128
4.9.4.39	wifi_fast_connect_start()	128
4.9.4.40	wifi_get_config()	129
4.9.4.41	wifi_init()	129
4.9.4.42	wifi_scan_get_ap_list()	130
4.9.4.43	wifi_scan_get_ap_num()	130
4.9.4.44	wifi_scan_get_ap_records()	131
4.9.4.45	wifi_scan_scan_stop()	131
4.9.4.46	wifi_scan_start()	131
4.9.4.47	wifi_set_config()	132
4.9.4.48	wifi_sta_get_ap_info()	132
4.9.4.49	wifi_start()	133
4.9.4.50	wifi_stop()	133
4.9.5	Variable Documentation	133
4.9.5.1	wifi_auto_connect_clear_ap_info_api	134
4.9.5.2	wifi_auto_connect_get_ap_info_api	134
4.9.5.3	wifi_auto_connect_get_ap_num_api	134
4.9.5.4	wifi_auto_connect_get_mode_api	134
4.9.5.5	wifi_auto_connect_init_api	134
4.9.5.6	wifi_auto_connect_reset_api	134
4.9.5.7	wifi_auto_connect_set_ap_num_api	134
4.9.5.8	wifi_auto_connect_set_mode_api	134
4.9.5.9	wifi_auto_connect_start_api	135
4.9.5.10	wifi_config_get_bandwidth_api	135
4.9.5.11	wifi_config_get_bssid_api	135
4.9.5.12	wifi_config_get_channel_api	135

4.9.5.13	wifi_config_get_dtim_interval_api	135
4.9.5.14	wifi_config_get_listen_interval_api	135
4.9.5.15	wifi_config_get_mac_address_api	135
4.9.5.16	wifi_config_get_opmode_api	135
4.9.5.17	wifi_config_get_ssid_api	136
4.9.5.18	wifi_config_set_bandwidth_api	136
4.9.5.19	wifi_config_set_bssid_api	136
4.9.5.20	wifi_config_set_channel_api	136
4.9.5.21	wifi_config_set_dtim_interval_api	136
4.9.5.22	wifi_config_set_listen_interval_api	136
4.9.5.23	wifi_config_set_mac_address_api	136
4.9.5.24	wifi_config_set_opmode_api	136
4.9.5.25	wifi_config_set_ssid_api	137
4.9.5.26	wifi_connection_connect_api	137
4.9.5.27	wifi_connection_disconnect_ap_api	137
4.9.5.28	wifi_connection_disconnect_sta_api	137
4.9.5.29	wifi_connection_get_rssi_api	137
4.9.5.30	wifi_connection_register_event_handler_api	137
4.9.5.31	wifi_connection_scan_start_api	137
4.9.5.32	wifi_connection_unregister_event_handler_api	137
4.9.5.33	wifi_convert_auth_mode_api	138
4.9.5.34	wifi_deinit_api	138
4.9.5.35	wifi_fast_connect_get_mode_api	138
4.9.5.36	wifi_fast_connect_set_mode_api	138
4.9.5.37	wifi_fast_connect_start_api	138
4.9.5.38	wifi_get_config_api	138
4.9.5.39	wifi_init_api	138
4.9.5.40	wifi_scan_get_ap_list_api	138
4.9.5.41	wifi_scan_get_ap_num_api	139
4.9.5.42	wifi_scan_get_ap_records_api	139

4.9.5.43	wifi_scan_start_api	139
4.9.5.44	wifi_scan_stop_api	139
4.9.5.45	wifi_set_config_api	139
4.9.5.46	wifi_sta_get_ap_info_api	139
4.9.5.47	wifi_start_api	139
4.9.5.48	wifi_stop_api	139
4.10	Enumeration	140
4.10.1	Detailed Description	140
4.10.2	Enumeration Type Documentation	140
4.10.2.1	wifi_auth_mode_t	141
4.10.2.2	wifi_bandwidth_t	142
4.10.2.3	wifi_cipher_type_t	142
4.10.2.4	wifi_event_t	142
4.10.2.5	wifi_mode_t	143
4.10.2.6	wifi_reason_code_t	143
4.10.2.7	wifi_scan_method_t	144
4.10.2.8	wifi_scan_type_t	145
4.10.2.9	wifi_sort_method_t	145
5	Data Structure Documentation	147
5.1	_wpa_ie_data Struct Reference	147
5.1.1	Field Documentation	147
5.1.1.1	capabilities	147
5.1.1.2	group_cipher	147
5.1.1.3	key_mgmt	148
5.1.1.4	mgmt_group_cipher	148
5.1.1.5	num_pmkid	148
5.1.1.6	pairwise_cipher	148
5.1.1.7	pmkid	148
5.1.1.8	proto	148
5.2	asso_data Struct Reference	148

5.2.1	Field Documentation	149
5.2.1.1	eap_workaround	149
5.2.1.2	eapol_flags	149
5.2.1.3	group_cipher	149
5.2.1.4	key_mgmt	149
5.2.1.5	leap	149
5.2.1.6	mgmt_group_cipher	150
5.2.1.7	non_leap	150
5.2.1.8	pairwise_cipher	150
5.2.1.9	passphrase	150
5.2.1.10	proto	150
5.2.1.11	psk	150
5.2.1.12	psk_set	150
5.3	auto_conn_info_t Struct Reference	150
5.3.1	Field Documentation	151
5.3.1.1	ap_channel	151
5.3.1.2	beacon_interval	151
5.3.1.3	bssid	151
5.3.1.4	capabilities	151
5.3.1.5	dtim_prod	152
5.3.1.6	fast_connect	152
5.3.1.7	free_ocpy	152
5.3.1.8	hid_ssid	152
5.3.1.9	hid_ssid_len	152
5.3.1.10	latest_beacon_rx_time	152
5.3.1.11	passphrase	152
5.3.1.12	psk	152
5.3.1.13	rsn_ie	153
5.3.1.14	rsssi	153
5.3.1.15	ssid	153

5.3.1.16	ssid_len	153
5.3.1.17	supported_rates	153
5.3.1.18	wpa_data	153
5.3.1.19	wpa_ie	153
5.4	auto_connect_cfg_t Struct Reference	153
5.4.1	Field Documentation	154
5.4.1.1	flag	154
5.4.1.2	front	154
5.4.1.3	max_save_num	154
5.4.1.4	pFCInfo	154
5.4.1.5	rear	154
5.4.1.6	retryCount	155
5.4.1.7	targetIdx	155
5.4.1.8	uFCApNum	155
5.5	event_msg_t Struct Reference	155
5.5.1	Detailed Description	155
5.5.2	Field Documentation	155
5.5.2.1	event	155
5.5.2.2	length	156
5.5.2.3	param	156
5.6	hap_control_t Struct Reference	156
5.6.1	Field Documentation	156
5.6.1.1	hap_ap_info	156
5.6.1.2	hap_bitvector	156
5.6.1.3	hap_en	156
5.6.1.4	hap_final_index	157
5.6.1.5	hap_index	157
5.6.1.6	hap_ssid	157
5.7	LE_BT_ADDR_T Struct Reference	157
5.7.1	Field Documentation	157

5.7.1.1	addr	157
5.7.1.2	type	157
5.8	LE_CM_CONNECTION_COMPLETE_IND_T Struct Reference	158
5.8.1	Field Documentation	158
5.8.1.1	conn_hdl	158
5.8.1.2	conn_interval	158
5.8.1.3	conn_latency	158
5.8.1.4	dev_id	158
5.8.1.5	peer_addr	159
5.8.1.6	peer_addr_type	159
5.8.1.7	role	159
5.8.1.8	status	159
5.8.1.9	supervision_timeout	159
5.9	LE_CM_MSG_ADVERTISE_REPORT_IND_T Struct Reference	159
5.9.1	Field Documentation	160
5.9.1.1	addr	160
5.9.1.2	addr_type	160
5.9.1.3	data	160
5.9.1.4	event_type	160
5.9.1.5	len	160
5.9.1.6	rsi	160
5.10	LE_CM_MSG_CONN_PARA_REQ_T Struct Reference	160
5.10.1	Field Documentation	161
5.10.1.1	conn_hdl	161
5.10.1.2	itv_max	161
5.10.1.3	itv_min	161
5.10.1.4	latency	161
5.10.1.5	sv_tmo	161
5.11	LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T Struct Reference	161
5.11.1	Field Documentation	162

5.11.1.1	conn_hdl	162
5.11.1.2	interval	162
5.11.1.3	latency	162
5.11.1.4	status	162
5.11.1.5	supervision_timeout	162
5.12	LE_CM_MSG_DATA_LEN_CHANGE_IND_T Struct Reference	162
5.12.1	Field Documentation	163
5.12.1.1	conn_hdl	163
5.12.1.2	max_rx_octets	163
5.12.1.3	max_rx_time	163
5.12.1.4	max_tx_octets	163
5.12.1.5	max_tx_time	163
5.13	LE_CM_MSG_DIRECT_ADV_REPORT_IND_T Struct Reference	163
5.13.1	Field Documentation	164
5.13.1.1	direct_addr	164
5.13.1.2	direct_addr_type	164
5.13.1.3	peer_addr	164
5.13.1.4	peer_addr_type	164
5.13.1.5	rsi	164
5.14	LE_CM_MSG_DISCONNECT_COMPLETE_IND_T Struct Reference	164
5.14.1	Field Documentation	165
5.14.1.1	conn_hdl	165
5.14.1.2	reason	165
5.14.1.3	status	165
5.15	LE_CM_MSG_ENCRYPTION_CHANGE_IND_T Struct Reference	165
5.15.1	Field Documentation	165
5.15.1.1	conn_hdl	166
5.15.1.2	devid	166
5.15.1.3	enabled	166
5.15.1.4	status	166

5.16 LE_CM_MSG_ENCRYPTION_REFRESH_IND_T Struct Reference	166
5.16.1 Field Documentation	166
5.16.1.1 conn_hdl	166
5.16.1.2 devid	167
5.16.1.3 enabled	167
5.16.1.4 status	167
5.17 LE_CM_MSG_INIT_COMPLETE_CFM_T Struct Reference	167
5.17.1 Field Documentation	167
5.17.1.1 status	167
5.18 LE_CM_MSG_LTK_REQ_IND_T Struct Reference	167
5.18.1 Field Documentation	168
5.18.1.1 conn_hdl	168
5.18.1.2 devid	168
5.18.1.3 ediv	168
5.18.1.4 rand	168
5.19 LE_CM_MSG_READ_ADV_TX_POWER_CFM_T Struct Reference	168
5.19.1 Field Documentation	169
5.19.1.1 pwr_level	169
5.19.1.2 status	169
5.20 LE_CM_MSG_READ_BD_ADDR_CFM_T Struct Reference	169
5.20.1 Field Documentation	169
5.20.1.1 bd_addr	169
5.20.1.2 status	169
5.21 LE_CM_MSG_READ_CHANNEL_MAP_CFM_T Struct Reference	170
5.21.1 Field Documentation	170
5.21.1.1 ch_map	170
5.21.1.2 conn_hdl	170
5.21.1.3 status	170
5.22 LE_CM_MSG_READ_PHY_CFM_T Struct Reference	170
5.22.1 Field Documentation	171

5.22.1.1	conn_hdl	171
5.22.1.2	rx_phy	171
5.22.1.3	status	171
5.22.1.4	tx_phy	171
5.23	LE_CM_MSG_READ_RESOLVING_LIST_SIZE_CFM_T Struct Reference	171
5.23.1	Field Documentation	171
5.23.1.1	size	171
5.23.1.2	status	172
5.24	LE_CM_MSG_READ_RSSI_CFM_T Struct Reference	172
5.24.1	Field Documentation	172
5.24.1.1	conn_hdl	172
5.24.1.2	rssi	172
5.24.1.3	status	172
5.25	LE_CM_MSG_READ_TX_POWER_CFM_T Struct Reference	172
5.25.1	Field Documentation	173
5.25.1.1	conn_hdl	173
5.25.1.2	status	173
5.25.1.3	tx_power	173
5.26	LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM_T Struct Reference	173
5.26.1	Field Documentation	173
5.26.1.1	size	174
5.26.1.2	status	174
5.27	LE_CM_MSG_SET_DATA_LENGTH_CFM_T Struct Reference	174
5.27.1	Field Documentation	174
5.27.1.1	conn_hdl	174
5.27.1.2	status	174
5.28	LE_CM_MSG_SET_DISCONNECT_CFM_T Struct Reference	174
5.28.1	Field Documentation	175
5.28.1.1	handle	175
5.28.1.2	status	175

5.29 LE_CM_MSG_SET_PHY_CFM_T Struct Reference	175
5.29.1 Field Documentation	175
5.29.1.1 conn_hdl	175
5.29.1.2 status	175
5.30 LE_CM_MSG_SIGNAL_UPDATE_REQ_T Struct Reference	176
5.30.1 Field Documentation	176
5.30.1.1 conn_hdl	176
5.30.1.2 identifier	176
5.30.1.3 interval_max	176
5.30.1.4 interval_min	176
5.30.1.5 slave_latency	176
5.30.1.6 timeout_multiplier	177
5.31 LE_CM_REQ_STATUS_T Struct Reference	177
5.31.1 Field Documentation	177
5.31.1.1 status	177
5.32 LE_CONN_PARA_T Struct Reference	177
5.32.1 Field Documentation	177
5.32.1.1 itv_max	177
5.32.1.2 itv_min	178
5.32.1.3 latency	178
5.32.1.4 sv_timeout	178
5.33 LE_GAP_ADVERTISING_PARAM_T Struct Reference	178
5.33.1 Field Documentation	178
5.33.1.1 channel_map	178
5.33.1.2 filter_policy	179
5.33.1.3 interval_max	179
5.33.1.4 interval_min	179
5.33.1.5 own_addr_type	179
5.33.1.6 peer_addr	179
5.33.1.7 peer_addr_type	179

5.33.1.8	type	179
5.34	LE_GAP_CONN_PARAM_T Struct Reference	179
5.34.1	Field Documentation	180
5.34.1.1	interval_max	180
5.34.1.2	interval_min	180
5.34.1.3	latency	180
5.34.1.4	supervision_timeout	180
5.35	LE_GAP_SCAN_PARAM_T Struct Reference	180
5.35.1	Field Documentation	181
5.35.1.1	filter_policy	181
5.35.1.2	interval	181
5.35.1.3	own_addr_type	181
5.35.1.4	type	181
5.35.1.5	window	181
5.36	LE_GATT_ATTR_T Struct Reference	181
5.36.1	Field Documentation	182
5.36.1.1	format	182
5.36.1.2	handle	182
5.36.1.3	len	182
5.36.1.4	maxLen	182
5.36.1.5	permit	182
5.36.1.6	pUuid	182
5.36.1.7	pVal	182
5.37	LE_GATT_MSG_ACCESS_READ_IND_T Struct Reference	183
5.37.1	Field Documentation	183
5.37.1.1	conn_hdl	183
5.37.1.2	devid	183
5.37.1.3	handle	183
5.37.1.4	offset	183
5.38	LE_GATT_MSG_ACCESS_WRITE_IND_T Struct Reference	183

5.38.1	Field Documentation	184
5.38.1.1	conn_hdl	184
5.38.1.2	devid	184
5.38.1.3	flag	184
5.38.1.4	handle	184
5.38.1.5	len	184
5.38.1.6	offset	185
5.38.1.7	pVal	185
5.39	LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND_T Struct Reference	185
5.39.1	Field Documentation	185
5.39.1.1	conn_hdl	185
5.39.1.2	devid	185
5.39.1.3	format	185
5.39.1.4	handle	186
5.39.1.5	uuid	186
5.40	LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND_T Struct Reference	186
5.40.1	Field Documentation	186
5.40.1.1	conn_hdl	186
5.40.1.2	devid	186
5.40.1.3	format	187
5.40.1.4	handle	187
5.40.1.5	property	187
5.40.1.6	uuid	187
5.40.1.7	val_hdl	187
5.41	LE_GATT_MSG_CHARACTERISTIC_VAL_IND_T Struct Reference	187
5.41.1	Field Documentation	188
5.41.1.1	att_err	188
5.41.1.2	conn_hdl	188
5.41.1.3	devid	188
5.41.1.4	handle	188

5.41.1.5	len	188
5.41.1.6	offset	188
5.41.1.7	val	188
5.42	LE_GATT_MSG_CONFIRMATION_CFM_T Struct Reference	189
5.42.1	Field Documentation	189
5.42.1.1	conn_hdl	189
5.42.1.2	devid	189
5.42.1.3	handle	189
5.43	LE_GATT_MSG_EXCHANGE_MTU_CFM_T Struct Reference	189
5.43.1	Field Documentation	190
5.43.1.1	conn_hdl	190
5.43.1.2	current_rx_mtu	190
5.43.1.3	devid	190
5.44	LE_GATT_MSG_EXCHANGE_MTU_IND_T Struct Reference	190
5.44.1	Field Documentation	190
5.44.1.1	client_rx_mtu	190
5.44.1.2	conn_hdl	191
5.44.1.3	devid	191
5.45	LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM_T Struct Reference	191
5.45.1	Field Documentation	191
5.45.1.1	att_err	191
5.45.1.2	conn_hdl	191
5.45.1.3	devid	191
5.45.1.4	err_hdl	192
5.45.1.5	status	192
5.46	LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM_T Struct Reference	192
5.46.1	Field Documentation	192
5.46.1.1	att_err	192
5.46.1.2	conn_hdl	192
5.46.1.3	devid	192

5.46.1.4	handle	193
5.46.1.5	status	193
5.47	LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_CFM_T Struct Reference	193
5.47.1	Field Documentation	193
5.47.1.1	att_err	193
5.47.1.2	conn_hdl	193
5.47.1.3	devid	193
5.47.1.4	handle	194
5.47.1.5	status	194
5.48	LE_GATT_MSG_FIND_CHARACTERISTIC_CFM_T Struct Reference	194
5.48.1	Field Documentation	194
5.48.1.1	att_err	194
5.48.1.2	conn_hdl	194
5.48.1.3	devid	194
5.48.1.4	handle	195
5.48.1.5	status	195
5.49	LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM_T Struct Reference	195
5.49.1	Field Documentation	195
5.49.1.1	att_err	195
5.49.1.2	conn_hdl	195
5.49.1.3	devid	195
5.49.1.4	handle	196
5.49.1.5	status	196
5.50	LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM_T Struct Reference	196
5.50.1	Field Documentation	196
5.50.1.1	att_err	196
5.50.1.2	conn_hdl	196
5.50.1.3	devid	196
5.50.1.4	handle	197
5.50.1.5	status	197

5.51 LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T Struct Reference	197
5.51.1 Field Documentation	197
5.51.1.1 conn_hdl	197
5.51.1.2 devid	197
5.51.1.3 end_hdl	198
5.51.1.4 format	198
5.51.1.5 handle	198
5.51.1.6 start_hdl	198
5.51.1.7 uuid	198
5.52 LE_GATT_MSG_INDICATE_IND_T Struct Reference	198
5.52.1 Field Documentation	198
5.52.1.1 conn_hdl	199
5.52.1.2 devid	199
5.52.1.3 handle	199
5.52.1.4 len	199
5.52.1.5 val	199
5.53 LE_GATT_MSG_NOTIFY_CFM_T Struct Reference	199
5.53.1 Field Documentation	199
5.53.1.1 conn_hdl	200
5.53.1.2 devid	200
5.53.1.3 handle	200
5.53.1.4 status	200
5.54 LE_GATT_MSG_NOTIFY_IND_T Struct Reference	200
5.54.1 Field Documentation	200
5.54.1.1 conn_hdl	200
5.54.1.2 devid	201
5.54.1.3 handle	201
5.54.1.4 len	201
5.54.1.5 val	201
5.55 LE_GATT_MSG_OPERATION_TIMEOUT_T Struct Reference	201

5.55.1	Field Documentation	201
5.55.1.1	att_op	201
5.55.1.2	conn_hdl	202
5.55.1.3	devid	202
5.56	LE_GATT_MSG_PREPARE_WRITE_RELIABLE_CFM_T Struct Reference	202
5.56.1	Field Documentation	202
5.56.1.1	att_err	202
5.56.1.2	conn_hdl	202
5.56.1.3	devid	202
5.56.1.4	handle	203
5.56.1.5	status	203
5.57	LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_CFM_T Struct Reference	203
5.57.1	Field Documentation	203
5.57.1.1	att_err	203
5.57.1.2	conn_hdl	203
5.57.1.3	devid	203
5.57.1.4	handle	204
5.57.1.5	status	204
5.58	LE_GATT_MSG_READ_CHARACTERISTIC_VALUE_CFM_T Struct Reference	204
5.58.1	Field Documentation	204
5.58.1.1	att_err	204
5.58.1.2	conn_hdl	204
5.58.1.3	devid	204
5.58.1.4	handle	205
5.58.1.5	status	205
5.59	LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM_T Struct Reference	205
5.59.1	Field Documentation	205
5.59.1.1	att_err	205
5.59.1.2	conn_hdl	205
5.59.1.3	devid	205

5.59.1.4	handle	206
5.59.1.5	status	206
5.60	LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL_CFM_T Struct Reference	206
5.60.1	Field Documentation	206
5.60.1.1	att_err	206
5.60.1.2	conn_hdl	206
5.60.1.3	devid	207
5.60.1.4	err_hdl	207
5.60.1.5	len	207
5.60.1.6	status	207
5.60.1.7	val	207
5.61	LE_GATT_MSG_SERVICE_INFO_IND_T Struct Reference	207
5.61.1	Field Documentation	208
5.61.1.1	conn_hdl	208
5.61.1.2	devid	208
5.61.1.3	end_hdl	208
5.61.1.4	format	208
5.61.1.5	start_hdl	208
5.61.1.6	uuid	208
5.62	LE_GATT_MSG_SIGNED_WRITE_CFM_T Struct Reference	208
5.62.1	Field Documentation	209
5.62.1.1	conn_hdl	209
5.62.1.2	devid	209
5.62.1.3	handle	209
5.62.1.4	status	209
5.63	LE_GATT_MSG_WRITE_CHAR_VAL_RELIABLE_CFM_T Struct Reference	209
5.63.1	Field Documentation	210
5.63.1.1	att_err	210
5.63.1.2	conn_hdl	210
5.63.1.3	devid	210

5.63.1.4	handle	210
5.63.1.5	status	210
5.64	LE_GATT_MSG_WRITE_CHAR_VALUE_CFM_T Struct Reference	210
5.64.1	Field Documentation	211
5.64.1.1	att_err	211
5.64.1.2	conn_hdl	211
5.64.1.3	devid	211
5.64.1.4	handle	211
5.64.1.5	status	211
5.65	LE_GATT_MSG_WRITE_LONG_CHAR_VALUE_CFM_T Struct Reference	211
5.65.1	Field Documentation	212
5.65.1.1	att_err	212
5.65.1.2	conn_hdl	212
5.65.1.3	devid	212
5.65.1.4	handle	212
5.65.1.5	status	212
5.66	LE_GATT_MSG_WRITE_NO_RSP_CFM_T Struct Reference	212
5.66.1	Field Documentation	213
5.66.1.1	conn_hdl	213
5.66.1.2	devid	213
5.66.1.3	handle	213
5.66.1.4	status	213
5.67	LE_GATT_SERVICE_T Struct Reference	213
5.67.1	Field Documentation	213
5.67.1.1	endHdl	214
5.67.1.2	pAttr	214
5.67.1.3	startHdl	214
5.67.1.4	svc_id	214
5.68	LE_SMP_MSG_ENCRYPTION_CHANGE_IND_T Struct Reference	214
5.68.1	Field Documentation	214

5.68.1.1	conn_hdl	214
5.68.1.2	enable	215
5.69	LE_SMP_MSG_ENCRYPTION_REFRESH_IND_T Struct Reference	215
5.69.1	Field Documentation	215
5.69.1.1	conn_hdl	215
5.69.1.2	status	215
5.70	LE_SMP_MSG_OOB_DATA_REQUEST_IND_T Struct Reference	215
5.70.1	Field Documentation	215
5.70.1.1	conn_hdl	216
5.71	LE_SMP_MSG_PAIRING_ACTION_IND_T Struct Reference	216
5.71.1	Field Documentation	216
5.71.1.1	action	216
5.71.1.2	conn_hdl	216
5.71.1.3	lost_bond	216
5.71.1.4	sc	216
5.72	LE_SMP_MSG_PAIRING_COMPLETE_IND_T Struct Reference	217
5.72.1	Field Documentation	217
5.72.1.1	authenticated	217
5.72.1.2	bonded	217
5.72.1.3	conn_hdl	217
5.72.1.4	peer_id_addr	217
5.72.1.5	sc	217
5.72.1.6	status	218
5.73	LE_SMP_MSG_PASSKEY_DISPLAY_IND_T Struct Reference	218
5.73.1	Field Documentation	218
5.73.1.1	conn_hdl	218
5.73.1.2	passkey	218
5.74	LE_SMP_MSG_PASSKEY_INPUT_IND_T Struct Reference	218
5.74.1	Field Documentation	218
5.74.1.1	conn_hdl	219

5.75 LE_SMP_MSG_SC_OOB_DATA_REQUEST_IND_T Struct Reference	219
5.75.1 Field Documentation	219
5.75.1.1 conn_hdl	219
5.76 LE_SMP_MSG_SLAVE_SECURITY_REQUEST_IND_T Struct Reference	219
5.76.1 Field Documentation	219
5.76.1.1 bondable	220
5.76.1.2 conn_hdl	220
5.76.1.3 keypress	220
5.76.1.4 mitm	220
5.76.1.5 sc	220
5.77 LE_SMP_MSG_USER_CONFIRM_IND_T Struct Reference	220
5.77.1 Field Documentation	220
5.77.1.1 confirm_num	221
5.77.1.2 conn_hdl	221
5.78 LE_SMP_SC_OOB_DATA_T Struct Reference	221
5.78.1 Field Documentation	221
5.78.1.1 confirm	221
5.78.1.2 rand	221
5.79 LE_SYS_MSG_BUF_OVERFLOW_T Struct Reference	221
5.79.1 Field Documentation	222
5.79.1.1 conn_hdl	222
5.80 mw_blewifi_cbs_store_t Struct Reference	222
5.80.1 Field Documentation	222
5.80.1.1 manufacture_name	222
5.81 mw_wifi_auto_connect_ap_info_t Struct Reference	222
5.81.1 Field Documentation	223
5.81.1.1 ap_channel	223
5.81.1.2 beacon_interval	223
5.81.1.3 bssid	223
5.81.1.4 capabilities	223

5.81.1.5	dtim_prod	224
5.81.1.6	fast_connect	224
5.81.1.7	free_ocpy	224
5.81.1.8	hid_ssid	224
5.81.1.9	hid_ssid_len	224
5.81.1.10	latest_beacon_rx_time	224
5.81.1.11	passphrase	224
5.81.1.12	psk	224
5.81.1.13	rsn_ie	225
5.81.1.14	rsi	225
5.81.1.15	ssid	225
5.81.1.16	ssid_len	225
5.81.1.17	supported_rates	225
5.81.1.18	wpa_data	225
5.81.1.19	wpa_ie	225
5.82	mw_wifi_sta_info_t Struct Reference	225
5.82.1	Field Documentation	226
5.82.1.1	au8Dot11MACAddress	226
5.82.1.2	u8SkipDtimPeriods	226
5.83	MwFimAutoConnectCFG_t Struct Reference	226
5.83.1	Field Documentation	226
5.83.1.1	flag	226
5.83.1.2	front	227
5.83.1.3	max_save_num	227
5.83.1.4	rear	227
5.83.1.5	targetIdx	227
5.84	rx_eapol_data Struct Reference	227
5.84.1	Field Documentation	227
5.84.1.1	frame_buffer	227
5.84.1.2	frame_length	228

5.85 S_WIFI_MLME_SCAN_CFG Struct Reference	228
5.85.1 Detailed Description	228
5.85.2 Field Documentation	228
5.85.2.1 ptScanReport	228
5.85.2.2 tScanType	228
5.85.2.3 u32ActiveScanDur	228
5.85.2.4 u32PassiveScanDur	229
5.85.2.5 u8ABssid	229
5.85.2.6 u8ASsid	229
5.85.2.7 u8Channel	229
5.85.2.8 u8MaxScanApNum	229
5.85.2.9 u8ResendCnt	229
5.86 scan_info_t Struct Reference	229
5.86.1 Field Documentation	230
5.86.1.1 ap_channel	230
5.86.1.2 beacon_interval	230
5.86.1.3 bssid	230
5.86.1.4 capabilities	230
5.86.1.5 dtim_prod	231
5.86.1.6 free_ocpy	231
5.86.1.7 latest_beacon_rx_time	231
5.86.1.8 rsn_ie	231
5.86.1.9 rssi	231
5.86.1.10 ssid	231
5.86.1.11 ssid_len	231
5.86.1.12 supported_rates	231
5.86.1.13 wpa_data	232
5.86.1.14 wpa_ie	232
5.87 scan_report_t Struct Reference	232
5.87.1 Field Documentation	232

5.87.1.1	pScanInfo	232
5.87.1.2	uScanApNum	232
5.88	T_RfCmd Struct Reference	232
5.88.1	Field Documentation	233
5.88.1.1	iArgc	233
5.88.1.2	saArgv	233
5.88.1.3	u32Type	233
5.89	T_RfEvt Struct Reference	233
5.89.1	Field Documentation	233
5.89.1.1	pParam	234
5.89.1.2	u16RfMode	234
5.89.1.3	u16RxCnt	234
5.89.1.4	u16RxCrcOkCnt	234
5.89.1.5	u32Freq	234
5.89.1.6	u32Mode	234
5.89.1.7	u32RfChannel	234
5.89.1.8	u32Type	234
5.89.1.9	u8Freq	235
5.89.1.10	u8IpcEnable	235
5.89.1.11	u8Len	235
5.89.1.12	u8Pkt	235
5.89.1.13	u8Reserved	235
5.89.1.14	u8Status	235
5.89.1.15	u8Unicast	235
5.90	wifi_active_scan_time_t Struct Reference	235
5.90.1	Detailed Description	236
5.90.2	Field Documentation	236
5.90.2.1	max	236
5.90.2.2	min	236
5.91	wifi_ap_config_t Struct Reference	236

5.91.1 Detailed Description	237
5.91.2 Field Documentation	237
5.91.2.1 auth_mode	237
5.91.2.2 beacon_interval	237
5.91.2.3 channel	237
5.91.2.4 encrypt_type	237
5.91.2.5 max_connection	237
5.91.2.6 password	237
5.91.2.7 password_length	238
5.91.2.8 ssid	238
5.91.2.9 ssid_hidden	238
5.91.2.10 ssid_length	238
5.92 wifi_auto_connect_info_t Struct Reference	238
5.92.1 Detailed Description	239
5.92.2 Field Documentation	239
5.92.2.1 ap_channel	239
5.92.2.2 beacon_interval	239
5.92.2.3 bssid	239
5.92.2.4 capabilities	239
5.92.2.5 dtim_prod	239
5.92.2.6 fast_connect	239
5.92.2.7 hid_ssid	240
5.92.2.8 latest_beacon_rx_time	240
5.92.2.9 passphrase	240
5.92.2.10 psk	240
5.92.2.11 rsn_ie	240
5.92.2.12 rssi	240
5.92.2.13 ssid	240
5.92.2.14 supported_rates	240
5.92.2.15 wpa_data	241

5.92.2.16 wpa_ie	241
5.93 wifi_cmd_t Struct Reference	241
5.93.1 Field Documentation	241
5.93.1.1 arg1	241
5.93.1.2 cmd_type	241
5.93.1.3 prvData	241
5.93.1.4 reserved	242
5.94 wifi_config_t Union Reference	242
5.94.1 Detailed Description	242
5.94.2 Field Documentation	242
5.94.2.1 ap_config	242
5.94.2.2 sta_config	242
5.95 wifi_event_info_t Union Reference	242
5.95.1 Detailed Description	243
5.95.2 Field Documentation	243
5.95.2.1 connected	243
5.95.2.2 disconnected	243
5.95.2.3 got_ip	243
5.95.2.4 scan_done	243
5.96 wifi_event_sta_connected_t Struct Reference	243
5.96.1 Detailed Description	244
5.96.2 Field Documentation	244
5.96.2.1 authmode	244
5.96.2.2 bssid	244
5.96.2.3 channel	244
5.96.2.4 ssid	244
5.96.2.5 ssid_len	244
5.97 wifi_event_sta_disconnected_t Struct Reference	245
5.97.1 Detailed Description	245
5.97.2 Field Documentation	245

5.97.2.1	bssid	245
5.97.2.2	reason	245
5.97.2.3	ssid	245
5.97.2.4	ssid_len	245
5.98	wifi_event_sta_got_ip_t Struct Reference	246
5.98.1	Field Documentation	246
5.98.1.1	ip_changed	246
5.99	wifi_event_sta_scan_done_t Struct Reference	246
5.99.1	Detailed Description	246
5.99.2	Field Documentation	246
5.99.2.1	number	246
5.99.2.2	scan_id	247
5.99.2.3	status	247
5.100	wifi_evt_t Struct Reference	247
5.100.1	Field Documentation	247
5.100.1.1	evt_type	247
5.100.1.2	prvData	247
5.101	wifi_fast_scan_threshold_t Struct Reference	247
5.101.1	Detailed Description	248
5.101.2	Field Documentation	248
5.101.2.1	authmode	248
5.101.2.2	rsi	248
5.102	wifi_init_config_t Struct Reference	248
5.102.1	Detailed Description	248
5.102.2	Field Documentation	248
5.102.2.1	event_handler	249
5.102.2.2	magic	249
5.103	wifi_scan_config_t Struct Reference	249
5.103.1	Detailed Description	249
5.103.2	Field Documentation	249

5.103.2.1 bssid	249
5.103.2.2 channel	249
5.103.2.3 scan_time	250
5.103.2.4 scan_type	250
5.103.2.5 show_hidden	250
5.103.2.6 ssid	250
5.104wifi_scan_info_t Struct Reference	250
5.104.1 Detailed Description	250
5.104.2 Field Documentation	251
5.104.2.1 auth_mode	251
5.104.2.2 beacon_interval	251
5.104.2.3 bssid	251
5.104.2.4 capability_info	251
5.104.2.5 channel	251
5.104.2.6 dtim_period	251
5.104.2.7 group_cipher	251
5.104.2.8 pairwise_cipher	252
5.104.2.9 rssi	252
5.104.2.10ssid	252
5.104.2.11ssid_length	252
5.105wifi_scan_list_t Struct Reference	252
5.105.1 Detailed Description	252
5.105.2 Field Documentation	252
5.105.2.1 ap_record	253
5.105.2.2 num	253
5.106wifi_scan_time_t Union Reference	253
5.106.1 Detailed Description	253
5.106.2 Field Documentation	253
5.106.2.1 active	253
5.106.2.2 passive	253

5.107wifi_sta_config_t Struct Reference	254
5.107.1 Detailed Description	254
5.107.2 Field Documentation	254
5.107.2.1 bssid	254
5.107.2.2 bssid_present	254
5.107.2.3 password	254
5.107.2.4 password_length	255
5.107.2.5 scan_method	255
5.107.2.6 sort_method	255
5.107.2.7 ssid	255
5.107.2.8 ssid_length	255
5.107.2.9 threshold	255
5.108wifi_wpa_ie_data_t Struct Reference	255
5.108.1 Detailed Description	256
5.108.2 Field Documentation	256
5.108.2.1 capabilities	256
5.108.2.2 group_cipher	256
5.108.2.3 key_mgmt	256
5.108.2.4 mgmt_group_cipher	256
5.108.2.5 num_pmkid	257
5.108.2.6 pairwise_cipher	257
5.108.2.7 pmkid	257
5.108.2.8 proto	257
Index	259

Chapter 1

SDK PREVIEW

- BLE APIs :
 - GAP APIs : ble GAP APIs
 - GATT APIs : ble GATT APIs
 - CM APIs : ble CM APIs
 - MSG APIs : ble MSG APIs
 - SMP APIs : ble SMP APIs
- WiFi APIs :
 - Station APIs : station APIs
 - Common APIs : common APIs
 - Enumerations : enumerations

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

BLE ALL APIs	9
BLE CM APIs	10
BLE GAP APIs	17
BLE GATT APIs	39
BLE MSG APIs	73
BLE SMP APIs	85
WIFI APIs	93
WIFI Common APIs	99
WIFI STA APIs	102
Enumeration	140

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

_wpa_ie_data	147
asso_data	148
auto_conn_info_t	150
auto_connect_cfg_t	153
event_msg_t	
Send information to event by event_msg_t	155
hap_control_t	156
LE_BT_ADDR_T	157
LE_CM_CONNECTION_COMPLETE_IND_T	158
LE_CM_MSG_ADVERTISE_REPORT_IND_T	159
LE_CM_MSG_CONN_PARA_REQ_T	160
LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T	161
LE_CM_MSG_DATA_LEN_CHANGE_IND_T	162
LE_CM_MSG_DIRECT_ADV_REPORT_IND_T	163
LE_CM_MSG_DISCONNECT_COMPLETE_IND_T	164
LE_CM_MSG_ENCRYPTION_CHANGE_IND_T	165
LE_CM_MSG_ENCRYPTION_REFRESH_IND_T	166
LE_CM_MSG_INIT_COMPLETE_CFM_T	167
LE_CM_MSG_LTK_REQ_IND_T	167
LE_CM_MSG_READ_ADV_TX_POWER_CFM_T	168
LE_CM_MSG_READ_BD_ADDR_CFM_T	169
LE_CM_MSG_READ_CHANNEL_MAP_CFM_T	170
LE_CM_MSG_READ_PHY_CFM_T	170
LE_CM_MSG_READ_RESOLVING_LIST_SIZE_CFM_T	171
LE_CM_MSG_READ_RSSI_CFM_T	172
LE_CM_MSG_READ_TX_POWER_CFM_T	172
LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM_T	173
LE_CM_MSG_SET_DATA_LENGTH_CFM_T	174
LE_CM_MSG_SET_DISCONNECT_CFM_T	174
LE_CM_MSG_SET_PHY_CFM_T	175
LE_CM_MSG_SIGNAL_UPDATE_REQ_T	176
LE_CM_REQ_STATUS_T	177
LE_CONN_PARA_T	177
LE_GAP_ADVERTISING_PARAM_T	178
LE_GAP_CONN_PARAM_T	179

LE_GAP_SCAN_PARAM_T	180
LE_GATT_ATTR_T	181
LE_GATT_MSG_ACCESS_READ_IND_T	183
LE_GATT_MSG_ACCESS_WRITE_IND_T	183
LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND_T	185
LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND_T	186
LE_GATT_MSG_CHARACTERISTIC_VAL_IND_T	187
LE_GATT_MSG_CONFIRMATION_CFM_T	189
LE_GATT_MSG_EXCHANGE_MTU_CFM_T	189
LE_GATT_MSG_EXCHANGE_MTU_IND_T	190
LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM_T	191
LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM_T	192
LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_CFM_T	193
LE_GATT_MSG_FIND_CHARACTERISTIC_CFM_T	194
LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM_T	195
LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM_T	196
LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T	197
LE_GATT_MSG_INDICATE_IND_T	198
LE_GATT_MSG_NOTIFY_CFM_T	199
LE_GATT_MSG_NOTIFY_IND_T	200
LE_GATT_MSG_OPERATION_TIMEOUT_T	201
LE_GATT_MSG_PREPARE_WRITE_RELIABLE_CFM_T	202
LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_CFM_T	203
LE_GATT_MSG_READ_CHARACTERISTIC_VALUE_CFM_T	204
LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM_T	205
LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL_CFM_T	206
LE_GATT_MSG_SERVICE_INFO_IND_T	207
LE_GATT_MSG_SIGNED_WRITE_CFM_T	208
LE_GATT_MSG_WRITE_CHAR_VAL_RELIABLE_CFM_T	209
LE_GATT_MSG_WRITE_CHAR_VALUE_CFM_T	210
LE_GATT_MSG_WRITE_LONG_CHAR_VALUE_CFM_T	211
LE_GATT_MSG_WRITE_NO_RSP_CFM_T	212
LE_GATT_SERVICE_T	213
LE_SMP_MSG_ENCRYPTION_CHANGE_IND_T	214
LE_SMP_MSG_ENCRYPTION_REFRESH_IND_T	215
LE_SMP_MSG_OOB_DATA_REQUEST_IND_T	215
LE_SMP_MSG_PAIRING_ACTION_IND_T	216
LE_SMP_MSG_PAIRING_COMPLETE_IND_T	217
LE_SMP_MSG_PASSKEY_DISPLAY_IND_T	218
LE_SMP_MSG_PASSKEY_INPUT_IND_T	218
LE_SMP_MSG_SC_OOB_DATA_REQUEST_IND_T	219
LE_SMP_MSG_SLAVE_SECURITY_REQUEST_IND_T	219
LE_SMP_MSG_USER_CONFIRM_IND_T	220
LE_SMP_SC_OOB_DATA_T	221
LE_SYS_MSG_BUF_OVERFLOW_T	221
mw_blewifi_cbs_store_t	222
mw_wifi_auto_connect_ap_info_t	222
mw_wifi_sta_info_t	225
MwFimAutoConnectCFG_t	226
rx_eapol_data	227
S_WIFI_MLME_SCAN_CFG	228
scan_info_t	229
scan_report_t	232
T_RfCmd	232
T_RfEvt	233
wifi_active_scan_time_t	
Range of active scan times per channel	235

wifi_ap_config_t	
This structure is the Wi-Fi configuration for initialization for Soft-AP mode	236
wifi_auto_connect_info_t	
This structure is the Wi-Fi auto connect for save in the flash (FIM)	238
wifi_cmd_t	241
wifi_config_t	
Wi-Fi configuration for initialization	242
wifi_event_info_t	
Wifi_event_info_t	242
wifi_event_sta_connected_t	
Wifi_event_sta_connected_t	243
wifi_event_sta_disconnected_t	
Wifi_event_sta_disconnected_t	245
wifi_event_sta_got_ip_t	246
wifi_event_sta_scan_done_t	
Wifi_event_sta_scan_done_t	246
wifi_evt_t	247
wifi_fast_scan_threshold_t	
Structure describing parameters for a Wi-Fi fast scan	247
wifi_init_config_t	
WiFi stack configuration parameters	248
wifi_scan_config_t	
Parameters for an SSID scan	249
wifi_scan_info_t	
This structure defines the inforamtion of scanned APs	250
wifi_scan_list_t	
This structure defines the list of scanned APs with their corresponding information	252
wifi_scan_time_t	
Aggregate of active & passive scan time per channel	253
wifi_sta_config_t	
This structure is the Wi-Fi configuration for initialization for STA mode	254
wifi_wpa_ie_data_t	
This structure is the Wi-Fi auto connect with wpa information for save in the flash (FIM)	255

Chapter 4

Module Documentation

4.1 BLE ALL APIs

BLE ALL APIs.

Modules

- [BLE CM APIs](#)
- [BLE GAP APIs](#)
- [BLE GATT APIs](#)
- [BLE MSG APIs](#)
- [BLE SMP APIs](#)

Functions

- `UINT8 LeSmpGetBondIdFromAddr (LE_BT_ADDR_T *peer_addr)`

4.1.1 Detailed Description

BLE ALL APIs.

4.1.2 Function Documentation

4.1.2.1 `LeSmpGetBondIdFromAddr()`

```
UINT8 LeSmpGetBondIdFromAddr (
    LE\_BT\_ADDR\_T * peer_addr )
```

4.2 BLE CM APIs

Data Structures

- struct [LE_CM_CONNECTION_COMPLETE_IND_T](#)
- struct [LE_CM_MSG_ADVERTISE_REPORT_IND_T](#)
- struct [LE_CM_MSG_CONN_PARA_REQ_T](#)
- struct [LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T](#)
- struct [LE_CM_MSG_DATA_LEN_CHANGE_IND_T](#)
- struct [LE_CM_MSG_DIRECT_ADV_REPORT_IND_T](#)
- struct [LE_CM_MSG_DISCONNECT_COMPLETE_IND_T](#)
- struct [LE_CM_MSG_ENCRYPTION_CHANGE_IND_T](#)
- struct [LE_CM_MSG_ENCRYPTION_REFRESH_IND_T](#)
- struct [LE_CM_MSG_INIT_COMPLETE_CFM_T](#)
- struct [LE_CM_MSG_LTK_REQ_IND_T](#)
- struct [LE_CM_MSG_READ_ADV_TX_POWER_CFM_T](#)
- struct [LE_CM_MSG_READ_BD_ADDR_CFM_T](#)
- struct [LE_CM_MSG_READ_CHANNEL_MAP_CFM_T](#)
- struct [LE_CM_MSG_READ_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_CFM,
LE_CM_MSG_CREATE_CONNECTION_CFM,
LE_CM_MSG_CANCEL_CONNECTION_CFM, LE_CM_MSG_READ_TX_POWER_CFM, LE_CM_MSG_READ_BD_ADDR_CFM,
LE_CM_MSG_READ_RSSI_CFM,
LE_CM_MSG_SET_RANDOM_ADDRESS_CFM, LE_CM_MSG_READ_ADV_TX_POWER_CFM, LE_CM_MSG_READ_WHITE_LIST_CFM,
LE_CM_MSG_CLEAR_WHITE_LIST_CFM,
LE_CM_MSG_ADD_TO_WHITE_LIST_CFM, LE_CM_MSG_REMOVE_FROM_WHITE_LIST_CFM,
LE_CM_MSG_SET_CHANNEL_MAP_CFM, LE_CM_MSG_READ_CHANNEL_MAP_CFM,
LE_CM_MSG_SET_DATA_LENGTH_CFM, LE_CM_MSG_DATA_LEN_CHANGE_IND, LE_CM_MSG_ADD_TO_RESOLVING_LIST_CFM,
LE_CM_MSG_REMOVE_FROM_RESOLVING_LIST_CFM,
LE_CM_MSG_CLEAR_RESOLVING_LIST_CFM, LE_CM_MSG_READ_RESOLVING_LIST_SIZE_CFM,
LE_CM_MSG_SET_RPA_TIMEOUT_CFM, LE_CM_MSG_SIGNAL_UPDATE_REQ,
LE_CM_MSG_CONN_UPDATE_COMPLETE_IND, LE_CM_MSG_CONN_PARAM_REQ, LE_CM_MSG_ENCRYPTION_CHANNEL_UPDATE_CFM,
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.

4.2.1 Detailed Description

4.2.2 Typedef Documentation

4.2.2.1 LE_CM_MSG_ADD_TO_RESOLVING_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_ADD_TO_RESOLVING_LIST_CFM_T
```

4.2.2.2 LE_CM_MSG_ADD_TO_WHITE_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_ADD_TO_WHITE_LIST_CFM_T
```

4.2.2.3 LE_CM_MSG_CANCEL_CONNECTION_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_CANCEL_CONNECTION_CFM_T
```

4.2.2.4 LE_CM_MSG_CLEAR_RESOLVING_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_CLEAR_RESOLVING_LIST_CFM_T
```

4.2.2.5 LE_CM_MSG_CLEAR_WHITE_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_CLEAR_WHITE_LIST_CFM_T
```

4.2.2.6 LE_CM_MSG_CREATE_CONNECTION_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_CREATE_CONNECTION_CFM_T
```

4.2.2.7 LE_CM_MSG_ENTER_ADVERTISING_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_ENTER_ADVERTISING_CFM_T
```

4.2.2.8 LE_CM_MSG_ENTER_SCANNING_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_ENTER_SCANNING_CFM_T
```

4.2.2.9 LE_CM_MSG_EXIT_ADVERTISING_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_EXIT_ADVERTISING_CFM_T
```

4.2.2.10 LE_CM_MSG_EXIT_SCANNING_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_EXIT_SCANNING_CFM_T
```

4.2.2.11 LE_CM_MSG_PHY_UPDATE_COMPLETE_IND_T

```
typedef LE_CM_MSG_READ_PHY_CFM_T LE_CM_MSG_PHY_UPDATE_COMPLETE_IND_T
```

4.2.2.12 LE_CM_MSG_REMOVE_FROM_RESOLVING_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_REMOVE_FROM_RESOLVING_LIST_CFM_T
```

4.2.2.13 LE_CM_MSG_REMOVE_FROM_WHITE_LIST_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_REMOVE_FROM_WHITE_LIST_CFM_T
```

4.2.2.14 LE_CM_MSG_SET_ADVERTISING_DATA_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_ADVERTISING_DATA_CFM_T
```

4.2.2.15 LE_CM_MSG_SET_ADVERTISING_PARAMS_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_ADVERTISING_PARAMS_CFM_T
```

4.2.2.16 LE_CM_MSG_SET_CHANNEL_MAP_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_CHANNEL_MAP_CFM_T
```

4.2.2.17 LE_CM_MSG_SET_DEFAULT_PHY_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_DEFAULT_PHY_CFM_T
```

4.2.2.18 LE_CM_MSG_SET_RANDOM_ADDRESS_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_RANDOM_ADDRESS_CFM_T
```

4.2.2.19 LE_CM_MSG_SET_RPA_TIMEOUT_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_RPA_TIMEOUT_CFM_T
```

4.2.2.20 LE_CM_MSG_SET_SCAN_PARAMS_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_SCAN_PARAMS_CFM_T
```

4.2.2.21 LE_CM_MSG_SET_SCAN_RSP_DATA_CFM_T

```
typedef LE_CM_REQ_STATUS_T LE_CM_MSG_SET_SCAN_RSP_DATA_CFM_T
```

4.2.3 Enumeration Type Documentation

4.2.3.1 anonymous enum

```
anonymous enum
```

BLE connection management message id.

Enumerator

LE_CM_MSG_INIT_COMPLETE_CFM	initialize complete
LE_CM_MSG_SET_DISCONNECT_CFM	set disconnect confirm
LE_CM_MSG_DISCONNECT_COMPLETE_IND	disconnect complete indication
LE_CM_MSG_SET_ADVERTISING_DATA_CFM	set advertising data confirm
LE_CM_MSG_SET_SCAN_RSP_DATA_CFM	set scan response data confirm
LE_CM_MSG_SET_ADVERTISING_PARAMS_CFM	set advertising parameters confirm
LE_CM_MSG_ENTER_ADVERTISING_CFM	enter advertising confirm
LE_CM_MSG_EXIT_ADVERTISING_CFM	exit advertising confirm
LE_CM_MSG_SET_SCAN_PARAMS_CFM	set scan parameters confirm
LE_CM_MSG_ENTER_SCANNING_CFM	enter scanning confirm
LE_CM_MSG_EXIT_SCANNING_CFM	exit scanning confirm
LE_CM_MSG_CREATE_CONNECTION_CFM	create connection confirm
LE_CM_MSG_CANCEL_CONNECTION_CFM	cancel connection confirm
LE_CM_MSG_READ_TX_POWER_CFM	read tx power confirm
LE_CM_MSG_READ_BD_ADDR_CFM	read device address confirm
LE_CM_MSG_READ_RSSI_CFM	read RSSI confirm
LE_CM_MSG_SET_RANDOM_ADDRESS_CFM	set random address confirm
LE_CM_MSG_READ_ADV_TX_POWER_CFM	read advertising tx power confirm
LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM	read whitelist size confirm

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

4.2.4 Function Documentation

4.2.4.1 LeCmInit()

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

BLE Connection Management Module Init.

Parameters

<i>the</i>	reference of BLE task.
------------	------------------------

Returns

None.

4.3 BLE GAP APIs

Data Structures

- struct [LE_GAP_ADVERTISING_PARAM_T](#)
- struct [LE_GAP_CONN_PARAM_T](#)
- struct [LE_GAP_SCAN_PARAM_T](#)

Macros

- #define [GAP_ADTYPE_128BIT_COMPLETE](#) 0x07
- #define [GAP_ADTYPE_128BIT_MORE](#) 0x06
- #define [GAP_ADTYPE_16BIT_COMPLETE](#) 0x03
- #define [GAP_ADTYPE_16BIT_MORE](#) 0x02
- #define [GAP_ADTYPE_32BIT_COMPLETE](#) 0x05
- #define [GAP_ADTYPE_32BIT_MORE](#) 0x04
- #define [GAP_ADTYPE_3D_INFO_DATA](#) 0x3D
- #define [GAP_ADTYPE_ADV_INTERVAL](#) 0x1A
- #define [GAP_ADTYPE_APPEARANCE](#) 0x19
- #define [GAP_ADTYPE_FLAGS](#) 0x01
- #define [GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED](#) 0x04
- #define [GAP_ADTYPE_FLAGS_GENERAL](#) 0x02
- #define [GAP_ADTYPE_FLAGS_LIMITED](#) 0x01
- #define [GAP_ADTYPE_LE_BD_ADDR](#) 0x1B
- #define [GAP_ADTYPE_LE_ROLE](#) 0x1C
- #define [GAP_ADTYPE_LOCAL_NAME_COMPLETE](#) 0x09
- #define [GAP_ADTYPE_LOCAL_NAME_SHORT](#) 0x08
- #define [GAP_ADTYPE_MANUFACTURER_SPECIFIC](#) 0xFF
- #define [GAP_ADTYPE_OOB_CLASS_OF_DEVICE](#) 0x0D
- #define [GAP_ADTYPE_OOB_SIMPLE_PAIRING_HASHC](#) 0x0E
- #define [GAP_ADTYPE_OOB_SIMPLE_PAIRING_RANDOM](#) 0x0F
- #define [GAP_ADTYPE_POWER_LEVEL](#) 0x0A
- #define [GAP_ADTYPE_PUBLIC_TARGET_ADDR](#) 0x17
- #define [GAP_ADTYPE_RANDOM_TARGET_ADDR](#) 0x18
- #define [GAP_ADTYPE_SERVICE_DATA](#) 0x16
- #define [GAP_ADTYPE_SERVICE_DATA_128BIT](#) 0x21
- #define [GAP_ADTYPE_SERVICE_DATA_32BIT](#) 0x20
- #define [GAP_ADTYPE_SERVICES_LIST_128BIT](#) 0x15
- #define [GAP_ADTYPE_SERVICES_LIST_16BIT](#) 0x14
- #define [GAP_ADTYPE_SIGNED_DATA](#) 0x13
- #define [GAP_ADTYPE_SIMPLE_PAIRING_HASHC_256](#) 0x1D
- #define [GAP_ADTYPE_SIMPLE_PAIRING_RANDOM_256](#) 0x1E
- #define [GAP_ADTYPE_SLAVE_CONN_INTERVAL_RANGE](#) 0x12
- #define [GAP_ADTYPE_SM_OOB_FLAG](#) 0x11
- #define [GAP_ADTYPE_SM_TK](#) 0x10
- #define [GAP_PUBLIC_ADDR](#) 0
- #define [GAP_RAND_ADDR_NRPA](#) 2
- #define [GAP_RAND_ADDR_RPA](#) 3
- #define [GAP_RAND_ADDR_STATIC](#) 1
- #define [GAP_SCAN_TYPE_ACTIVE](#) 1
- #define [GAP_SCAN_TYPE_PASSIVE](#) 0
- #define [GAP_TX_PWR_CURR_VAL](#) 0
- #define [GAP_TX_PWR_MAX_VAL](#) 1

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

Functions

- `LE_ERR_STATE LeGapAddToResolvingList (LE_BT_ADDR_T *bt_addr, UINT8 *irk)`
Add device to resolving-list.
- `LE_ERR_STATE LeGapAddToWhiteList (LE_BT_ADDR_T *bt_addr)`
Add device to whitelist.
- `LE_ERR_STATE LeGapAdvertisingEnable (BOOL start)`
Enable or disable advertising function.
- `LE_ERR_STATE LeGapCentralConnectReq (LE_BT_ADDR_T *taddr, UINT8 own_addr_type)`
Central connect request.
- `LE_ERR_STATE LeGapCentralSetDataChannel (UINT8 *ch)`
Central set data channel.
- `LE_ERR_STATE LeGapClearResolvingList (void)`
Clear the resolving-list in the controller.
- `LE_ERR_STATE LeGapClearWhiteList (void)`
Clear whitelist in the controller.
- `LE_ERR_STATE LeGapConnectCancelReq (void)`
Cancel connect request.
- `void LeGapConnParaRequestRsp (UINT16 conn_hdl, BOOL accept)`
Connection parameters request response.
- `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.

4.3.1 Detailed Description

4.3.2 Macro Definition Documentation

4.3.2.1 GAP_ADTYPE_128BIT_COMPLETE

```
#define GAP_ADTYPE_128BIT_COMPLETE 0x07
```

4.3.2.2 GAP_ADTYPE_128BIT_MORE

```
#define GAP_ADTYPE_128BIT_MORE 0x06
```

4.3.2.3 GAP_ADTYPE_16BIT_COMPLETE

```
#define GAP_ADTYPE_16BIT_COMPLETE 0x03
```

4.3.2.4 GAP_ADTYPE_16BIT_MORE

```
#define GAP_ADTYPE_16BIT_MORE 0x02
```

4.3.2.5 GAP_ADTYPE_32BIT_COMPLETE

```
#define GAP_ADTYPE_32BIT_COMPLETE 0x05
```

4.3.2.6 GAP_ADTYPE_32BIT_MORE

```
#define GAP_ADTYPE_32BIT_MORE 0x04
```

4.3.2.7 GAP_ADTYPE_3D_INFO_DATA

```
#define GAP_ADTYPE_3D_INFO_DATA 0x3D
```

4.3.2.8 GAP_ADTYPE_ADV_INTERVAL

```
#define GAP_ADTYPE_ADV_INTERVAL 0x1A
```

4.3.2.9 GAP_ADTYPE_APPEARANCE

```
#define GAP_ADTYPE_APPEARANCE 0x19
```

4.3.2.10 GAP_ADTYPE_FLAGS

```
#define GAP_ADTYPE_FLAGS 0x01
```

4.3.2.11 GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED

```
#define GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED 0x04
```

4.3.2.12 GAP_ADTYPE_FLAGS_GENERAL

```
#define GAP_ADTYPE_FLAGS_GENERAL 0x02
```

4.3.2.13 GAP_ADTYPE_FLAGS_LIMITED

```
#define GAP_ADTYPE_FLAGS_LIMITED 0x01
```

4.3.2.14 GAP_ADTYPE_LE_BD_ADDR

```
#define GAP_ADTYPE_LE_BD_ADDR 0x1B
```

4.3.2.15 GAP_ADTYPE_LE_ROLE

```
#define GAP_ADTYPE_LE_ROLE 0x1C
```

4.3.2.16 GAP_ADTYPE_LOCAL_NAME_COMPLETE

```
#define GAP_ADTYPE_LOCAL_NAME_COMPLETE 0x09
```

4.3.2.17 GAP_ADTYPE_LOCAL_NAME_SHORT

```
#define GAP_ADTYPE_LOCAL_NAME_SHORT 0x08
```

4.3.2.18 GAP_ADTYPE_MANUFACTURER_SPECIFIC

```
#define GAP_ADTYPE_MANUFACTURER_SPECIFIC 0xFF
```

4.3.2.19 GAP_ADTYPE_OOB_CLASS_OF_DEVICE

```
#define GAP_ADTYPE_OOB_CLASS_OF_DEVICE 0x0D
```

4.3.2.20 GAP_ADTYPE_OOB_SIMPLE_PAIRING_HASHC

```
#define GAP_ADTYPE_OOB_SIMPLE_PAIRING_HASHC 0x0E
```

4.3.2.21 GAP_ADTYPE_OOB_SIMPLE_PAIRING_RANDR

```
#define GAP_ADTYPE_OOB_SIMPLE_PAIRING_RANDR 0x0F
```

4.3.2.22 GAP_ADTYPE_POWER_LEVEL

```
#define GAP_ADTYPE_POWER_LEVEL 0x0A
```

4.3.2.23 GAP_ADTYPE_PUBLIC_TARGET_ADDR

```
#define GAP_ADTYPE_PUBLIC_TARGET_ADDR 0x17
```

4.3.2.24 GAP_ADTYPE_RANDOM_TARGET_ADDR

```
#define GAP_ADTYPE_RANDOM_TARGET_ADDR 0x18
```

4.3.2.25 GAP_ADTYPE_SERVICE_DATA

```
#define GAP_ADTYPE_SERVICE_DATA 0x16
```

4.3.2.26 GAP_ADTYPE_SERVICE_DATA_128BIT

```
#define GAP_ADTYPE_SERVICE_DATA_128BIT 0x21
```

4.3.2.27 GAP_ADTYPE_SERVICE_DATA_32BIT

```
#define GAP_ADTYPE_SERVICE_DATA_32BIT 0x20
```

4.3.2.28 GAP_ADTYPE_SERVICES_LIST_128BIT

```
#define GAP_ADTYPE_SERVICES_LIST_128BIT 0x15
```

4.3.2.29 GAP_ADTYPE_SERVICES_LIST_16BIT

```
#define GAP_ADTYPE_SERVICES_LIST_16BIT 0x14
```

4.3.2.30 GAP_ADTYPE_SIGNED_DATA

```
#define GAP_ADTYPE_SIGNED_DATA 0x13
```

4.3.2.31 GAP_ADTYPE_SIMPLE_PAIRING_HASHC_256

```
#define GAP_ADTYPE_SIMPLE_PAIRING_HASHC_256 0x1D
```

4.3.2.32 GAP_ADTYPE_SIMPLE_PAIRING_RANDR_256

```
#define GAP_ADTYPE_SIMPLE_PAIRING_RANDR_256 0x1E
```

4.3.2.33 GAP_ADTYPE_SLAVE_CONN_INTERVAL_RANGE

```
#define GAP_ADTYPE_SLAVE_CONN_INTERVAL_RANGE 0x12
```

4.3.2.34 GAP_ADTYPE_SM_OOB_FLAG

```
#define GAP_ADTYPE_SM_OOB_FLAG 0x11
```

4.3.2.35 GAP_ADTYPE_SM_TK

```
#define GAP_ADTYPE_SM_TK 0x10
```

4.3.2.36 GAP_PUBLIC_ADDR

```
#define GAP_PUBLIC_ADDR 0
```

4.3.2.37 GAP_RAND_ADDR_NRPA

```
#define GAP_RAND_ADDR_NRPA 2
```

4.3.2.38 GAP_RAND_ADDR_RPA

```
#define GAP_RAND_ADDR_RPA 3
```

4.3.2.39 GAP_RAND_ADDR_STATIC

```
#define GAP_RAND_ADDR_STATIC 1
```

4.3.2.40 GAP_SCAN_TYPE_ACTIVE

```
#define GAP_SCAN_TYPE_ACTIVE 1
```

4.3.2.41 GAP_SCAN_TYPE_PASSIVE

```
#define GAP_SCAN_TYPE_PASSIVE 0
```

4.3.2.42 GAP_TX_PWR_CURR_VAL

```
#define GAP_TX_PWR_CURR_VAL 0
```

4.3.2.43 GAP_TX_PWR_MAX_VAL

```
#define GAP_TX_PWR_MAX_VAL 1
```

4.3.2.44 GAPBOND_IO_CAP_DISPLAY_ONLY

```
#define GAPBOND_IO_CAP_DISPLAY_ONLY 0x00
```

4.3.2.45 GAPBOND_IO_CAP_DISPLAY_YES_NO

```
#define GAPBOND_IO_CAP_DISPLAY_YES_NO 0x01
```

4.3.2.46 GAPBOND_IO_CAP_KEYBOARD_DISPLAY

```
#define GAPBOND_IO_CAP_KEYBOARD_DISPLAY 0x04
```

4.3.2.47 GAPBOND_IO_CAP_KEYBOARD_ONLY

```
#define GAPBOND_IO_CAP_KEYBOARD_ONLY 0x02
```

4.3.2.48 GAPBOND_IO_CAP_NO_INPUT_NO_OUTPUT

```
#define GAPBOND_IO_CAP_NO_INPUT_NO_OUTPUT 0x03
```

4.3.2.49 GAPBOND_PAIRING_MODE_INITIATE

```
#define GAPBOND_PAIRING_MODE_INITIATE 0x02
```

4.3.2.50 GAPBOND_PAIRING_MODE_NO_PAIRING

```
#define GAPBOND_PAIRING_MODE_NO_PAIRING 0x00
```

4.3.2.51 GAPBOND_PAIRING_MODE_WAIT_FOR_REQ

```
#define GAPBOND_PAIRING_MODE_WAIT_FOR_REQ 0x01
```

4.3.2.52 LE_GAP_ADV_MAX_SIZE

```
#define LE_GAP_ADV_MAX_SIZE 31
```

4.3.3 Function Documentation

4.3.3.1 LeGapAddToResolvingList()

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

Add device to resolving-list.

Parameters

<i>bt_addr</i>	BT device address.
<i>irk</i>	IRK, Identity Resolving Key

Returns

- SYS_ERR_SUCCESS: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.2 LeGapAddToWhiteList()

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

Add device to whitelist.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.3 LeGapAdvertisingEnable()

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

Enable or disable advertising function.

Parameters

<i>start</i>	TRUE is enable, FALSE is disable.
--------------	-----------------------------------

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.4 LeGapCentralConnectReq()

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

Central connect request.

Parameters

<i>taddr</i>	advertisers device address.
<i>own_addr_type</i>	owner address type.

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.5 LeGapCentralSetDataChannel()

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

Central set data channel.

Parameters

<i>ch</i>	data channel.
-----------	---------------

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.6 LeGapClearResolvingList()

```
LE_ERR_STATE LeGapClearResolvingList (  
    void )
```

Clear the resolving-list in the controller.

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.7 LeGapClearWhiteList()

```
LE_ERR_STATE LeGapClearWhiteList (  
    void )
```

Clear whitelist in the controller.

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.8 LeGapConnectCancelReq()

```
LE_ERR_STATE LeGapConnectCancelReq (  
    void )
```

Cancel connect request.

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.9 LeGapConnParaRequestRsp()

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

Connection parameters request response.

Parameters

<i>conn_hdl</i>	connection handle.
<i>accept</i>	TRUE is accept, FALSE is not.

Returns

None.

4.3.3.10 LeGapConnUpdateRequest()

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

Connection parameters update request.

Parameters

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

Returns

None.

4.3.3.11 LeGapConnUpdateResponse()

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

Connection parameters update response.

Parameters

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

Returns

None.

4.3.3.12 LeGapDisconnectReq()

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

Disconnect the physical connection.

Parameters

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

Returns

- SYS_ERR_SUCCESS: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.13 LeGapGenRandAddr()

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

Called to generation random address.

Parameters

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

Returns

- SYS_ERR_SUCCESS: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.14 LeGapGetBtAddr()

```
void LeGapGetBtAddr (
    void )
```

Get owner device address.

4.3.3.15 LeGapReadAdvChannelTxPower()

```
void LeGapReadAdvChannelTxPower (
    void )
```

Read ADV channel txpower.

4.3.3.16 LeGapReadChannelMap()

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

Read channel map.

Parameters

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

Returns

- SYS_ERR_SUCCESS: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.17 LeGapReadPhy()

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

4.3.3.18 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](#).

4.3.3.19 LeGapReadRssi()

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

Read RSSI value from controller.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.20 `LeGapReadTxPower()`

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

Read tx power value for the specified connection.

Parameters

<i>conn_hdl</i>	connection handle.
<i>type</i>	current tx power, or maximum tx power. Don't support.

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.21 `LeGapReadWhiteListSize()`

```
void LeGapReadWhiteListSize (
    void )
```

Read whitelist size in the controller.

4.3.3.22 `LeGapRemoveFromWhiteList()`

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

Remove device from whitelist.

Remove device from resolving-list.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.23 LeGapScanningReq()

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

Request scanning start.

Parameters

<i>start</i>	TRUE is start, FALSE is not.
<i>filter</i>	scan policy, refer to <code>LE_HCI_SCAN_FILT_*</code> in ble_hci_if.h

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.24 LeGapSetAdvData()

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

Called to set ADV data.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.25 LeGapSetAdvParameter()

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

Called to set ADV parameters.

Parameters

<i>params</i>	advertising params.
---------------	---------------------

Returns

- SYS_ERR_SUCCESS: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.26 LeGapSetConnParameter()

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

Called to set connection parameters.

Parameters

<i>interval_min</i>	mininum connection interval.
<i>interval_max</i>	maxinum connection interval.
<i>slave_latency</i>	slave letency.
<i>supervision_timeout</i>	supervison timeout.

Returns

- SYS_ERR_SUCCESS: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.27 LeGapSetDataChannelPduLen()

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

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

Set data channel PDU length.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.28 LeGapSetDefaultPhy()

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

4.3.3.29 LeGapSetPhy()

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

4.3.3.30 LeGapSetRandAddr()

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

Called to set random address.

Parameters

<i>addr</i>	the random address which should be set.
-------------	---

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.31 `LeGapSetRpaTimeout()`

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

Set resolvable private address timeout.

Parameters

<i>timeout</i>	RPA_Timeout, measured in seconds.
----------------	-----------------------------------

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.32 `LeGapSetStaticAddr()`

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

Called to set static address.

Parameters

<i>addr</i>	the static address which should be set.
-------------	---

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.33 `LeSetScanParameter()`

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

Called to set scan parameters.

Parameters

<i>params</i>	scan parameters.
---------------	------------------

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.3.3.34 LeSetScanRspData()

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

Called to set scan response data.

Parameters

<i>len</i>	scan response data length.
<i>data</i>	scan response data.

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4 BLE GATT APIs

Data Structures

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

Macros

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

- `#define CHARACTERISTIC_DECL_UUID128(pVal) {0, LE_GATT_UUID16, (UINT16 *)&gcCharacteristicUuid, LE_GATT_PERMIT_READ, 0, 19, (UINT8 *)&(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_UUID16(pVal) {0, LE_GATT_UUID16, (UINT16 *)&gcIncludeUuid, LE_GATT_PERMIT_READ, 0, 6, (UINT8 *)&(pVal)}`
- `#define LE_ATT_UUID_SIZE 2`
- `#define LE_GATT_CHAR_PROP_AUTH 0x40`
- `#define LE_GATT_CHAR_PROP_BCAST 0x01`
- *Characteristic Properties Bit.*
- `#define LE_GATT_CHAR_PROP_EXT_PROP 0x80`
- `#define LE_GATT_CHAR_PROP_IND 0x20`
- `#define LE_GATT_CHAR_PROP_NTF 0x10`
- `#define LE_GATT_CHAR_PROP_RD 0x02`
- `#define LE_GATT_CHAR_PROP_WR 0x08`
- `#define LE_GATT_CHAR_PROP_WR_NO_RESP 0x04`
- `#define LE_GATT_CLIENT_CFG_INDICATION 0x02`
- `#define LE_GATT_CLIENT_CFG_NOTIFICATION 0x01`
- `#define LE_GATT_EXT_PROP_RELIABLE_WR 0x0001`
- `#define LE_GATT_EXT_PROP_WR_AUX 0x0002`
- `#define LE_GATT_FLAG_PREPARE_WRITE 0x02`
- `#define LE_GATT_FLAG_WRITE_CMD 0x01`
- `#define LE_GATT_FLAG_WRITE_REQ 0x00`
- `#define LE_GATT_PERM_AUTH_READABLE (0x1<<4)`
- `#define LE_GATT_PERM_AUTH_WRITABLE (0x1<<6)`
- `#define LE_GATT_PERM_NONE (0x00)`
- `#define LE_GATT_PERM_READ (0x1<<1)`
- `#define LE_GATT_PERM_RELIABLE_WRITE (0x1<<5)`
- `#define LE_GATT_PERM_WRITE_CMD (0x1<<2)`
- `#define LE_GATT_PERM_WRITE_REQ (0x1<<3)`
- `#define LE_GATT_PERMIT_AUTHEN_READ (0x0040)`
- `#define LE_GATT_PERMIT_AUTHEN_WRITE (0x0080)`
- `#define LE_GATT_PERMIT_AUTHOR_READ (0x0004)`
- `#define LE_GATT_PERMIT_AUTHOR_WRITE (0x0008)`

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

Enumerations

- enum {
`LE_GATT_MSG_INIT_CFM = LE_GATT_MSG_BASE, LE_GATT_MSG_EXCHANGE_MTU_IND, LE_GATT_MSG_EXCHANGE_MTU_RSP, LE_GATT_MSG_ACCESS_READ_IND, LE_GATT_MSG_ACCESS_WRITE_IND, LE_GATT_MSG_SERVICE_INFO_IND, LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICES_IND, 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, UI↔NT16 uuid)
Called to register an include service.
- LE_GATT_SERVICE_T * [LeGattRegisterService](#) (LE_GATT_ATTR_T *attrTable, UINT16 numAttr)
Called to register a service.

- LE_ERR_STATE [LeGattSignedWriteNoRsp](#) (UINT16 conn_hdl, UINT16 handle, UINT16 len, UINT8 *val)
Signed write without response.
- void [LeGattStopCurrentProcedure](#) (UINT16 conn_hdl)
Stop current procedure.
- LE_ERR_STATE [LeGattWriteCharVal](#) (UINT16 conn_hdl, UINT16 handle, UINT16 len, UINT8 *val)
Write characteristic value.
- LE_ERR_STATE [LeGattWriteCharValReliable](#) (UINT16 conn_hdl, UINT16 handle, UINT16 offset, UINT16 len, UINT8 *val)
Write characteristic value reliable.
- LE_ERR_STATE [LeGattWriteLongCharVal](#) (UINT16 conn_hdl, UINT16 handle, UINT16 offset, UINT16 len, UINT8 *val)
Write long characteristic value.
- LE_ERR_STATE [LeGattWriteNoRsp](#) (UINT16 conn_hdl, UINT16 handle, UINT16 len, UINT8 *val)
Write without response.

Variables

- const UINT16 [gcCharacteristicUuid](#)
- const UINT16 [gcCharAggregateUuid](#)
- const UINT16 [gcCharExtPropUuid](#)
- const UINT16 [gcCharFormatUuid](#)
- const UINT16 [gcCharUserDescUuid](#)
- const UINT16 [gcClientCharConfigUuid](#)
- const UINT16 [gcExtReportRefUuid](#)
- const UINT16 [gcIncludeUuid](#)
- const UINT16 [gcPrimaryServiceUuid](#)
- const UINT16 [gcReportRefUuid](#)
- const UINT16 [gcSecondaryServiceUuid](#)
- const UINT16 [gcServerCharConfigUuid](#)
- const UINT16 [gcValidRangeUuid](#)

4.4.1 Detailed Description

4.4.2 Macro Definition Documentation

4.4.2.1 CHARAggregateDescriptor

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

4.4.2.2 CHAR_CLIENT_CONFIG_DESCRIPTOR

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

4.4.2.3 CHAR_DECL_UUID16_ATTR_VAL

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

4.4.2.4 CHAR_EXT_PROP_DESCRIPTOR

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

4.4.2.5 CHAR_PRESENT_FORMAT_DESCRIPTOR

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

4.4.2.6 CHAR_SERVER_CONFIG_DESCRIPTOR

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

4.4.2.7 CHAR_USER_DESC_DESCRIPTOR

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

4.4.2.8 CHARACTERISTIC_DECL_UUID128

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

4.4.2.9 CHARACTERISTIC_DECL_UUID16

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

4.4.2.10 CHARACTERISTIC_UUID128

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

4.4.2.11 CHARACTERISTIC_UUID16

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

4.4.2.12 GATT_CHAR_AGG_FORMAT_UUID

```
#define GATT_CHAR_AGG_FORMAT_UUID 0x2905
```

4.4.2.13 GATT_CHAR_EXT_PROPS_UUID

```
#define GATT_CHAR_EXT_PROPS_UUID 0x2900
```

4.4.2.14 GATT_CHAR_FORMAT_UUID

```
#define GATT_CHAR_FORMAT_UUID 0x2904
```

4.4.2.15 GATT_CHAR_USER_DESC_UUID

```
#define GATT_CHAR_USER_DESC_UUID 0x2901
```

4.4.2.16 GATT_CHARACTERISTIC_UUID

```
#define GATT_CHARACTERISTIC_UUID 0x2803
```

4.4.2.17 GATT_CLIENT_CHAR_CFG_UUID

```
#define GATT_CLIENT_CHAR_CFG_UUID 0x2902
```

4.4.2.18 GATT_EXT_REPORT_REF_UUID

```
#define GATT_EXT_REPORT_REF_UUID 0x2907
```

4.4.2.19 GATT_INCLUDE_UUID

```
#define GATT_INCLUDE_UUID 0x2802
```

4.4.2.20 GATT_PRIMARY_SERVICE_UUID

```
#define GATT_PRIMARY_SERVICE_UUID 0x2800
```

4.4.2.21 GATT_REPORT_REF_UUID

```
#define GATT_REPORT_REF_UUID 0x2908
```

4.4.2.22 GATT_SECONDARY_SERVICE_UUID

```
#define GATT_SECONDARY_SERVICE_UUID 0x2801
```

4.4.2.23 GATT_SERV_CHAR_CFG_UUID

```
#define GATT_SERV_CHAR_CFG_UUID 0x2903
```

4.4.2.24 GATT_VALID_RANGE_UUID

```
#define GATT_VALID_RANGE_UUID 0x2906
```

4.4.2.25 INCLUDE_DECL_UUID128

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

4.4.2.26 INCLUDE_DECL_UUID128_ATTR_VAL

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

4.4.2.27 INCLUDE_DECL_UUID16_ATTR_VAL

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

4.4.2.28 INCLUDE_DECL_UUIN16

```
#define INCLUDE_DECL_UUIN16(  
    pVal ) {0, LE_GATT_UUID16, (UINT16 *)&gcIncludeUuid, LE_GATT_PERMIT_READ, 0, 6,  
(UINT8 *) (pVal)}
```

4.4.2.29 LE_ATT_UUID_SIZE

```
#define LE_ATT_UUID_SIZE 2
```

4.4.2.30 LE_GATT_CHAR_PROP_AUTH

```
#define LE_GATT_CHAR_PROP_AUTH 0x40
```

4.4.2.31 LE_GATT_CHAR_PROP_BCAST

```
#define LE_GATT_CHAR_PROP_BCAST 0x01
```

Characteristic Properties Bit.

4.4.2.32 LE_GATT_CHAR_PROP_EXT_PROP

```
#define LE_GATT_CHAR_PROP_EXT_PROP 0x80
```

4.4.2.33 LE_GATT_CHAR_PROP_IND

```
#define LE_GATT_CHAR_PROP_IND 0x20
```

4.4.2.34 LE_GATT_CHAR_PROP_NTF

```
#define LE_GATT_CHAR_PROP_NTF 0x10
```

4.4.2.35 LE_GATT_CHAR_PROP_RD

```
#define LE_GATT_CHAR_PROP_RD 0x02
```

4.4.2.36 LE_GATT_CHAR_PROP_WR

```
#define LE_GATT_CHAR_PROP_WR 0x08
```

4.4.2.37 LE_GATT_CHAR_PROP_WR_NO_RESP

```
#define LE_GATT_CHAR_PROP_WR_NO_RESP 0x04
```

4.4.2.38 LE_GATT_CLIENT_CFG_INDICATION

```
#define LE_GATT_CLIENT_CFG_INDICATION 0x02
```

4.4.2.39 LE_GATT_CLIENT_CFG_NOTIFICATION

```
#define LE_GATT_CLIENT_CFG_NOTIFICATION 0x01
```

4.4.2.40 LE_GATT_EXT_PROP_RELIABLE_WR

```
#define LE_GATT_EXT_PROP_RELIABLE_WR 0x0001
```

4.4.2.41 LE_GATT_EXT_PROP_WR_AUX

```
#define LE_GATT_EXT_PROP_WR_AUX 0x0002
```

4.4.2.42 LE_GATT_FLAG_PREPARE_WRITE

```
#define LE_GATT_FLAG_PREPARE_WRITE 0x02
```

4.4.2.43 LE_GATT_FLAG_WRITE_CMD

```
#define LE_GATT_FLAG_WRITE_CMD 0x01
```

4.4.2.44 LE_GATT_FLAG_WRITE_REQ

```
#define LE_GATT_FLAG_WRITE_REQ 0x00
```

4.4.2.45 LE_GATT_PERM_AUTH_READABLE

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

4.4.2.46 LE_GATT_PERM_AUTH_WRITABLE

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

4.4.2.47 LE_GATT_PERM_NONE

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

4.4.2.48 LE_GATT_PERM_READ

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

4.4.2.49 LE_GATT_PERM_RELIABLE_WRITE

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

4.4.2.50 LE_GATT_PERM_WRITE_CMD

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

4.4.2.51 LE_GATT_PERM_WRITE_REQ

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


4.4.2.52 LE_GATT_PERMIT_AUTHEN_READ

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

4.4.2.53 LE_GATT_PERMIT_AUTHEN_WRITE

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

4.4.2.54 LE_GATT_PERMIT_AUTHOR_READ

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

4.4.2.55 LE_GATT_PERMIT_AUTHOR_WRITE

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

4.4.2.56 LE_GATT_PERMIT_ENCRYPT_READ

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

4.4.2.57 LE_GATT_PERMIT_ENCRYPT_WRITE

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

4.4.2.58 LE_GATT_PERMIT_READ

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

4.4.2.59 LE_GATT_PERMIT_READABLE

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

4.4.2.60 LE_GATT_PERMIT_SC_AUTHEN_READ

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

4.4.2.61 LE_GATT_PERMIT_SC_AUTHEN_WRITE

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

4.4.2.62 LE_GATT_PERMIT_WRITABLE

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

4.4.2.63 LE_GATT_PERMIT_WRITE

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

4.4.2.64 PRIMARY_SERVICE_DECL_UUID128

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

4.4.2.65 PRIMARY_SERVICE_DECL_UUID16

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

4.4.2.66 SECONDARY_SERVICE_DECL_UUID128

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

4.4.2.67 SECONDARY_SERVICE_DECL_UUID16

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

4.4.3 Enumeration Type Documentation

4.4.3.1 anonymous enum

anonymous enum

BLE GATT message id.

Enumerator

LE_GATT_MSG_INIT_CFM	initialize confirm message
LE_GATT_MSG_EXCHANGE_MTU_IND	exchange MTU indication
LE_GATT_MSG_EXCHANGE_MTU_CFM	exchange MTU confirm
LE_GATT_MSG_ACCESS_READ_IND	access read indication
LE_GATT_MSG_ACCESS_WRITE_IND	access write indication
LE_GATT_MSG_SERVICE_INFO_IND	service information indication
LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE↔ _CFM	find all primary service confirm
LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY↔ _UUID_CFM	find primary service by UUID confirm
LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND	include service information
LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM	find include service confirm
LE_GATT_MSG_CHARACTERISTIC_DECL_INF↔ O_IND	characteristic declaration info indication
LE_GATT_MSG_FIND_CHARACTERISTIC_CFM	find characteristic confirm
LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND	characteristic descriptor info indication
LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM	find all characteristic descriptors confirm
LE_GATT_MSG_CHARACTERISTIC_VAL_IND	characteristic value, indication message
LE_GATT_MSG_READ_CHARACTERISTIC_VAL↔ UE_CFM	read characteristic value, confirm message
LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_C↔ FM	read characteristic value by UUID confirm message
LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM	read long characteristic value confirm message
LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL ↔ 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

4.4.4 Function Documentation

4.4.4.1 LeGattAccessReadRsp()

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

Gatt access read response.

Parameters

<i>conn_hdl</i>	connection handle.
<i>handle</i>	attribute handle.
<i>att_err</i>	0 is OK, others refer to LE_ATT_ERR_* in ble_att_if.h .

Returns

- SYS_ERR_SUCCESS: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.2 LeGattAccessWriteRsp()

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

Gatt access write response.

Parameters

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

Returns

- SYS_ERR_SUCCESS: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.3 LeGattChangeAttrVal()

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

Change attribute value.

Parameters

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

Returns

- SYS_ERR_SUCCESS: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.4 LeGattCharValConfirmation()

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

Prepare write characteristic value response.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.5 LeGattCharValIndicate()

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

Gatt characteristic value indication.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.6 LeGattCharValNotify()

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

Gatt characteristic value notification.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.7 `LeGattExchangeMtuReq()`

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

Exchange MTU request.

Parameters

<i>conn_hdl</i>	connection handle.
<i>mtu</i>	MTU.

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.8 `LeGattExchangeMtuRsp()`

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

Exchange MTU response.

Parameters

<i>conn_hdl</i>	connection handle.
<i>mtu</i>	MTU.

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.9 LeGattExecuteWriteCharValReliable()

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

Execute write characteristic value request.

Parameters

<i>conn_hdl</i>	connection handle.
<i>yesno</i>	execute write or not.

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.10 LeGattFindAllCharacteristic()

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

Find all characteristic.

Parameters

<i>conn_hdl</i>	connection handle.
<i>start_hdl</i>	start handle.
<i>end_hdl</i>	end handle.

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.11 LeGattFindAllCharDescriptor()

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

Find all characteristic description.

Parameters

<i>conn_hdl</i>	connection handle.
<i>start_hdl</i>	start handle.
<i>end_hdl</i>	end handle.

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.12 `LeGattFindAllPrimaryService()`

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

Find all primary service.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.13 `LeGattFindCharacteristicByUuid()`

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

Find characteristic by UUID.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.14 LeGattFindIncludedService()

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

Find include service.

Parameters

<i>conn_hdl</i>	connection handle.
<i>start_hdl</i>	start handle.
<i>end_hdl</i>	end handle.

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.15 LeGattFindPrimaryServiceByUuid()

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

Find primary service by UUID.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.16 LeGattGetAttrHandle()

```

UINT16 LeGattGetAttrHandle (
    LE_GATT_SERVICE_T * svc,
    UINT16 attrId )

```

Get attribute handle.

Parameters

<i>svc</i>	service.
<i>attrId</i>	attribute index of service.

Returns

- SYS_ERR_SUCCESS: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.17 LeGattGetAttrVal()

```

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

```

Get attribute value.

Parameters

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

Returns

- SYS_ERR_SUCCESS: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.18 LeGattGetAttrValLen()

```

UINT16 LeGattGetAttrValLen (
    LE_GATT_SERVICE_T * svc,
    UINT16 attrId )

```

Get the length of attribute value.

Parameters

<i>svc</i>	service.
<i>attrId</i>	attribute index of service.

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.19 `LeGattGetAttrValMaxLen()`

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

Get the max length of attribute value.

Parameters

<i>svc</i>	service.
<i>attrId</i>	attribute index of service.

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.20 `LeGattInit()`

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

BLE Gatt module init.

Parameters

<i>appTask</i>	the reference of BLE task.
----------------	----------------------------

Returns

None.

4.4.4.21 LeGattModifyAttrVal()

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

Modify attribute value.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.22 LeGattPrepareWriteCharValReliable()

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

Prepare write characteristic value request.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.23 LeGattReadCharValByUuid()

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

Read a characteristic value by UUID.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.24 LeGattReadCharValue()

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

Read a characteristic value.

Parameters

<i>conn_hdl</i>	connection handle.
<i>handle</i>	characteristic value handle.

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.25 LeGattReadLongCharVal()

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

Read a long characteristic value.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.26 LeGattReadMultipleCharVal()

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

Read Multiple characteristic values.

Parameters

<i>conn_hdl</i>	connection handle.
<i>count</i>	handle count.
<i>handle</i>	handle table.

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.27 LeGattRegisterIncludeService()

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



```

UINT16 start_hdl,
UINT16 end_hdl,
UINT16 uuid )

```

Called to register an include service.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.28 LeGattRegisterService()

```

LE_GATT_SERVICE_T* LeGattRegisterService (
    LE_GATT_ATTR_T * attrTable,
    UINT16 numAttr )

```

Called to register a service.

Parameters

<i>attrTable</i>	service attribute table.
<i>numAttr</i>	the attribute number of service.

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.29 LeGattSignedWriteNoRsp()

```

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

```

Signed write without response.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.30 LeGattStopCurrentProcedure()

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

Stop current procedure.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.31 LeGattWriteCharVal()

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

Write characteristic value.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.32 `LeGattWriteCharValReliable()`

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

Write characteristic value reliable.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.33 `LeGattWriteLongCharVal()`

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

Write long characteristic value.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.4.34 LeGattWriteNoRsp()

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

Write without response.

Parameters

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

Returns

- `SYS_ERR_SUCCESS`: success.
- others: refer to error code in [ble_err.h](#).

4.4.5 Variable Documentation**4.4.5.1 gcCharacteristicUuid**

```
const UINT16 gcCharacteristicUuid
```

4.4.5.2 gcCharAggregateUuid

```
const UINT16 gcCharAggregateUuid
```

4.4.5.3 gcCharExtPropUuid

```
const UINT16 gcCharExtPropUuid
```

4.4.5.4 gcCharFormatUuid

```
const UINT16 gcCharFormatUuid
```

4.4.5.5 gcCharUserDescUuid

```
const UINT16 gcCharUserDescUuid
```

4.4.5.6 gcClientCharConfigUuid

```
const UINT16 gcClientCharConfigUuid
```

4.4.5.7 gcExtReportRefUuid

```
const UINT16 gcExtReportRefUuid
```

4.4.5.8 gcIncludeUuid

```
const UINT16 gcIncludeUuid
```

4.4.5.9 gcPrimaryServiceUuid

```
const UINT16 gcPrimaryServiceUuid
```

4.4.5.10 gcReportRefUuid

```
const UINT16 gcReportRefUuid
```

4.4.5.11 gcSecondaryServiceUuid

```
const UINT16 gcSecondaryServiceUuid
```

4.4.5.12 gcServerCharConfigUuid

```
const UINT16 gcServerCharConfigUuid
```

4.4.5.13 gcValidRangeUuid

```
const UINT16 gcValidRangeUuid
```

4.5 BLE MSG APIs

Data Structures

- struct [LE_SYS_MSG_BUF_OVERFLOW_T](#)

Macros

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

Typedefs

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

Enumerations

- `enum { LE_SYS_MSG_BUF_OVERFLOW = (LE_SYS_MSG_BASE + 1), LE_SYS_MSG_TOP }`
BLE system message id.

Functions

- UINT16 [LeCancelAllMessage](#) (TASK task, MESSAGEID id)
Cancel all message in queue.
- UINT16 [LeCancelAllSubMessage](#) (TASK task, MESSAGEID id, MSGSUBID subId)
Cancel all sub message in queue.
- BOOL [LeCancelFirstMessage](#) (TASK task, MESSAGEID id)
Cancel the first message in queue.
- BOOL [LeCancelFirstSubMessage](#) (TASK task, MESSAGEID id, MSGSUBID subId)
Cancel the first sub message in queue.
- UINT16 [LeGetSubMsgId](#) (UINT16 *s)
Get sub message id.
- BOOL [LeHostCreateTask](#) (TASK task, TASKHANDLER hdl)
Create BLE task.
- void [LeHostMessageLoop](#) (void)
message loop run.
- void [LeSendMessage](#) (TASK task, MESSAGEID msgId, MESSAGE msg)
Send message to BLE task.
- void [LeSendMessageAfter](#) (TASK task, MESSAGEID msgId, MESSAGE msg, UINT32 delay)
Delay, then send message to BLE task.
- void [LeSendMessageUnlock](#) (TASK task, MESSAGEID id, MESSAGE msg, MSGLOCK lock)
Send message until lock is 0.
- void [LeSendSubMessage](#) (TASK task, MESSAGEID msgId, MSGSUBID subId, MESSAGE msg)
Send sub message.
- void [LeSendSubMessageAfter](#) (TASK task, MESSAGEID msgId, MSGSUBID subId, MESSAGE msg, UINT32 delay)
Delay, then send sub message.
- void [LeSendSubMessageUnlock](#) (TASK task, MESSAGEID id, MSGSUBID subId, MESSAGE msg, MSGLOCK lock)
Send sub message until lock is 0.

4.5.1 Detailed Description

4.5.2 Macro Definition Documentation

4.5.2.1 LE_ATT_MSG_BASE

```
#define LE_ATT_MSG_BASE 0x1400
```

4.5.2.2 LE_CM_MSG_BASE

```
#define LE_CM_MSG_BASE 0x1100
```


4.5.2.3 LE_GATT_MSG_BASE

```
#define LE_GATT_MSG_BASE 0x1500
```

4.5.2.4 LE_HCI_MSG_BASE

```
#define LE_HCI_MSG_BASE 0x1000
```

4.5.2.5 LE_L2CAP_MSG_BASE

```
#define LE_L2CAP_MSG_BASE 0x1200
```

4.5.2.6 LE_SMP_MSG_BASE

```
#define LE_SMP_MSG_BASE 0x1300
```

4.5.2.7 LE_SYS_MSG_BASE

```
#define LE_SYS_MSG_BASE 0x8000
```

4.5.2.8 MESSAGE_ALLOCATE

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

4.5.2.9 MESSAGE_BULID

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

4.5.2.10 MESSAGE_DATA_BULID

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

4.5.2.11 MESSAGE_OFFSET

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

4.5.2.12 T_HOUR

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

4.5.2.13 T_MIN

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

4.5.2.14 T_SEC

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

4.5.3 Typedef Documentation

4.5.3.1 MESSAGE

```
typedef MsgData MESSAGE
```

4.5.3.2 MESSAGEID

```
typedef UINT16 MESSAGEID
```

4.5.3.3 MsgData

```
typedef void const* MsgData
```

4.5.3.4 MsgLock

```
typedef const UINT8* MsgLock
```

4.5.3.5 MSGLOCK

```
typedef MsgLock MSGLOCK
```

4.5.3.6 MSGSUBID

```
typedef UINT16 MSGSUBID
```

4.5.3.7 MSGTIMER

```
typedef UINT32 MSGTIMER
```

4.5.3.8 Task

```
typedef TASKPACK* Task
```

4.5.3.9 TASK

```
typedef Task TASK
```

4.5.3.10 TASKHANDLER

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

4.5.3.11 TASKPACK

```
typedef void** TASKPACK
```

4.5.4 Enumeration Type Documentation

4.5.4.1 anonymous enum

anonymous enum

BLE system message id.

Enumerator

LE_SYS_MSG_BUF_OVERFLOW	message buffer overflow
LE_SYS_MSG_TOP	top of system message id

4.5.5 Function Documentation

4.5.5.1 LeCancelAllMessage()

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

Cancel all message in queue.

Parameters

<i>task</i>	task.
<i>id</i>	message id.

Returns

0 is ok, others is error.

4.5.5.2 LeCancelAllSubMessage()

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

Cancel all sub message in queue.

Parameters

<i>task</i>	the task of recvice message.
<i>id</i>	message id.
<i>sub↔ id</i>	sub message id.

Returns

0 is ok, others is error.

4.5.5.3 LeCancelFirstMessage()

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

Cancel the first message in queue.

Parameters

<i>task</i>	task.
<i>id</i>	message id.

Returns

True is ok, false is error.

4.5.5.4 LeCancelFirstSubMessage()

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

Cancel the first sub message in queue.

Parameters

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

Returns

True is ok, false is error.

4.5.5.5 LeGetSubMsgId()

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

Get sub message id.

Parameters

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

Returns

0 is ok, others is error.

4.5.5.6 LeHostCreateTask()

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

Create BLE task.

Parameters

<i>task</i>	the reference of BLE task.
<i>hdl</i>	callback handle of BLE task.

Returns

TRUE is success, FALSE is failed.

4.5.5.7 LeHostMessageLoop()

```
void LeHostMessageLoop (
    void )
```

message loop run.

Returns

None.

4.5.5.8 LeSendMessage()

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

Send message to BLE task.

Parameters

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

Returns

None.

4.5.5.9 LeSendMessageAfter()

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

```

MESSAGEID msgId,
MESSAGE msg,
UINT32 delay )

```

Delay, then send message to BLE task.

Parameters

<i>task</i>	reference of BLE task.
<i>msg</i> ↔ <i>Id</i>	message ID.
<i>msg</i>	message.
<i>delay</i>	delay time, ms.

Returns

None.

4.5.5.10 LeSendMessageUnlock()

```

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

```

Send message until lock is 0.

Parameters

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

Returns

None.

4.5.5.11 LeSendSubMessage()

```

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

```

Send sub message.

Parameters

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

Returns

None.

4.5.5.12 LeSendSubMessageAfter()

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

Delay, then send sub message.

Parameters

<i>task</i>	the task of recvice message.
<i>msg↔ Id</i>	message id.
<i>subId</i>	sub message id.
<i>msg</i>	message.
<i>delay</i>	delay time.

Returns

None.

4.5.5.13 LeSendSubMessageUnlock()

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

Send sub message until lock is 0.

Parameters

<i>task</i>	the task of recvice message.
<i>id</i>	message id.
<i>sub</i> \leftrightarrow <i>Id</i>	sub message id.
<i>msg</i>	message.
<i>lock</i>	lock number.

Returns

None.

4.6 BLE SMP APIs

Data Structures

- struct [LE_SMP_MSG_ENCRYPTION_CHANGE_IND_T](#)
- struct [LE_SMP_MSG_ENCRYPTION_REFRESH_IND_T](#)
- struct [LE_SMP_MSG_OOB_DATA_REQUEST_IND_T](#)
- struct [LE_SMP_MSG_PAIRING_ACTION_IND_T](#)
- struct [LE_SMP_MSG_PAIRING_COMPLETE_IND_T](#)
- struct [LE_SMP_MSG_PASSKEY_DISPLAY_IND_T](#)
- struct [LE_SMP_MSG_PASSKEY_INPUT_IND_T](#)
- struct [LE_SMP_MSG_SC_OOB_DATA_REQUEST_IND_T](#)
- struct [LE_SMP_MSG_SLAVE_SECURITY_REQUEST_IND_T](#)
- struct [LE_SMP_MSG_USER_CONFIRM_IND_T](#)
- struct [LE_SMP_SC_OOB_DATA_T](#)

Macros

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

Enumerations

- enum {
[LE_SMP_MSG_SLAVE_SECURITY_REQUEST_IND](#) = [LE_SMP_MSG_BASE](#), [LE_SMP_MSG_PAIRING_ACTION_IND](#),
[LE_SMP_MSG_PASSKEY_DISPLAY_IND](#), [LE_SMP_MSG_PASSKEY_INPUT_IND](#),
[LE_SMP_MSG_OOB_DATA_REQUEST_IND](#), [LE_SMP_MSG_SC_OOB_DATA_REQUEST_IND](#), [LE_SMP_MSG_USER_CONFIRM_IND](#),
[LE_SMP_MSG_ENCRYPTION_CHANGE_IND](#),
[LE_SMP_MSG_ENCRYPTION_REFRESH_IND](#), [LE_SMP_MSG_PAIRING_COMPLETE_IND](#), [LE_SMP_LONG_TERM_KEY_IND](#),
[LE_SMP_KEYS_IND](#),
[LE_SMP_MSG_TOP](#) }
BLE SMP message id.
- enum {
[LE_SMP_PAIR_JUST_WORK](#), [LE_SMP_PAIR_OOB](#), [LE_SMP_PAIR_PASSKEY_INPUT](#), [LE_SMP_PAIR_DISPLAY](#),
[LE_SMP_PAIR_NUM_COMPARE](#) }

Functions

- void [LeSmpInit](#) (TASK appTask)
BLE SMP Module Init.
- void [LeSmpOobAuthDataRsp](#) (UINT16 conn_hdl, UINT8 *data, UINT16 len)
SMP OOB authenticate data response.
- UINT16 [LeSmpOobPresent](#) (UINT16 conn_hdl, BOOL oob_present)
SMP OOB present.
- void [LeSmpPasskeyInput](#) (UINT16 conn_hdl, UINT32 passkey)
Input passkey.
- UINT16 [LeSmpScOobComputeConfirmVal](#) (UINT8 *rand, UINT8 *confirm)
SMP secure connection OOB compute confirm value.
- void [LeSmpScOobDataRsp](#) (UINT16 conn_hdl, UINT8 *our_rand, [LE_SMP_SC_OOB_DATA_T](#) *peer)
OOB data response.
- UINT16 [LeSmpSecurityReq](#) (UINT16 conn_hdl)
BLE SMP security request.
- UINT16 [LeSmpSecurityRsp](#) (UINT16 conn_hdl, BOOL accept)
BLE SMP security request.
- UINT16 [LeSmpSetDefaultConfig](#) (UINT8 iocap, BOOL mitm, BOOL sc, BOOL bond)
Set default configure for pairing.
- UINT16 [LeSmpUserConfirmRsp](#) (UINT16 conn_hdl, BOOL accept)
User confirm response.

4.6.1 Detailed Description

4.6.2 Macro Definition Documentation

4.6.2.1 LE_MAX_BOND_COUNT

```
#define LE_MAX_BOND_COUNT 8
```

4.6.2.2 LE_SM_IO_CAP_DISP_ONLY

```
#define LE_SM_IO_CAP_DISP_ONLY 0x00
```

display only

4.6.2.3 LE_SM_IO_CAP_DISP_YES_NO

```
#define LE_SM_IO_CAP_DISP_YES_NO 0x01
```

display + yes or no

4.6.2.4 LE_SM_IO_CAP_KEYBOARD_DISP

```
#define LE_SM_IO_CAP_KEYBOARD_DISP 0x04
```

display + keyboard

4.6.2.5 LE_SM_IO_CAP_KEYBOARD_ONLY

```
#define LE_SM_IO_CAP_KEYBOARD_ONLY 0x02
```

keyboard only

4.6.2.6 LE_SM_IO_CAP_NO_IO

```
#define LE_SM_IO_CAP_NO_IO 0x03
```

no input and output

4.6.2.7 LE_SM_PAIR_MITM_NO

```
#define LE_SM_PAIR_MITM_NO 0x00
```

4.6.2.8 LE_SM_PAIR_MITM_YES

```
#define LE_SM_PAIR_MITM_YES 0x01
```

4.6.2.9 LE_SM_PAIR_OOB_NO

```
#define LE_SM_PAIR_OOB_NO 0x00
```

4.6.2.10 LE_SM_PAIR_OOB_YES

```
#define LE_SM_PAIR_OOB_YES 0x01
```

4.6.2.11 LE_SM_PAIR_SC_NO

```
#define LE_SM_PAIR_SC_NO 0x00
```

4.6.2.12 LE_SM_PAIR_SC_YES

```
#define LE_SM_PAIR_SC_YES 0x01
```

4.6.3 Enumeration Type Documentation

4.6.3.1 anonymous enum

```
anonymous enum
```

BLE SMP message id.

Enumerator

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

4.6.3.2 anonymous enum

```
anonymous enum
```

Enumerator

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

4.6.4 Function Documentation

4.6.4.1 LeSmpInit()

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

BLE SMP Module Init.

Parameters

<i>appTask</i>	the reference of BLE task.
----------------	----------------------------

Returns

None.

4.6.4.2 LeSmpOobAuthDataRsp()

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

SMP OOB authenticate data response.

Parameters

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

Returns

None.

4.6.4.3 LeSmpOobPresent()

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

SMP OOB present.

Parameters

<i>conn_hdl</i>	connection handle.
<i>oob_present</i>	present or not.

Returns

0 is Ok, others refer to SMP_ERR_* in [ble_err.h](#).

4.6.4.4 LeSmpPasskeyInput()

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

Input passkey.

Parameters

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

Returns

None.

4.6.4.5 LeSmpScOobComputeConfirmVal()

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

SMP secure connection OOB compute confirm value.

Parameters

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

Returns

0 is Ok, others refer to SMP_ERR_* in [ble_err.h](#).

4.6.4.6 LeSmpScOobDataRsp()

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

OOB data response.

Parameters

<i>conn_hdl</i>	connection handle.
<i>our_rand</i>	our random data.
<i>peer</i>	peer OOB data.

Returns

None.

4.6.4.7 LeSmpSecurityReq()

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

BLE SMP security request.

Parameters

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

Returns

0 is Ok, others refer to SMP_ERR_* in [ble_err.h](#).

4.6.4.8 LeSmpSecurityRsp()

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

BLE SMP security request.

Parameters

<i>conn_hdl</i>	connection handle.
<i>accept</i>	TRUE is accept, FALSE is not.

Returns

0 is Ok, others refer to SMP_ERR_* in [ble_err.h](#).

4.6.4.9 LeSmpSetDefaultConfig()

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

Set default configure for pairing.

Parameters

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

Returns

0 is Ok, others refer to SMP_ERR_* in [ble_err.h](#).

4.6.4.10 LeSmpUserConfirmRsp()

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

User confirm response.

Parameters

<i>conn_hdl</i>	connection handle.
<i>accept</i>	yes or no.

Returns

0 is Ok, others refer to SMP_ERR_* in [ble_err.h](#).

4.7 WIFI APIS

WIFI APIS.

Modules

- [WIFI Common APIs](#)
- [WIFI STA APIs](#)
- [Enumeration](#)

Data Structures

- struct [wifi_active_scan_time_t](#)
Range of active scan times per channel.
- struct [wifi_ap_config_t](#)
This structure is the Wi-Fi configuration for initialization for Soft-AP mode.
- struct [wifi_auto_connect_info_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_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_ENABLE, WIFI_AUTO_CONNECT_DISABLE }`
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)`
install 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 handler (internal use)

4.7.1 Detailed Description

WIFI APIs.

4.7.2 Macro Definition Documentation

4.7.2.1 WIFI_BEACON_INTERVAL_LENGTH

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

Beacon interval length in a frame header.

4.7.2.2 WIFI_CAPABILITY_INFO_LENGTH

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

Length of capability information in a frame header.

4.7.2.3 WIFI_LENGTH_802_11

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

Length of 802.11 MAC header.

4.7.2.4 WIFI_LENGTH_PASSPHRASE

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

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

4.7.2.5 WIFI_MAC_ADDRESS_LENGTH

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

MAC address length.

4.7.2.6 WIFI_MAX_LENGTH_OF_SSID

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

The maximum length of SSID.

4.7.2.7 WIFI_MAX_SCAN_AP_NUM

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

maximum number of ap list items which can stored

4.7.2.8 WIFI_MAX_SUPPORTED_RATES

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

maximum number of supported data rate

4.7.3 Typedef Documentation

4.7.3.1 wifi_ap_record_t

```
typedef wifi_scan_info_t wifi_ap_record_t
```

4.7.3.2 wifi_event_notify_cb_t

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

4.7.4 Enumeration Type Documentation

4.7.4.1 wifi_auto_connet_mode_e

```
enum wifi_auto_connet_mode_e
```

WiFi auto connect mode parameters.

Enumerator

WIFI_AUTO_CONNECT_ENABLE	
WIFI_AUTO_CONNECT_DISABLE	

4.7.5 Function Documentation

4.7.5.1 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	<i>event</i>	event type Set the event type,Options are <ul style="list-style-type: none"> • 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	<i>payload</i>	Data block that transmitted to event
in	<i>length</i>	The length of data block

Returns

0 : success
other : failed

4.7.5.2 wifi_install_default_event_handlers()

```
void wifi_install_default_event_handlers (
    void )
```

install default event handler for wifi event (internal use)

4.7.5.3 wifi_register_event_handler()

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

register wifi event handler (internal use)

Parameters

in	<i>idx</i>	one of the enums of bt_scan_mode_t
in	<i>handler</i>	the Wi-Fi event handler

Returns

0 : success
other : failed

4.8 WIFI Common APIs

Data Structures

- struct [event_msg_t](#)
Send information to event by [event_msg_t](#).
- union [wifi_event_info_t](#)
[wifi_event_info_t](#)
- struct [wifi_event_sta_connected_t](#)
[wifi_event_sta_connected_t](#)
- struct [wifi_event_sta_disconnected_t](#)
[wifi_event_sta_disconnected_t](#)
- struct [wifi_event_sta_got_ip_t](#)
- struct [wifi_event_sta_scan_done_t](#)
[wifi_event_sta_scan_done_t](#)

Typedefs

- typedef int(* [wifi_event_cb_t](#)) ([wifi_event_id_t](#) event, void *data, uint16_t length)
Application specified event callback function.

Functions

- int [wifi_event_loop_init](#) ([wifi_event_cb_t](#) cb)
Event Loop Initialization Create the event handler and call back funtion.
- int [wifi_event_loop_send](#) ([event_msg_t](#) *msg)
Send an event to event task.
- void [wifi_event_loop_set_cb](#) ([wifi_event_cb_t](#) cb, void *ctx)
Set application specified event callback function.
- int [wifi_event_process_handler](#) ([wifi_event_t](#) event, uint8_t *payload, uint32_t length)
Default event handler for system events.

4.8.1 Detailed Description

4.8.2 Typedef Documentation

4.8.2.1 [wifi_event_cb_t](#)

```
typedef int(* wifi\_event\_cb\_t) (wifi\_event\_id\_t event, void *data, uint16_t length)
```

Application specified event callback function.

4.8.3 Function Documentation

4.8.3.1 [wifi_event_loop_init\(\)](#)

```
int wifi\_event\_loop\_init (  
    wifi\_event\_cb\_t cb )
```

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

Parameters

<code>cb</code>	: application specified event callback
-----------------	--

Returns

0 : success
other : failed

4.8.3.2 wifi_event_loop_send()

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

Send an event to event task.

Attention

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

Parameters

<code>event_msg_t</code>	* msg: Send information to event by msg
--------------------------	---

Returns

0 : success
other : failed

4.8.3.3 wifi_event_loop_set_cb()

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

Set application specified event callback function.

Attention

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

Parameters

<i>wifi_event_cb_t</i>	cb : callback
<i>void</i>	*ctx : reserved for user

4.8.3.4 `wifi_event_process_handler()`

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

Default event handler for system events.

This function performs default handling of system events.

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

Parameters

in	<i>event</i>	event type Set the event type,Options are <ul style="list-style-type: none"> • 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	<i>payload</i>	Data block transmitted to event
in	<i>length</i>	The length of the data block

Returns

0 : success
other : failed

4.9 WIFI STA APIs

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_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_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_opmode_fp_t) (uint8_t *mode)`
- `typedef int(* wifi_config_get_ssid_fp_t) (uint8_t *ssid, uint8_t *ssid_length)`
- `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_opmode_fp_t) (uint8_t mode)`
- `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_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) (int wpa_pro, 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 number of AP information.
- int [wifi_auto_connect_get_mode](#) (uint8_t *mode)
Get the auto connect mode.
- 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 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_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_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_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_opmode](#) (uint8_t mode)
- int [wifi_config_set_skip_dtim](#) (uint8_t value)
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_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](#) (int wpa_pro, 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_scan_stop](#) (void)
Stop scanning process.
- int [wifi_scan_start](#) (const [wifi_scan_config_t](#) *config, bool block)
Scan all available APs. After invoke the [wifi_set_config\(\)](#) and [wifi_start\(\)](#), then call [wifi_scan_start\(\)](#) to scan APs.
- int [wifi_set_config](#) ([wifi_mode_t](#) interface, [wifi_config_t](#) *conf)
Set configuration of OPL1000 STA.
- int [wifi_sta_get_ap_info](#) ([wifi_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_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_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_opmode_fp_t](#) [wifi_config_get_opmode_api](#)
- [wifi_config_get_ssid_fp_t](#) [wifi_config_get_ssid_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_opmode_fp_t](#) [wifi_config_set_opmode_api](#)
- [wifi_config_set_ssid_fp_t](#) [wifi_config_set_ssid_api](#)
- [wifi_connection_connect_fp_t](#) [wifi_connection_connect_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](#)

4.9.1 Detailed Description

4.9.2 Macro Definition Documentation

4.9.2.1 WIFI_READY_TIME

```
#define WIFI_READY_TIME 2000
```

4.9.3 Typedef Documentation

4.9.3.1 wifi_auto_connect_clear_ap_info_fp_t

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

4.9.3.2 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)
```

4.9.3.3 wifi_auto_connect_get_ap_num_fp_t

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

4.9.3.4 wifi_auto_connect_get_mode_fp_t

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

4.9.3.5 wifi_auto_connect_init_fp_t

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


4.9.3.6 wifi_auto_connect_reset_fp_t

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

4.9.3.7 wifi_auto_connect_set_ap_num_fp_t

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

4.9.3.8 wifi_auto_connect_set_mode_fp_t

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

4.9.3.9 wifi_auto_connect_start_fp_t

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

4.9.3.10 wifi_config_get_bandwidth_fp_t

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

4.9.3.11 wifi_config_get_bssid_fp_t

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

4.9.3.12 wifi_config_get_channel_fp_t

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

4.9.3.13 wifi_config_get_dtim_interval_fp_t

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

4.9.3.14 `wifi_config_get_listen_interval_fp_t`

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

4.9.3.15 `wifi_config_get_mac_address_fp_t`

```
typedef int(* wifi_config_get_mac_address_fp_t) (wifi\_mode\_t interface, uint8_t *address)
```

4.9.3.16 `wifi_config_get_opmode_fp_t`

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

4.9.3.17 `wifi_config_get_ssid_fp_t`

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

4.9.3.18 `wifi_config_set_bandwidth_fp_t`

```
typedef int(* wifi_config_set_bandwidth_fp_t) (wifi\_mode\_t interface, wifi\_bandwidth\_t bandwidth)
```

4.9.3.19 `wifi_config_set_bssid_fp_t`

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

4.9.3.20 `wifi_config_set_channel_fp_t`

```
typedef int(* wifi_config_set_channel_fp_t) (wifi\_mode\_t interface, uint8_t channel)
```

4.9.3.21 `wifi_config_set_dtim_interval_fp_t`

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

4.9.3.22 wifi_config_set_listen_interval_fp_t

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

4.9.3.23 wifi_config_set_mac_address_fp_t

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

4.9.3.24 wifi_config_set_opmode_fp_t

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

4.9.3.25 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)
```

4.9.3.26 wifi_connection_connect_fp_t

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

4.9.3.27 wifi_connection_disconnect_ap_fp_t

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

4.9.3.28 wifi_connection_disconnect_sta_fp_t

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

4.9.3.29 `wifi_connection_get_rssi_fp_t`

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

4.9.3.30 `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)
```

4.9.3.31 `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)
```

4.9.3.32 `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)
```

4.9.3.33 `wifi_convert_auth_mode_fp_t`

```
typedef int(* wifi_convert_auth_mode_fp_t) (int wpa_pro, int privacy)
```

4.9.3.34 `wifi_deinit_fp_t`

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

4.9.3.35 `wifi_event_handler_t`

```
typedef int32_t(* wifi_event_handler_t) (wifi\_event\_t event, uint8_t *payload, uint32_t length)
```

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

Parameters

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

Returns

The return value is reserved and it is ignored.

4.9.3.36 `wifi_fast_connect_get_mode_fp_t`

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

4.9.3.37 `wifi_fast_connect_set_mode_fp_t`

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

4.9.3.38 `wifi_fast_connect_start_fp_t`

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

4.9.3.39 `wifi_get_config_fp_t`

```
typedef int(* wifi_get_config_fp_t) (wifi\_mode\_t interface, wifi\_config\_t *conf)
```

4.9.3.40 `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

<code>ctx</code>	is context pointer that provided to <code>wifi_init()</code> . It will be passed back to the callback.
------------------	--

4.9.3.41 `wifi_init_fp_t`

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

4.9.3.42 `wifi_result_t`

```
typedef int32_t wifi_result_t
```

4.9.3.43 `wifi_scan_get_ap_list_fp_t`

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

4.9.3.44 `wifi_scan_get_ap_num_fp_t`

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

4.9.3.45 `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)
```

4.9.3.46 `wifi_scan_start_fp_t`

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

4.9.3.47 wifi_scan_stop_fp_t

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

4.9.3.48 wifi_set_config_fp_t

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

4.9.3.49 wifi_sta_get_ap_info_fp_t

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

4.9.3.50 wifi_start_fp_t

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

4.9.3.51 wifi_stop_fp_t

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

4.9.4 Function Documentation**4.9.4.1 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	<i>index</i>	The index of AP position <ul style="list-style-type: none">• Range is 0 to 2
----	--------------	--

Returns

0 : success
other : failed

4.9.4.2 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	<i>index</i>	The index of AP position <ul style="list-style-type: none">• Range is 0 to 2
out	<i>mode</i>	Get the AP information

Returns

0 : success
other : failed

4.9.4.3 wifi_auto_connect_get_ap_num()

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

Get the number of AP information.

Attention

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

Parameters

out	mode	Get the number of AP information
-----	------	----------------------------------

Returns

0 : success
other : failed

4.9.4.4 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

4.9.4.5 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

4.9.4.6 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

4.9.4.7 wifi_auto_connect_set_ap_num()

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

Set the number of AP information.

Attention

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

Parameters

in	num	The number of AP information will be saved in flash. <ul style="list-style-type: none">• Range is 1 to 3
----	-----	--

Returns

0 : success
other : failed

4.9.4.8 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 <ul style="list-style-type: none">• WIFI_AUTO_CONNECT_ENABLE• WIFI_AUTO_CONNECT_DISABLE
----	------	--

Returns

0 : success
other : failed

4.9.4.9 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

4.9.4.10 wifi_config_get_bandwidth()

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

Get the bandwidth of OPL1000 specified interface.

Attention

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

Parameters

in	<i>interface</i>	Configure the current wifi working mode,The options are <ul style="list-style-type: none">• WIFI_MODE_STA• WIFI_MODE_AP (currently not support)
out	<i>bandwidth</i>	Get the bandwidth value of the current wifi module working through the pointer

Returns

0 : success
other : failed

4.9.4.11 wifi_config_get_bssid()

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

get bssid after scan

Parameters

out	<i>bssid</i>	the string of bssid
-----	--------------	---------------------

Returns

0 : success
other : failed

4.9.4.12 wifi_config_get_channel()

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

Get the primary/secondary channel of OPL1000.

Attention

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

Parameters

in	<i>interface</i>	Configure the current wifi working mode,The options are <ul style="list-style-type: none"> • WIFI_MODE_STA • WIFI_MODE_AP (currently not support)
out	<i>channel</i>	Get Current module wifi work channel number

Returns

0 : success
other : failed

4.9.4.13 wifi_config_get_dtim_interval()

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

4.9.4.14 wifi_config_get_listen_interval()

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

4.9.4.15 wifi_config_get_mac_address()

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

Get mac of specified interface.

Parameters

in	<i>interface</i>	Configure the current wifi working mode,The options are <ul style="list-style-type: none"> • WIFI_MODE_STA • WIFI_MODE_AP (currently not support)
out	<i>address</i>	Get the MAC address of the device through this interface, The address is similar to this structure: xx:xx:xx:xx:xx:xx

Returns

0 : success
other : failed

4.9.4.16 wifi_config_get_opmode()

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

4.9.4.17 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	<i>value</i>	Get the Skip DTIM value in current wifi setting
-----	--------------	---

Returns

0 : success
other : failed

4.9.4.18 wifi_config_get_ssid()

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

Get ssid value of AP.

Parameters

out	<i>ssid</i>	Get ssid by pointer
out	<i>ssid_length</i>	Get the length of the ssid character

Returns

0 : success
other : failed

4.9.4.19 wifi_config_set_bandwidth()

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

Set the bandwidth of OPL1000 specified interface.

Parameters

in	<i>interface</i>	Configure the current wifi working mode, The options are <ul style="list-style-type: none"> WIFI_MODE_STA WIFI_MODE_AP (currently not support)
in	<i>bandwidth</i>	Set the working bandwidth of wifi

Returns

0 : success
other : failed

4.9.4.20 wifi_config_set_bssid()

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

config OPL1000 Wi-Fi bssid.

Parameters

in	<i>bssid</i>	the string of bssid
----	--------------	---------------------

Returns

0 : success
other : failed

4.9.4.21 wifi_config_set_channel()

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

Set primary/secondary channel of OPL1000.

Attention

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

Parameters

in	<i>interface</i>	Configure the current wifi working mode, The options are <ul style="list-style-type: none"> • WIFI_MODE_STA • WIFI_MODE_AP (currently not support)
in	<i>channel</i>	Set current Wi-Fi work channel number

Returns

0 : success
other : failed

4.9.4.22 wifi_config_set_dtim_interval()

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

4.9.4.23 wifi_config_set_listen_interval()

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

4.9.4.24 wifi_config_set_mac_address()

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

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

Attention

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

Parameters

in	<i>interface</i>	Configure the current wifi working mode,The options are <ul style="list-style-type: none">• WIFI_MODE_STA• WIFI_MODE_AP (currently not support)
in	<i>address</i>	set MAC address

Returns

0 : success
other : failed

4.9.4.25 wifi_config_set_opmode()

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

4.9.4.26 wifi_config_set_skip_dtim()

```
int wifi_config_set_skip_dtim (
    uint8_t value )
```

Set the Skip DTIM value of OPL1000.

Parameters

in	<i>value</i>	Set the Skip DTIM 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

4.9.4.27 `wifi_config_set_ssid()`

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

Set the ssid value of the current device.

Parameters

in	<i>interface</i>	Configure the current wifi working mode, The options are <ul style="list-style-type: none"> • WIFI_MODE_STA • WIFI_MODE_AP (currently not support)
in	<i>ssid</i>	Set the value of ssid
in	<i>ssid_length</i>	The length of ssid parameter

Returns

0 : success
other : failed

4.9.4.28 `wifi_connection_connect()`

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

Connect OPL1000 Wi-Fi station to certain AP.

Attention

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

Parameters

in	<i>config</i>	Establish connection parameters
----	---------------	---------------------------------

Returns

0 : success
other : failed

4.9.4.29 wifi_connection_disconnect_ap()

```
int wifi_connection_disconnect_ap (
    void )
```

Disconnect the link between OPL1000 and connected AP.

Returns

0 : success
other : failed

4.9.4.30 wifi_connection_disconnect_sta()

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

Disconnect the link between the current device and the station.

Parameters

in	<i>address</i>	station address
----	----------------	-----------------

Returns

0 : success
other : failed

4.9.4.31 wifi_connection_get_rssi()

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

get signal strength of AP

Attention

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

Parameters

out	<i>rssi</i>	rssi value
-----	-------------	------------

Returns

0 : success
 other : failed

4.9.4.32 wifi_connection_register_event_handler()

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

register wifi call back handler

Parameters

in	<i>event</i>	The type of the registered event. Options are <ul style="list-style-type: none"> • 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	<i>handler</i>	registered event handler

Returns

0 : success
 other : failed

4.9.4.33 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 )
```

4.9.4.34 `wifi_connection_unregister_event_handler()`

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

unregister wifi call back handler

Parameters

in	<i>event</i>	The type of the unregistered event. Options please refer to wifi_connection_register_event_handler()
in	<i>handler</i>	unregistered event handler

Returns

0 : success
other : failed

4.9.4.35 `wifi_convert_auth_mode()`

```
int wifi_convert_auth_mode (
    int wpa_pro,
    int privacy )
```

4.9.4.36 `wifi_deinit()`

```
int wifi_deinit (
    void )
```

De-init Wi-Fi Initialization and Configuration functions.

Attention

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

Returns

0 : success
other : failed

4.9.4.37 `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	<i>index</i>	The index of AP position <ul style="list-style-type: none">• Range is 0 to 2
out	<i>mode</i>	Get the fast connect mode

Returns

0 : success
other : failed

4.9.4.38 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	<i>index</i>	The index of AP position <ul style="list-style-type: none">• Range is 0 to 2
in	<i>mode</i>	The fast connect mode

Returns

0 : success
other : failed

4.9.4.39 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	<i>index</i>	The index of AP position <ul style="list-style-type: none">• Range is 0 to 2
----	--------------	--

Returns

0 : success
other : failed

4.9.4.40 wifi_get_config()

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

Get configuration of specified interface.

Parameters

in	<i>interface</i>	Configure wifi working mode, The options are <ul style="list-style-type: none">• WIFI_MODE_STA• WIFI_MODE_AP (currently not support)
out	<i>conf</i>	return wifi's current operating parameters

Returns

0 : success
other : failed

4.9.4.41 wifi_init()

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

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

Attention

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

Parameters

in	<i>config</i>	pointer to Wi-Fi init configuration structure; can point to a temporary variable.
in	<i>init_cb</i>	pointer to Wi-Fi init complete configuration structure; can point to a temporary variable.

Returns

0 : success
other : failed

4.9.4.42 wifi_scan_get_ap_list()

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

Get list of APs that found in last scan operation.

Attention

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

Parameters

out	<i>scan_list</i>	store APs' informaton that found in last scan operation
-----	------------------	---

Returns

0 : success
other : failed

4.9.4.43 wifi_scan_get_ap_num()

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

Get the number of scanned APs.

Parameters

out	<i>number</i>	store number of APs found in last scan operation
-----	---------------	--

Attention

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

Returns

the scan result of AP number

4.9.4.44 wifi_scan_get_ap_records()

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

Get AP list found in last scan operation.

Parameters

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

Returns

0 : success
other : failed

4.9.4.45 wifi_scan_scan_stop()

```
int wifi_scan_scan_stop (
    void )
```

Stop scanning process.

Attention

This API shall be called after [wifi_scan_start\(\)](#)

Returns

0 : success
other : failed

4.9.4.46 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	<i>config</i>	Configure parameters for scan operation
in	<i>block</i>	if block is true, this API blocks the caller until scan operation is done, otherwise it returns immediately

Returns

0 : success
other : failed

4.9.4.47 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 means user want to check MAC address of certain AP.
3. OPL1000 is limited to working on one channel.

Parameters

in	<i>interface</i>	Configure wifi working mode, The options are <ul style="list-style-type: none"> • WIFI_MODE_STA • WIFI_MODE_AP (currently not support)
in	<i>conf</i>	structure of configuration paremeters

Returns

0 : success
other : failed

4.9.4.48 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	<i>ap_info</i>	get AP information from list
-----	----------------	------------------------------

Returns

0 : success
other : failed

4.9.4.49 wifi_start()

```
int wifi_start (
    void )
```

Start Wi-Fi working.

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

Returns

0 : success
other : failed

4.9.4.50 wifi_stop()

```
int wifi_stop (
    void )
```

Stop wifi working.

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

Returns

0 : success
other : failed

4.9.5 Variable Documentation

4.9.5.1 wifi_auto_connect_clear_ap_info_api

wifi_auto_connect_clear_ap_info_fp_t wifi_auto_connect_clear_ap_info_api

4.9.5.2 wifi_auto_connect_get_ap_info_api

wifi_auto_connect_get_ap_info_fp_t wifi_auto_connect_get_ap_info_api

4.9.5.3 wifi_auto_connect_get_ap_num_api

wifi_auto_connect_get_ap_num_fp_t wifi_auto_connect_get_ap_num_api

4.9.5.4 wifi_auto_connect_get_mode_api

wifi_auto_connect_get_mode_fp_t wifi_auto_connect_get_mode_api

4.9.5.5 wifi_auto_connect_init_api

wifi_auto_connect_init_fp_t wifi_auto_connect_init_api

4.9.5.6 wifi_auto_connect_reset_api

wifi_auto_connect_reset_fp_t wifi_auto_connect_reset_api

4.9.5.7 wifi_auto_connect_set_ap_num_api

wifi_auto_connect_set_ap_num_fp_t wifi_auto_connect_set_ap_num_api

4.9.5.8 wifi_auto_connect_set_mode_api

wifi_auto_connect_set_mode_fp_t wifi_auto_connect_set_mode_api

4.9.5.9 wifi_auto_connect_start_api

wifi_auto_connect_start_fp_t wifi_auto_connect_start_api

4.9.5.10 wifi_config_get_bandwidth_api

wifi_config_get_bandwidth_fp_t wifi_config_get_bandwidth_api

4.9.5.11 wifi_config_get_bssid_api

wifi_config_get_bssid_fp_t wifi_config_get_bssid_api

4.9.5.12 wifi_config_get_channel_api

wifi_config_get_channel_fp_t wifi_config_get_channel_api

4.9.5.13 wifi_config_get_dtim_interval_api

wifi_config_get_dtim_interval_fp_t wifi_config_get_dtim_interval_api

4.9.5.14 wifi_config_get_listen_interval_api

wifi_config_get_listen_interval_fp_t wifi_config_get_listen_interval_api

4.9.5.15 wifi_config_get_mac_address_api

wifi_config_get_mac_address_fp_t wifi_config_get_mac_address_api

4.9.5.16 wifi_config_get_opmode_api

wifi_config_get_opmode_fp_t wifi_config_get_opmode_api

4.9.5.17 `wifi_config_get_ssid_api`

`wifi_config_get_ssid_fp_t` `wifi_config_get_ssid_api`

4.9.5.18 `wifi_config_set_bandwidth_api`

`wifi_config_set_bandwidth_fp_t` `wifi_config_set_bandwidth_api`

4.9.5.19 `wifi_config_set_bssid_api`

`wifi_config_set_bssid_fp_t` `wifi_config_set_bssid_api`

4.9.5.20 `wifi_config_set_channel_api`

`wifi_config_set_channel_fp_t` `wifi_config_set_channel_api`

4.9.5.21 `wifi_config_set_dtim_interval_api`

`wifi_config_set_dtim_interval_fp_t` `wifi_config_set_dtim_interval_api`

4.9.5.22 `wifi_config_set_listen_interval_api`

`wifi_config_set_listen_interval_fp_t` `wifi_config_set_listen_interval_api`

4.9.5.23 `wifi_config_set_mac_address_api`

`wifi_config_set_mac_address_fp_t` `wifi_config_set_mac_address_api`

4.9.5.24 `wifi_config_set_opmode_api`

`wifi_config_set_opmode_fp_t` `wifi_config_set_opmode_api`

4.9.5.25 wifi_config_set_ssid_api

wifi_config_set_ssid_fp_t wifi_config_set_ssid_api

4.9.5.26 wifi_connection_connect_api

wifi_connection_connect_fp_t wifi_connection_connect_api

4.9.5.27 wifi_connection_disconnect_ap_api

wifi_connection_disconnect_ap_fp_t wifi_connection_disconnect_ap_api

4.9.5.28 wifi_connection_disconnect_sta_api

wifi_connection_disconnect_sta_fp_t wifi_connection_disconnect_sta_api

4.9.5.29 wifi_connection_get_rssi_api

wifi_connection_get_rssi_fp_t wifi_connection_get_rssi_api

4.9.5.30 wifi_connection_register_event_handler_api

wifi_connection_register_event_handler_fp_t wifi_connection_register_event_handler_api

4.9.5.31 wifi_connection_scan_start_api

wifi_connection_scan_start_fp_t wifi_connection_scan_start_api

4.9.5.32 wifi_connection_unregister_event_handler_api

wifi_connection_unregister_event_handler_fp_t wifi_connection_unregister_event_handler_api

4.9.5.33 `wifi_convert_auth_mode_api`

`wifi_convert_auth_mode_fp_t` `wifi_convert_auth_mode_api`

4.9.5.34 `wifi_deinit_api`

`wifi_deinit_fp_t` `wifi_deinit_api`

4.9.5.35 `wifi_fast_connect_get_mode_api`

`wifi_fast_connect_get_mode_fp_t` `wifi_fast_connect_get_mode_api`

4.9.5.36 `wifi_fast_connect_set_mode_api`

`wifi_fast_connect_set_mode_fp_t` `wifi_fast_connect_set_mode_api`

4.9.5.37 `wifi_fast_connect_start_api`

`wifi_fast_connect_start_fp_t` `wifi_fast_connect_start_api`

4.9.5.38 `wifi_get_config_api`

`wifi_get_config_fp_t` `wifi_get_config_api`

4.9.5.39 `wifi_init_api`

`wifi_init_fp_t` `wifi_init_api`

4.9.5.40 `wifi_scan_get_ap_list_api`

`wifi_scan_get_ap_list_fp_t` `wifi_scan_get_ap_list_api`

4.9.5.41 wifi_scan_get_ap_num_api

wifi_scan_get_ap_num_fp_t wifi_scan_get_ap_num_api

4.9.5.42 wifi_scan_get_ap_records_api

wifi_scan_get_ap_records_fp_t wifi_scan_get_ap_records_api

4.9.5.43 wifi_scan_start_api

wifi_scan_start_fp_t wifi_scan_start_api

4.9.5.44 wifi_scan_stop_api

wifi_scan_stop_fp_t wifi_scan_stop_api

4.9.5.45 wifi_set_config_api

wifi_set_config_fp_t wifi_set_config_api

4.9.5.46 wifi_sta_get_ap_info_api

wifi_sta_get_ap_info_fp_t wifi_sta_get_ap_info_api

4.9.5.47 wifi_start_api

wifi_start_fp_t wifi_start_api

4.9.5.48 wifi_stop_api

wifi_stop_fp_t wifi_stop_api

4.10 Enumeration

Enumerations

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

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

- enum `wifi_bandwidth_t` { `WIFI_BW_HT20` = 1, `WIFI_BW_HT40` }
- enum `wifi_cipher_type_t` {
`WIFI_CIPHER_TYPE_NONE` = 0, `WIFI_CIPHER_TYPE_WEP40`, `WIFI_CIPHER_TYPE_WEP104`,
`WIFI_CIPHER_TYPE_TKIP`,
`WIFI_CIPHER_TYPE_CCMP`, `WIFI_CIPHER_TYPE_TKIP_CCMP`, `WIFI_CIPHER_TYPE_UNKNOWN` }

This enumeration defines wireless security cipher suits.

- enum `wifi_event_t` {
`WIFI_EVENT_NONE` = -1, `WIFI_EVENT_INIT_COMPLETE` = 0, `WIFI_EVENT_SCAN_COMPLETE`,
`WIFI_EVENT_STA_START`,
`WIFI_EVENT_STA_STOP`, `WIFI_EVENT_STA_CONNECTED`, `WIFI_EVENT_STA_DISCONNECTED`,
`WIFI_EVENT_STA_CONNECTION_FAILED`,
`WIFI_EVENT_STA_GOT_IP`, `WIFI_EVENT_STA_AUTO_CONNECT_FAILED`, `WIFI_EVENT_MAX` }

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

- enum `wifi_mode_t` { `WIFI_MODE_NULL` = 0, `WIFI_MODE_STA`, `WIFI_MODE_AP`, `WIFI_MODE_MAX` }
- enum `wifi_reason_code_t` {
`WIFI_REASON_CODE_SUCCESS`, `WIFI_REASON_CODE_FIND_AP_FAIL`, `WIFI_REASON_CODE_PREV_AUTH_INVALID`,
`WIFI_REASON_CODE_DEAUTH_LEAVING_BSS`,
`WIFI_REASON_CODE_DISASSOC_INACTIVITY`, `WIFI_REASON_CODE_DISASSOC_AP_OVERLOAD`,
`WIFI_REASON_CODE_CLASS_2_ERR`, `WIFI_REASON_CODE_CLASS_3_ERR`,
`WIFI_REASON_CODE_DISASSOC_LEAVING_BSS`, `WIFI_REASON_CODE_ASSOC_BEFORE_AUTH`,
`WIFI_REASON_CODE_DISASSOC_PWR_CAP_UNACCEPTABLE`, `WIFI_REASON_CODE_DISASSOC_SUP_CHS_UNACQ`,
`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` }

4.10.1 Detailed Description

4.10.2 Enumeration Type Documentation

4.10.2.1 wifi_auth_mode_t

```
enum wifi_auth_mode_t
```

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

Enumerator

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

4.10.2.2 wifi_bandwidth_t

```
enum wifi_bandwidth_t
```

Enumerator

WIFI_BW_HT20	Bandwidth is HT20
WIFI_BW_HT40	Bandwidth is HT40

4.10.2.3 wifi_cipher_type_t

```
enum wifi_cipher_type_t
```

This enumeration defines wireless security cipher suits.

Enumerator

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

4.10.2.4 wifi_event_t

```
enum wifi_event_t
```

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

Enumerator

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

4.10.2.5 wifi_mode_t

```
enum wifi_mode_t
```

Enumerator

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

4.10.2.6 wifi_reason_code_t

```
enum wifi_reason_code_t
```

This enumeration defines the reason code of the WIFI_EVENT_STA_CONNECTION_FAILED event in [wifi_event_t](#). Find the details for the reason code below.

Enumerator

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

Enumerator

WIFI_REASON_CODE_DISASSOC_LEAVING_BSS	8 Disassociated because sending STA is leaving (or has left) BSS.
WIFI_REASON_CODE_ASSOC_BEFORE_AUTH	9 STA requesting (re)association is not authenticated with responding STA.
WIFI_REASON_CODE_DISASSOC_PWR_CAP_↔ UNACCEPTABLE	10 Disassociated because the information in the Power Capability element is unacceptable.
WIFI_REASON_CODE_DISASSOC_SUP_CHS_U↔ NACCEPTABLE	11 Disassociated because the information in the Supported Channels element is unacceptable.
WIFI_REASON_CODE_INVALID_INFO_ELEM	13 Invalid information element.
WIFI_REASON_CODE_MIC_FAILURE	14 Message integrity code (MIC) failure.
WIFI_REASON_CODE_4_WAY_HANDSHAKE_TI↔ MEOUT	15 4-Way Handshake time out.
WIFI_REASON_CODE_GROUP_KEY_UPDATE_↔ TIMEOUT	16 Group Key Handshake time out.
WIFI_REASON_CODE_DIFFERENT_INFO_ELEM	17 Information element in 4-Way Handshake different from (Re)Association Request/Probe Response/Beacon frame.
WIFI_REASON_CODE_GROUP_CIPHER_INVALID↔ 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.

4.10.2.7 wifi_scan_method_t

```
enum wifi_scan_method_t
```

Enumerator

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

4.10.2.8 wifi_scan_type_t

```
enum wifi_scan_type_t
```

This enumeration defines the wireless STA scan type.

Enumerator

WIFI_SCAN_TYPE_ACTIVE	Actively scan a network by sending 802.11 probe(s)
WIFI_SCAN_TYPE_PASSIVE	Passively scan a network by listening for beacons from APs
WIFI_SCAN_TYPE_MIX	Active + Passive scan

4.10.2.9 wifi_sort_method_t

```
enum wifi_sort_method_t
```

Enumerator

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

Chapter 5

Data Structure Documentation

5.1 `_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`

5.1.1 Field Documentation

5.1.1.1 `capabilities`

```
int capabilities
```

5.1.1.2 `group_cipher`

```
int group_cipher
```

5.1.1.3 key_mgmt

```
int key_mgmt
```

5.1.1.4 mgmt_group_cipher

```
int mgmt_group_cipher
```

5.1.1.5 num_pmkid

```
size_t num_pmkid
```

5.1.1.6 pairwise_cipher

```
int pairwise_cipher
```

5.1.1.7 pmkid

```
const u8* pmkid
```

5.1.1.8 proto

```
int proto
```

5.2 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](#)

5.2.1 Field Documentation

5.2.1.1 eap_workaround

unsigned int eap_workaround

5.2.1.2 eapol_flags

int eapol_flags

5.2.1.3 group_cipher

int group_cipher

5.2.1.4 key_mgmt

int key_mgmt

5.2.1.5 leap

int leap

5.2.1.6 mgmt_group_cipher

```
int mgmt_group_cipher
```

5.2.1.7 non_leap

```
int non_leap
```

5.2.1.8 pairwise_cipher

```
int pairwise_cipher
```

5.2.1.9 passphrase

```
char* passphrase
```

5.2.1.10 proto

```
int proto
```

5.2.1.11 psk

```
u8 psk[32]
```

5.2.1.12 psk_set

```
int psk_set
```

5.3 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]

5.3.1 Field Documentation

5.3.1.1 ap_channel

u8 ap_channel

5.3.1.2 beacon_interval

u16 beacon_interval

5.3.1.3 bssid

u8 bssid[MAC_ADDR_LEN]

5.3.1.4 capabilities

u16 capabilities

5.3.1.5 dtim_prod

u8 dtim_prod

5.3.1.6 fast_connect

u8 fast_connect

5.3.1.7 free_ocpy

bool free_ocpy

5.3.1.8 hid_ssid

s8 hid_ssid[IEEE80211_MAX_SSID_LEN+1]

5.3.1.9 hid_ssid_len

u8 hid_ssid_len

5.3.1.10 latest_beacon_rx_time

u64 latest_beacon_rx_time

5.3.1.11 passphrase

s8 passphrase[MAX_LEN_OF_PASSPHRASE]

5.3.1.12 psk

u8 psk[32]

5.3.1.13 rsn_ie

u8 rsn_ie[256]

5.3.1.14 rssi

s8 rssi

5.3.1.15 ssid

s8 ssid[IEEE80211_MAX_SSID_LEN+1]

5.3.1.16 ssid_len

u8 ssid_len

5.3.1.17 supported_rates

u8 supported_rates[IEEE80211_MAX_SUPPORTED_RATES]

5.3.1.18 wpa_data

wpa_ie_data_t wpa_data

5.3.1.19 wpa_ie

u8 wpa_ie[257]

5.4 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](#)

5.4.1 Field Documentation

5.4.1.1 flag

bool flag

5.4.1.2 front

s8 front

5.4.1.3 max_save_num

u8 max_save_num

5.4.1.4 pFCInfo

[auto_conn_info_t](#)* pFCInfo

5.4.1.5 rear

s8 rear

5.4.1.6 retryCount

u8 retryCount

5.4.1.7 targetIdx

u8 targetIdx

5.4.1.8 uFCapNum

u32 uFCapNum

5.5 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](#)

5.5.1 Detailed Description

Send information to event by [event_msg_t](#).

5.5.2 Field Documentation

5.5.2.1 event

uint32_t event

event type

5.5.2.2 length

uint32_t length

Packet length

5.5.2.3 param

uint8_t* param

event paramment

5.6 hap_control_t Struct Reference

```
#include <controller_wifi_com_patch.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]

5.6.1 Field Documentation

5.6.1.1 hap_ap_info

[auto_conn_info_t](#)* hap_ap_info

5.6.1.2 hap_bitvector

u16 hap_bitvector

5.6.1.3 hap_en

u8 hap_en

5.6.1.4 hap_final_index

u8 hap_final_index

5.6.1.5 hap_index

u8 hap_index

5.6.1.6 hap_ssid

char hap_ssid[IEEE80211_MAX_SSID_LEN+1]

5.7 LE_BT_ADDR_T Struct Reference

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

Data Fields

- BD_ADDR [addr](#)
- UINT8 [type](#)

5.7.1 Field Documentation

5.7.1.1 addr

BD_ADDR addr

address

5.7.1.2 type

UINT8 type

address type

5.8 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](#)

5.8.1 Field Documentation

5.8.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.8.1.2 conn_interval

UINT16 conn_interval

connection interval

5.8.1.3 conn_latency

UINT16 conn_latency

connection latency

5.8.1.4 dev_id

UINT16 dev_id

device ID

5.8.1.5 peer_addr

BD_ADDR peer_addr

peer address

5.8.1.6 peer_addr_type

UINT8 peer_addr_type

peer address type

5.8.1.7 role

UINT8 role

master or slave

5.8.1.8 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.8.1.9 supervison_timeout

UINT16 supervison_timeout

supervision timeout

5.9 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](#)

5.9.1 Field Documentation

5.9.1.1 addr

BD_ADDR addr

address

5.9.1.2 addr_type

UINT8 addr_type

address type

5.9.1.3 data

UINT8 data[1]

5.9.1.4 event_type

UINT8 event_type

5.9.1.5 len

UINT8 len

5.9.1.6 rssi

INT8 rssi

RSSI

5.10 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](#)

5.10.1 Field Documentation

5.10.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.10.1.2 itv_max

UINT16 itv_max

maximum connection interval

5.10.1.3 itv_min

UINT16 itv_min

minimum connection interval

5.10.1.4 latency

UINT16 latency

slave latency

5.10.1.5 sv_tmo

UINT32 sv_tmo

supervision timeout

5.11 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](#)

5.11.1 Field Documentation

5.11.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.11.1.2 interval

UINT16 interval

connection interval

5.11.1.3 latency

UINT16 latency

slave letency

5.11.1.4 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.11.1.5 supervision_timeout

UINT32 supervision_timeout

supervision timeout

5.12 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](#)

5.12.1 Field Documentation

5.12.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.12.1.2 max_rx_octets

UINT16 max_rx_octets

connMaxRxOctets

5.12.1.3 max_rx_time

UINT16 max_rx_time

connMaxRxTime

5.12.1.4 max_tx_octets

UINT16 max_tx_octets

connMaxTxOctets

5.12.1.5 max_tx_time

UINT16 max_tx_time

connMaxTxTime

5.13 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](#)

5.13.1 Field Documentation

5.13.1.1 direct_addr

BD_ADDR direct_addr

direct address

5.13.1.2 direct_addr_type

UINT8 direct_addr_type

direct address type

5.13.1.3 peer_addr

BD_ADDR peer_addr

peer address

5.13.1.4 peer_addr_type

UINT8 peer_addr_type

peer address type

5.13.1.5 rssi

INT8 rssi

RSSI

5.14 LE_CM_MSG_DISCONNECT_COMPLETE_IND_T Struct Reference

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

Data Fields

- UINT16 [conn_hdl](#)
- UINT8 [reason](#)
- UINT16 [status](#)

5.14.1 Field Documentation

5.14.1.1 conn_hdl

UINT16 [conn_hdl](#)

connection handle

5.14.1.2 reason

UINT8 [reason](#)

disconnect reason

5.14.1.3 status

UINT16 [status](#)

refer to LE_ERR_STATE in [ble_err.h](#)

5.15 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](#)

5.15.1 Field Documentation

5.15.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.15.1.2 devid

UINT16 devid

device ID

5.15.1.3 enabled

UINT8 enabled

5.15.1.4 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.16 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](#)

5.16.1 Field Documentation

5.16.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.16.1.2 devid

UINT16 devid

device ID

5.16.1.3 enabled

BOOL enabled

enable or disable

5.16.1.4 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.17 LE_CM_MSG_INIT_COMPLETE_CFM_T Struct Reference

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

Data Fields

- UINT16 [status](#)

5.17.1 Field Documentation

5.17.1.1 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.18 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]

5.18.1 Field Documentation

5.18.1.1 conn_hdl

UINT16 [conn_hdl](#)

connection handle

5.18.1.2 devid

UINT16 [devid](#)

device ID

5.18.1.3 ediv

UINT16 [ediv](#)

5.18.1.4 rand

UINT8 [rand](#)[8]

5.19 LE_CM_MSG_READ_ADV_TX_POWER_CFM_T Struct Reference

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

Data Fields

- INT8 [pwr_level](#)
- UINT16 [status](#)

5.19.1 Field Documentation

5.19.1.1 pwr_level

INT8 pwr_level

power level

5.19.1.2 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.20 LE_CM_MSG_READ_BD_ADDR_CFM_T Struct Reference

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

Data Fields

- BD_ADDR [bd_addr](#)
- UINT16 [status](#)

5.20.1 Field Documentation

5.20.1.1 bd_addr

BD_ADDR bd_addr

5.20.1.2 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.21 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](#)

5.21.1 Field Documentation

5.21.1.1 [ch_map](#)

```
UINT8 ch_map[5]
```

channel map

5.21.1.2 [conn_hdl](#)

```
UINT16 conn_hdl
```

connection handle

5.21.1.3 [status](#)

```
UINT16 status
```

refer to LE_ERR_STATE in [ble_err.h](#)

5.22 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](#)

5.22.1 Field Documentation

5.22.1.1 conn_hdl

UINT16 conn_hdl

5.22.1.2 rx_phy

UINT8 rx_phy

5.22.1.3 status

UINT16 status

5.22.1.4 tx_phy

UINT8 tx_phy

5.23 LE_CM_MSG_READ_RESOLVING_LIST_SIZE_CFM_T Struct Reference

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

Data Fields

- [UINT8 size](#)
- [UINT16 status](#)

5.23.1 Field Documentation

5.23.1.1 size

UINT8 size

resolving list size

5.23.1.2 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.24 LE_CM_MSG_READ_RSSI_CFM_T Struct Reference

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

Data Fields

- UINT16 [conn_hdl](#)
- INT8 [rssi](#)
- UINT16 [status](#)

5.24.1 Field Documentation

5.24.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.24.1.2 rssi

INT8 rssi

RSSI

5.24.1.3 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.25 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](#)

5.25.1 Field Documentation

5.25.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.25.1.2 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.25.1.3 tx_power

INT8 tx_power

tx power

5.26 LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM_T Struct Reference

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

Data Fields

- UINT8 [size](#)
- UINT16 [status](#)

5.26.1 Field Documentation

5.26.1.1 size

UINT8 size

white list size

5.26.1.2 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.27 LE_CM_MSG_SET_DATA_LENGTH_CFM_T Struct Reference

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

Data Fields

- UINT16 [conn_hdl](#)
- UINT16 [status](#)

5.27.1 Field Documentation

5.27.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.27.1.2 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.28 LE_CM_MSG_SET_DISCONNECT_CFM_T Struct Reference

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

Data Fields

- UINT16 [handle](#)
- UINT16 [status](#)

5.28.1 Field Documentation

5.28.1.1 handle

UINT16 handle

connection handle

5.28.1.2 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.29 LE_CM_MSG_SET_PHY_CFM_T Struct Reference

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

Data Fields

- UINT16 [conn_hdl](#)
- UINT16 [status](#)

5.29.1 Field Documentation

5.29.1.1 conn_hdl

UINT16 conn_hdl

5.29.1.2 status

UINT16 status

5.30 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](#)

5.30.1 Field Documentation

5.30.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.30.1.2 identifier

UINT16 identifier

5.30.1.3 interval_max

UINT16 interval_max

maximum connection interval

5.30.1.4 interval_min

UINT16 interval_min

minimum connection interval

5.30.1.5 slave_latency

UINT16 slave_latency

slave latency

5.30.1.6 timeout_multiplier

UINT32 timeout_multiplier

5.31 LE_CM_REQ_STATUS_T Struct Reference

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

Data Fields

- UINT16 [status](#)

5.31.1 Field Documentation

5.31.1.1 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.32 LE_CONN_PARA_T Struct Reference

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

Data Fields

- UINT16 [itv_max](#)
- UINT16 [itv_min](#)
- UINT16 [latency](#)
- UINT16 [sv_timeout](#)

5.32.1 Field Documentation

5.32.1.1 itv_max

UINT16 itv_max

maximum connection interval

5.32.1.2 itv_min

UINT16 itv_min

mininum connection interval

5.32.1.3 latency

UINT16 latency

slave latency

5.32.1.4 sv_timeout

UINT16 sv_timeout

supervision timeout

5.33 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](#)

5.33.1 Field Documentation

5.33.1.1 channel_map

UINT8 channel_map

advertising channel map

5.33.1.2 filter_policy

UINT8 filter_policy

advertising filter policy

5.33.1.3 interval_max

UINT16 interval_max

maximum advertising interval

5.33.1.4 interval_min

UINT16 interval_min

minimum advertising interval

5.33.1.5 own_addr_type

UINT8 own_addr_type

owner address type

5.33.1.6 peer_addr

BD_ADDR peer_addr

peer address

5.33.1.7 peer_addr_type

UINT8 peer_addr_type

peer address type

5.33.1.8 type

UINT8 type

advertising type

5.34 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](#)

5.34.1 Field Documentation

5.34.1.1 interval_max

UINT16 interval_max

maximum connection interval

5.34.1.2 interval_min

UINT16 interval_min

mininum connection interval

5.34.1.3 latency

UINT16 latency

slave latency

5.34.1.4 supervision_timeout

UINT16 supervision_timeout

supervision timeout for the LE Link

5.35 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](#)

5.35.1 Field Documentation

5.35.1.1 filter_policy

UINT8 filter_policy

scan filter policy

5.35.1.2 interval

UINT16 interval

scan interval

5.35.1.3 own_addr_type

UINT8 own_addr_type

owner address type

5.35.1.4 type

UINT8 type

scan type

5.35.1.5 window

UINT16 window

scan window

5.36 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](#)

5.36.1 Field Documentation

5.36.1.1 format

UINT8 format

UUID type

5.36.1.2 handle

UINT16 handle

handle

5.36.1.3 len

UINT16 len

value length

5.36.1.4 maxLen

UINT16 maxLen

maximum value length

5.36.1.5 permit

UINT16 permit

permit

5.36.1.6 pUuid

UINT16* const pUuid

UUID

5.36.1.7 pVal

UINT8* const pVal

value

5.37 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](#)

5.37.1 Field Documentation

5.37.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.37.1.2 devid

UINT16 devid

device index

5.37.1.3 handle

UINT16 handle

attribute handle

5.37.1.4 offset

UINT16 offset

attribute handle value

5.38 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](#)

5.38.1 Field Documentation

5.38.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.38.1.2 devid

UINT16 devid

device ID

5.38.1.3 flag

UINT8 flag

refer to LE_GATT_FLAG_* in [ble_gatt_if.h](#)

5.38.1.4 handle

UINT16 handle

attribute handle

5.38.1.5 len

UINT16 len

length written

5.38.1.6 offset

UINT16 offset

attribute handle value

5.38.1.7 pVal

UINT8* pVal

value written

5.39 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]

5.39.1 Field Documentation

5.39.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.39.1.2 devid

UINT16 devid

device ID

5.39.1.3 format

UINT8 format

UUID type

5.39.1.4 handle

UINT16 handle

characteristic descriptor handle

5.39.1.5 uuid

UINT16 uuid[8]

UUID

5.40 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](#)

5.40.1 Field Documentation

5.40.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.40.1.2 devid

UINT16 devid

device ID

5.40.1.3 format

UINT8 format

UUID type

5.40.1.4 handle

UINT16 handle

characteristic declaration handle

5.40.1.5 property

UINT8 property

property

5.40.1.6 uuid

UINT16 uuid[8]

UUID

5.40.1.7 val_hdl

UINT16 val_hdl

characteristic value handle

5.41 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](#)

5.41.1 Field Documentation

5.41.1.1 att_err

UINT8 att_err

0 is ok, others refer to LE_ATT_ERR_* in [ble_att_if.h](#)

5.41.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.41.1.3 devid

UINT16 devid

device ID

5.41.1.4 handle

UINT16 handle

characteristic value handle

5.41.1.5 len

UINT16 len

value length

5.41.1.6 offset

UINT16 offset

value position offset

5.41.1.7 val

UINT8* val

value

5.42 LE_GATT_MSG_CONFIRMATION_CFM_T Struct Reference

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

Data Fields

- UINT16 [conn_hdl](#)
- UINT16 [devid](#)
- UINT16 [handle](#)

5.42.1 Field Documentation

5.42.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.42.1.2 devid

UINT16 devid

device ID

5.42.1.3 handle

UINT16 handle

attribute handle

5.43 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](#)

5.43.1 Field Documentation

5.43.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.43.1.2 current_rx_mtu

UINT16 current_rx_mtu

current receive MTU

5.43.1.3 devid

UINT16 devid

device ID

5.44 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](#)

5.44.1 Field Documentation

5.44.1.1 client_rx_mtu

UINT16 client_rx_mtu

client receive MTU

5.44.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.44.1.3 devid

UINT16 devid

device ID

5.45 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](#)

5.45.1 Field Documentation

5.45.1.1 att_err

UINT8 att_err

0 is ok, others refer to LE_ATT_ERR_* in [ble_att_if.h](#)

5.45.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.45.1.3 devid

UINT16 devid

device ID

5.45.1.4 err_hdl

UINT16 err_hdl

TBD

5.45.1.5 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.46 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](#)

5.46.1 Field Documentation

5.46.1.1 att_err

UINT8 att_err

0 is ok, others refer to LE_ATT_ERR_* in [ble_att_if.h](#)

5.46.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.46.1.3 devid

UINT16 devid

device ID

5.46.1.4 handle

UINT16 handle

characteristic descriptor handle

5.46.1.5 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.47 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](#)

5.47.1 Field Documentation

5.47.1.1 att_err

UINT8 att_err

0 is ok, others refer to LE_ATT_ERR_* in [ble_att_if.h](#)

5.47.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.47.1.3 devid

UINT16 devid

device ID

5.47.1.4 handle

UINT16 handle

5.47.1.5 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.48 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](#)

5.48.1 Field Documentation

5.48.1.1 att_err

UINT8 att_err

0 is ok, others refer to LE_ATT_ERR_* in [ble_att_if.h](#)

5.48.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.48.1.3 devid

UINT16 devid

device ID

5.48.1.4 handle

UINT16 handle

characteristic descriptor handle

5.48.1.5 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.49 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](#)

5.49.1 Field Documentation

5.49.1.1 att_err

UINT8 att_err

0 is ok, others refer to LE_ATT_ERR_* in [ble_att_if.h](#)

5.49.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.49.1.3 devid

UINT16 devid

device ID

5.49.1.4 handle

UINT16 handle

include service start handle

5.49.1.5 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.50 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](#)

5.50.1 Field Documentation

5.50.1.1 att_err

UINT8 att_err

0 is ok, others refer to LE_ATT_ERR_* in [ble_att_if.h](#)

5.50.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.50.1.3 devid

UINT16 devid

device ID

5.50.1.4 handle

UINT16 handle

service start handle

5.50.1.5 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.51 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]

5.51.1 Field Documentation

5.51.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.51.1.2 devid

UINT16 devid

device ID

5.51.1.3 end_hdl

UINT16 end_hdl

end handle

5.51.1.4 format

UINT8 format

UUID type

5.51.1.5 handle

UINT16 handle

include servie handle

5.51.1.6 start_hdl

UINT16 start_hdl

start handle

5.51.1.7 uuid

UINT16 uuid[8]

UUID

5.52 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](#)

5.52.1 Field Documentation

5.52.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.52.1.2 devid

UINT16 devid

device ID

5.52.1.3 handle

UINT16 handle

attribute handle

5.52.1.4 len

UINT16 len

value length

5.52.1.5 val

UINT8* val

value

5.53 LE_GATT_MSG_NOTIFY_CFM_T Struct Reference

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

Data Fields

- UINT16 [conn_hdl](#)
- UINT16 [devid](#)
- UINT16 [handle](#)
- UINT16 [status](#)

5.53.1 Field Documentation

5.53.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.53.1.2 devid

UINT16 devid

device ID

5.53.1.3 handle

UINT16 handle

attribute handle

5.53.1.4 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.54 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](#)

5.54.1 Field Documentation

5.54.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.54.1.2 devid

UINT16 devid

device ID

5.54.1.3 handle

UINT16 handle

attribute handle

5.54.1.4 len

UINT16 len

value length

5.54.1.5 val

UINT8* val

value

5.55 LE_GATT_MSG_OPERATION_TIMEOUT_T Struct Reference

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

Data Fields

- UINT8 [att_op](#)
- UINT16 [conn_hdl](#)
- UINT16 [devid](#)

5.55.1 Field Documentation

5.55.1.1 att_op

UINT8 att_op

refer to LE_ATT_OP_* in [ble_att_if.h](#)

5.55.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.55.1.3 devid

UINT16 devid

device ID

5.56 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](#)

5.56.1 Field Documentation

5.56.1.1 att_err

UINT8 att_err

0 is ok, others refer to LE_ATT_ERR_* in [ble_att_if.h](#)

5.56.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.56.1.3 devid

UINT16 devid

device ID

5.56.1.4 handle

UINT16 handle

attribute handle

5.56.1.5 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.57 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](#)

5.57.1 Field Documentation

5.57.1.1 att_err

UINT8 att_err

0 is ok, others refer to LE_ATT_ERR_* in [ble_att_if.h](#)

5.57.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.57.1.3 devid

UINT16 devid

device ID

5.57.1.4 handle

UINT16 handle

characteristic value handle

5.57.1.5 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.58 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](#)

5.58.1 Field Documentation

5.58.1.1 att_err

UINT8 att_err

0 is ok, others refer to LE_ATT_ERR_* in [ble_att_if.h](#)

5.58.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.58.1.3 devid

UINT16 devid

device ID

5.58.1.4 handle

UINT16 handle

characteristic value handle

5.58.1.5 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.59 LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM_T Struct Reference

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

Data Fields

- UINT8 [att_err](#)
- UINT16 [conn_hdl](#)
- UINT16 [devid](#)
- UINT16 [handle](#)
- UINT16 [status](#)

5.59.1 Field Documentation

5.59.1.1 att_err

UINT8 att_err

0 is ok, others refer to LE_ATT_ERR_* in [ble_att_if.h](#)

5.59.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.59.1.3 devid

UINT16 devid

device ID

5.59.1.4 handle

UINT16 handle

characteristic value handle

5.59.1.5 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.60 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](#)

5.60.1 Field Documentation

5.60.1.1 att_err

UINT8 att_err

0 is ok, others refer to LE_ATT_ERR_* in [ble_att_if.h](#)

5.60.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.60.1.3 devid

UINT16 devid

device ID

5.60.1.4 err_hdl

UINT16 err_hdl

TBD

5.60.1.5 len

UINT16 len

value length

5.60.1.6 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.60.1.7 val

UINT8* val

value

5.61 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]

5.61.1 Field Documentation

5.61.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.61.1.2 devid

UINT16 devid

device ID

5.61.1.3 end_hdl

UINT16 end_hdl

end handle

5.61.1.4 format

UINT8 format

UUID type

5.61.1.5 start_hdl

UINT16 start_hdl

start handle

5.61.1.6 uuid

UINT16 uuid[8]

UUID

5.62 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](#)

5.62.1 Field Documentation

5.62.1.1 conn_hdl

UINT16 [conn_hdl](#)

connection handle

5.62.1.2 devid

UINT16 [devid](#)

device ID

5.62.1.3 handle

UINT16 [handle](#)

attribute handle

5.62.1.4 status

UINT16 [status](#)

refer to LE_ERR_STATE in [ble_err.h](#)

5.63 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](#)

5.63.1 Field Documentation

5.63.1.1 att_err

UINT8 att_err

0 is ok, others refer to LE_ATT_ERR_* in [ble_att_if.h](#)

5.63.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.63.1.3 devid

UINT16 devid

device ID

5.63.1.4 handle

UINT16 handle

characteristic value handle

5.63.1.5 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.64 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](#)

5.64.1 Field Documentation

5.64.1.1 att_err

UINT8 att_err

0 is ok, others refer to LE_ATT_ERR_* in [ble_att_if.h](#)

5.64.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.64.1.3 devid

UINT16 devid

device ID

5.64.1.4 handle

UINT16 handle

attribute handle

5.64.1.5 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.65 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](#)

5.65.1 Field Documentation

5.65.1.1 att_err

UINT8 att_err

0 is ok, others refer to LE_ATT_ERR_* in [ble_att_if.h](#)

5.65.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.65.1.3 devid

UINT16 devid

device ID

5.65.1.4 handle

UINT16 handle

characteristic value handle

5.65.1.5 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.66 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](#)

5.66.1 Field Documentation

5.66.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.66.1.2 devid

UINT16 devid

device ID

5.66.1.3 handle

UINT16 handle

attribute handle

5.66.1.4 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.67 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](#)

5.67.1 Field Documentation

5.67.1.1 endHdl

UINT16 endHdl

end handle

5.67.1.2 pAttr

LE_GATT_ATTR_T* pAttr

pointer attribute table

5.67.1.3 startHdl

UINT16 startHdl

start handle

5.67.1.4 svc_id

UINT16 svc_id

service ID

5.68 LE_SMP_MSG_ENCRYPTION_CHANGE_IND_T Struct Reference

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

Data Fields

- UINT16 [conn_hdl](#)
- BOOL [enable](#)

5.68.1 Field Documentation

5.68.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.68.1.2 enable

BOOL enable

enable or disable

5.69 LE_SMP_MSG_ENCRYPTION_REFRESH_IND_T Struct Reference

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

Data Fields

- UINT16 [conn_hdl](#)
- UINT16 [status](#)

5.69.1 Field Documentation

5.69.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.69.1.2 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.70 LE_SMP_MSG_OOB_DATA_REQUEST_IND_T Struct Reference

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

Data Fields

- UINT16 [conn_hdl](#)

5.70.1 Field Documentation

5.70.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.71 LE_SMP_MSG_PAIRING_ACTION_IND_T Struct Reference

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

Data Fields

- UINT8 [action](#)
- UINT16 [conn_hdl](#)
- BOOL [lost_bond](#)
- UINT8 [sc](#)

5.71.1 Field Documentation

5.71.1.1 action

UINT8 action

refer to LE_SM_IO_CAP_* in [ble_smp_if.h](#)

5.71.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.71.1.3 lost_bond

BOOL lost_bond

remote lost bond

5.71.1.4 sc

UINT8 sc

secure connection

5.72 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](#)

5.72.1 Field Documentation

5.72.1.1 [authenticated](#)

`UINT8` [authenticated](#)

[authenticated](#)

5.72.1.2 [bonded](#)

`UINT8` [bonded](#)

[bonded](#)

5.72.1.3 [conn_hdl](#)

`UINT16` [conn_hdl](#)

connection handle

5.72.1.4 [peer_id_addr](#)

[LE_BT_ADDR_T](#) [peer_id_addr](#)

peer device address

5.72.1.5 [sc](#)

`UINT8` [sc](#)

secure connection

5.72.1.6 status

UINT16 status

refer to LE_ERR_STATE in [ble_err.h](#)

5.73 LE_SMP_MSG_PASSKEY_DISPLAY_IND_T Struct Reference

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

Data Fields

- UINT16 [conn_hdl](#)
- UINT32 [passkey](#)

5.73.1 Field Documentation

5.73.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.73.1.2 passkey

UINT32 passkey

passkey

5.74 LE_SMP_MSG_PASSKEY_INPUT_IND_T Struct Reference

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

Data Fields

- UINT16 [conn_hdl](#)

5.74.1 Field Documentation

5.74.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.75 LE_SMP_MSG_SC_OOB_DATA_REQUEST_IND_T Struct Reference

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

Data Fields

- UINT16 [conn_hdl](#)

5.75.1 Field Documentation

5.75.1.1 conn_hdl

UINT16 conn_hdl

connection handle

5.76 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](#)

5.76.1 Field Documentation

5.76.1.1 bondable

UINT8 bondable

bonding

5.76.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.76.1.3 keypress

UINT8 keypress

keypress status

5.76.1.4 mitm

UINT8 mitm

MITM

5.76.1.5 sc

UINT8 sc

secure connection

5.77 LE_SMP_MSG_USER_CONFIRM_IND_T Struct Reference

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

Data Fields

- UINT32 [confirm_num](#)
- UINT16 [conn_hdl](#)

5.77.1 Field Documentation

5.77.1.1 confirm_num

UINT32 confirm_num

confirm number

5.77.1.2 conn_hdl

UINT16 conn_hdl

connection handle

5.78 LE_SMP_SC_OOB_DATA_T Struct Reference

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

Data Fields

- UINT8 [confirm](#) [16]
- UINT8 [rand](#) [16]

5.78.1 Field Documentation

5.78.1.1 confirm

UINT8 confirm[16]

confirm data

5.78.1.2 rand

UINT8 rand[16]

random data

5.79 LE_SYS_MSG_BUF_OVERFLOW_T Struct Reference

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

Data Fields

- UINT16 [conn_hdl](#)

5.79.1 Field Documentation

5.79.1.1 conn_hdl

UINT16 [conn_hdl](#)

connection handle

5.80 mw_blewifi_cbs_store_t Struct Reference

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

Data Fields

- uint8_t [manufacture_name](#) [STA_INFO_MAX_MANUF_NAME_SIZE]

5.80.1 Field Documentation

5.80.1.1 manufacture_name

uint8_t [manufacture_name](#) [STA_INFO_MAX_MANUF_NAME_SIZE]

5.81 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]

5.81.1 Field Documentation

5.81.1.1 [ap_channel](#)

u8 [ap_channel](#)

5.81.1.2 [beacon_interval](#)

u16 [beacon_interval](#)

5.81.1.3 [bssid](#)

u8 [bssid](#)[MAC_ADDR_LEN]

5.81.1.4 [capabilities](#)

u16 [capabilities](#)

5.81.1.5 dtim_prod

u8 dtim_prod

5.81.1.6 fast_connect

u8 fast_connect

5.81.1.7 free_ocpy

bool free_ocpy

5.81.1.8 hid_ssid

s8 hid_ssid[IEEE80211_MAX_SSID_LEN+1]

5.81.1.9 hid_ssid_len

u8 hid_ssid_len

5.81.1.10 latest_beacon_rx_time

u64 latest_beacon_rx_time

5.81.1.11 passphrase

s8 passphrase[64]

5.81.1.12 psk

u8 psk[32]

5.81.1.13 rsn_ie

```
u8 rsn_ie[256]
```

5.81.1.14 rssi

```
s8 rssi
```

5.81.1.15 ssid

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

5.81.1.16 ssid_len

```
u8 ssid_len
```

5.81.1.17 supported_rates

```
u8 supported_rates[IEEE80211_MAX_SUPPORTED_RATES]
```

5.81.1.18 wpa_data

```
wpa_ie_data_t wpa_data
```

5.81.1.19 wpa_ie

```
u8 wpa_ie[257]
```

5.82 mw_wifi_sta_info_t Struct Reference

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

Data Fields

- uint8_t [au8Dot11MACAddress](#) [MAC_ADDR_LEN]
- uint8_t [u8SkipDtimPeriods](#)

5.82.1 Field Documentation

5.82.1.1 au8Dot11MACAddress

```
uint8_t au8Dot11MACAddress [MAC_ADDR_LEN]
```

5.82.1.2 u8SkipDtimPeriods

```
uint8_t u8SkipDtimPeriods
```

5.83 MwFimAutoConnectCFG_t Struct Reference

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

Data Fields

- bool [flag](#)
- s8 [front](#)
- u8 [max_save_num](#)
- s8 [rear](#)
- u8 [targetIdx](#)

5.83.1 Field Documentation

5.83.1.1 flag

```
bool flag
```


5.83.1.2 front

s8 front

5.83.1.3 max_save_num

u8 max_save_num

5.83.1.4 rear

s8 rear

5.83.1.5 targetIdx

u8 targetIdx

5.84 rx_eapol_data Struct Reference

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

Data Fields

- u8 [frame_buffer](#) [384]
- unsigned int [frame_length](#)

5.84.1 Field Documentation

5.84.1.1 frame_buffer

u8 frame_buffer[384]

5.84.1.2 frame_length

```
unsigned int frame_length
```

5.85 S_WIFI_MLME_SCAN_CFG Struct Reference

```
#include <controller_wifi_com_patch.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](#)

5.85.1 Detailed Description

The parameter of MLME_CMD_SCAN

5.85.2 Field Documentation

5.85.2.1 ptScanReport

```
scan\_report\_t* ptScanReport
```

The scan report which filled by MSQ, report to APS

5.85.2.2 tScanType

```
E\_WIFI\_MLME\_SCAN\_TYPE tScanType
```

scan type. active, passive, or mix mode

5.85.2.3 u32ActiveScanDur

```
uint32\_t u32ActiveScanDur
```

Scan duration per scan counter in channel. units: millisecond

5.85.2.4 u32PassiveScanDur

```
uint32_t u32PassiveScanDur
```

Scan duration per channel. units: millisecond

5.85.2.5 u8aBssid

```
uint8_t u8aBssid[MAC_ADDR_LEN]
```

Not supported yet. MAC address of AP

5.85.2.6 u8aSsid

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

Not supported yet. SSID of AP

5.85.2.7 u8Channel

```
uint8_t u8Channel
```

Only specific channel or scan all channels

5.85.2.8 u8MaxScanApNum

```
uint8_t u8MaxScanApNum
```

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

5.85.2.9 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

5.86 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]

5.86.1 Field Documentation

5.86.1.1 ap_channel

`uint8_t ap_channel`

5.86.1.2 beacon_interval

`uint16_t beacon_interval`

5.86.1.3 bssid

`uint8_t bssid`[MAC_ADDR_LEN]

5.86.1.4 capabilities

`uint16_t capabilities`

5.86.1.5 dtim_prod

```
uint8_t dtim_prod
```

5.86.1.6 free_ocpy

```
unsigned char free_ocpy
```

5.86.1.7 latest_beacon_rx_time

```
uint64_t latest_beacon_rx_time
```

5.86.1.8 rsn_ie

```
u8 rsn_ie[256]
```

5.86.1.9 rssi

```
int8_t rssi
```

5.86.1.10 ssid

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

5.86.1.11 ssid_len

```
uint8_t ssid_len
```

5.86.1.12 supported_rates

```
uint8_t supported_rates[IEEE80211_MAX_SUPPORTED_RATES]
```

5.86.1.13 wpa_data

`wpa_ie_data_t` wpa_data

5.86.1.14 wpa_ie

`u8` wpa_ie[257]

5.87 scan_report_t Struct Reference

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

Data Fields

- `scan_info_t` * pScanInfo
- `u32` uScanApNum

5.87.1 Field Documentation

5.87.1.1 pScanInfo

`scan_info_t`* pScanInfo

5.87.1.2 uScanApNum

`u32` uScanApNum

5.88 T_RfCmd Struct Reference

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

Data Fields

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

5.88.1 Field Documentation

5.88.1.1 iArgc

```
int iArgc
```

5.88.1.2 saArgv

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

5.88.1.3 u32Type

```
uint32_t u32Type
```

5.89 T_RfEvt Struct Reference

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

Data Fields

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

5.89.1 Field Documentation

5.89.1.1 pParam

`void* pParam`

5.89.1.2 u16RfMode

`uint16_t u16RfMode`

5.89.1.3 u16RxCnt

`uint16_t u16RxCnt`

5.89.1.4 u16RxCrcOkCnt

`uint16_t u16RxCrcOkCnt`

5.89.1.5 u32Freq

`uint32_t u32Freq`

5.89.1.6 u32Mode

`uint32_t u32Mode`

5.89.1.7 u32RfChannel

`uint32_t u32RfChannel`

5.89.1.8 u32Type

`uint32_t u32Type`

5.89.1.9 u8Freq

```
uint8_t u8Freq
```

5.89.1.10 u8IpcEnable

```
uint8_t u8IpcEnable
```

5.89.1.11 u8Len

```
uint8_t u8Len
```

5.89.1.12 u8Pkt

```
uint8_t u8Pkt
```

5.89.1.13 u8Reserved

```
uint8_t u8Reserved
```

5.89.1.14 u8Status

```
uint8_t u8Status
```

5.89.1.15 u8Unicast

```
uint8_t u8Unicast
```

5.90 wifi_active_scan_time_t Struct Reference

Range of active scan times per channel.

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

Data Fields

- uint32_t [max](#)
- uint32_t [min](#)

5.90.1 Detailed Description

Range of active scan times per channel.

5.90.2 Field Documentation

5.90.2.1 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.

5.90.2.2 min

```
uint32_t min
```

minimum active scan time per channel, units: millisecond

5.91 wifi_ap_config_t Struct Reference

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

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

Data Fields

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

5.91.1 Detailed Description

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

5.91.2 Field Documentation

5.91.2.1 auth_mode

wifi_auth_mode_t auth_mode

The authentication mode.

5.91.2.2 beacon_interval

uint16_t beacon_interval

Beacon interval, 100 ~ 60000 ms, default 100 ms

5.91.2.3 channel

uint8_t channel

The channel of Soft-AP.

5.91.2.4 encrypt_type

wifi_cipher_type_t encrypt_type

The encryption mode.

5.91.2.5 max_connection

uint8_t max_connection

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

5.91.2.6 password

uint8_t password[WIFI_LENGTH_PASSPHRASE]

The password of the Soft-AP.

5.91.2.7 password_length

```
uint8_t password_length
```

The length of the password.

5.91.2.8 ssid

```
uint8_t ssid[WIFI_MAX_LENGTH_OF_SSID]
```

The SSID of the Soft-AP.

5.91.2.9 ssid_hidden

```
uint8_t ssid_hidden
```

Broadcast SSID or not, default 0, broadcast the SSID

5.91.2.10 ssid_length

```
uint8_t ssid_length
```

The length of the SSID.

5.92 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](#)]
- unsigned long long [latest_beacon_rx_time](#)
- char [passphrase](#) [64]
- uint8_t [psk](#) [32]
- uint8_t [rsn_ie](#) [100]
- char [rssi](#)
- char [ssid](#) [[WIFI_MAX_LENGTH_OF_SSID](#)]
- uint8_t [supported_rates](#) [[WIFI_MAX_SUPPORTED_RATES](#)]
- [wifi_wpa_ie_data_t](#) [wpa_data](#)
- uint8_t [wpa_ie](#) [100]

5.92.1 Detailed Description

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

5.92.2 Field Documentation

5.92.2.1 ap_channel

uint8_t ap_channel

5.92.2.2 beacon_interval

uint16_t beacon_interval

5.92.2.3 bssid

uint8_t bssid[WIFI_MAC_ADDRESS_LENGTH]

5.92.2.4 capabilities

uint16_t capabilities

5.92.2.5 dtim_prod

uint8_t dtim_prod

5.92.2.6 fast_connect

uint8_t fast_connect

5.92.2.7 hid_ssid

```
char hid_ssid[WIFI_MAX_LENGTH_OF_SSID]
```

5.92.2.8 latest_beacon_rx_time

```
unsigned long long latest_beacon_rx_time
```

5.92.2.9 passphrase

```
char passphrase[64]
```

5.92.2.10 psk

```
uint8_t psk[32]
```

5.92.2.11 rsn_ie

```
uint8_t rsn_ie[100]
```

5.92.2.12 rssi

```
char rssi
```

5.92.2.13 ssid

```
char ssid[WIFI_MAX_LENGTH_OF_SSID]
```

5.92.2.14 supported_rates

```
uint8_t supported_rates[WIFI_MAX_SUPPORTED_RATES]
```

5.92.2.15 wpa_data

wifi_wpa_ie_data_t wpa_data

5.92.2.16 wpa_ie

uint8_t wpa_ie[100]

5.93 wifi_cmd_t Struct Reference

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

Data Fields

- u32 [arg1](#):8
- u32 [cmd_type](#):8
- void * [prvData](#)
- u32 [reserved](#):16

5.93.1 Field Documentation

5.93.1.1 arg1

u32 arg1

5.93.1.2 cmd_type

u32 cmd_type

5.93.1.3 prvData

void* prvData

5.93.1.4 reserved

u32 reserved

5.94 wifi_config_t Union Reference

Wi-Fi configuration for initialization.

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

Data Fields

- [wifi_ap_config_t ap_config](#)
- [wifi_sta_config_t sta_config](#)

5.94.1 Detailed Description

Wi-Fi configuration for initialization.

5.94.2 Field Documentation

5.94.2.1 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`.

5.94.2.2 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`.

5.95 wifi_event_info_t Union Reference

```
wifi_event_info_t
```

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


Data Fields

- [wifi_event_sta_connected_t](#) connected
- [wifi_event_sta_disconnected_t](#) disconnected
- [wifi_event_sta_got_ip_t](#) got_ip
- [wifi_event_sta_scan_done_t](#) scan_done

5.95.1 Detailed Description

[wifi_event_info_t](#)

5.95.2 Field Documentation

5.95.2.1 connected

[wifi_event_sta_connected_t](#) connected

station connected to AP

5.95.2.2 disconnected

[wifi_event_sta_disconnected_t](#) disconnected

station disconnected to AP

5.95.2.3 got_ip

[wifi_event_sta_got_ip_t](#) got_ip

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

5.95.2.4 scan_done

[wifi_event_sta_scan_done_t](#) scan_done

station scan (APs) done

5.96 wifi_event_sta_connected_t Struct Reference

[wifi_event_sta_connected_t](#)

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

Data Fields

- [wifi_auth_mode_t authmode](#)
- `uint8_t bssid` [6]
- `uint8_t channel`
- `uint8_t ssid` [32]
- `uint8_t ssid_len`

5.96.1 Detailed Description

[wifi_event_sta_connected_t](#)

5.96.2 Field Documentation

5.96.2.1 authmode

[wifi_auth_mode_t](#) authmode

5.96.2.2 bssid

`uint8_t bssid`[6]

BSSID of connected AP

5.96.2.3 channel

`uint8_t channel`

channel of connected AP

5.96.2.4 ssid

`uint8_t ssid`[32]

SSID of connected AP

5.96.2.5 ssid_len

`uint8_t ssid_len`

SSID length of connected AP

5.97 wifi_event_sta_disconnected_t Struct Reference

[wifi_event_sta_disconnected_t](#)

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

Data Fields

- `uint8_t` [bssid](#) [6]
- `uint8_t` [reason](#)
- `uint8_t` [ssid](#) [32]
- `uint8_t` [ssid_len](#)

5.97.1 Detailed Description

[wifi_event_sta_disconnected_t](#)

5.97.2 Field Documentation

5.97.2.1 bssid

```
uint8_t bssid[6]
```

BSSID of disconnected AP

5.97.2.2 reason

```
uint8_t reason
```

reason of disconnection

5.97.2.3 ssid

```
uint8_t ssid[32]
```

SSID of disconnected AP

5.97.2.4 ssid_len

```
uint8_t ssid_len
```

SSID length of disconnected AP

5.98 `wifi_event_sta_got_ip_t` Struct Reference

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

Data Fields

- `bool ip_changed`

5.98.1 Field Documentation

5.98.1.1 `ip_changed`

```
bool ip_changed
```

5.99 `wifi_event_sta_scan_done_t` Struct Reference

[wifi_event_sta_scan_done_t](#)

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

Data Fields

- `uint8_t number`
- `uint8_t scan_id`
- `uint32_t status`

5.99.1 Detailed Description

[wifi_event_sta_scan_done_t](#)

5.99.2 Field Documentation

5.99.2.1 `number`

```
uint8_t number
```

5.99.2.2 scan_id

uint8_t scan_id

5.99.2.3 status

uint32_t status

status of scanning APs

5.100 wifi_evt_t Struct Reference

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

Data Fields

- uint32_t [evt_type](#)
- void * [prvData](#)

5.100.1 Field Documentation

5.100.1.1 evt_type

uint32_t evt_type

5.100.1.2 prvData

void* prvData

5.101 wifi_fast_scan_threshold_t Struct Reference

Structure describing parameters for a Wi-Fi fast scan.

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

Data Fields

- [wifi_auth_mode_t authmode](#)
- [int8_t rssi](#)

5.101.1 Detailed Description

Structure describing parameters for a Wi-Fi fast scan.

5.101.2 Field Documentation

5.101.2.1 authmode

```
wifi_auth_mode_t authmode
```

The weakest authmode to accept in the fast scan mode

5.101.2.2 rssi

```
int8_t rssi
```

The minimum rssi to accept in the fast scan mode

5.102 wifi_init_config_t Struct Reference

WiFi stack configuration parameters.

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

Data Fields

- [wifi_event_notify_cb_t event_handler](#)
- [int magic](#)

5.102.1 Detailed Description

WiFi stack configuration parameters.

5.102.2 Field Documentation

5.102.2.1 event_handler

wifi_event_notify_cb_t event_handler

WiFi event handler

5.102.2.2 magic

int magic

WiFi init magic number, it should be the last field

5.103 wifi_scan_config_t Struct Reference

Parameters for an SSID scan.

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

Data Fields

- uint8_t * [bssid](#)
- uint8_t [channel](#)
- [wifi_scan_time_t](#) scan_time
- [wifi_scan_type_t](#) scan_type
- bool [show_hidden](#)
- uint8_t * [ssid](#)

5.103.1 Detailed Description

Parameters for an SSID scan.

5.103.2 Field Documentation

5.103.2.1 bssid

uint8_t* bssid

MAC address of AP

5.103.2.2 channel

uint8_t channel

channel, scan the specific channel

5.103.2.3 scan_time

`wifi_scan_time_t scan_time`

scan time per channel

5.103.2.4 scan_type

`wifi_scan_type_t scan_type`

scan type, active or passive

5.103.2.5 show_hidden

`bool show_hidden`

enable to scan AP whose SSID is hidden

5.103.2.6 ssid

`uint8_t* ssid`

SSID of AP

5.104 wifi_scan_info_t Struct Reference

This structure defines the information of scanned APs.

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

Data Fields

- `wifi_auth_mode_t auth_mode`
- `uint16_t beacon_interval`
- `uint8_t bssid [WIFI_MAC_ADDRESS_LENGTH]`
- `uint16_t capability_info`
- `uint8_t channel`
- `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`

5.104.1 Detailed Description

This structure defines the information of scanned APs.

5.104.2 Field Documentation

5.104.2.1 auth_mode

[wifi_auth_mode_t](#) auth_mode

Please refer to the definition of [wifi_auth_mode_t](#).

5.104.2.2 beacon_interval

uint16_t beacon_interval

Indicates the beacon interval.

5.104.2.3 bssid

uint8_t bssid[WIFI_MAC_ADDRESS_LENGTH]

AP's MAC address.

5.104.2.4 capability_info

uint16_t capability_info

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

5.104.2.5 channel

uint8_t channel

The channel used.

5.104.2.6 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.

5.104.2.7 group_cipher

[wifi_cipher_type_t](#) group_cipher

group cipher of AP

5.104.2.8 pairwise_cipher

wifi_cipher_type_t pairwise_cipher

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

5.104.2.9 rssi

int rssi

Records the RSSI value when probe response is received.

5.104.2.10 ssid

uint8_t ssid[WIFI_MAX_LENGTH_OF_SSID]

Stores the predefined SSID.

5.104.2.11 ssid_length

uint8_t ssid_length

Length of the SSID.

5.105 wifi_scan_list_t Struct Reference

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

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

Data Fields

- [wifi_scan_info_t ap_record](#) [WIFI_MAX_SCAN_AP_NUM]
- int [num](#)

5.105.1 Detailed Description

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

5.105.2 Field Documentation

5.105.2.1 ap_record

```
wifi_scan_info_t ap_record[WIFI_MAX_SCAN_AP_NUM]
```

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

5.105.2.2 num

```
int num
```

number of AP in the list

5.106 wifi_scan_time_t Union Reference

Aggregate of active & passive scan time per channel.

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

Data Fields

- [wifi_active_scan_time_t](#) `active`
- `uint32_t` `passive`

5.106.1 Detailed Description

Aggregate of active & passive scan time per channel.

5.106.2 Field Documentation

5.106.2.1 active

```
wifi_active_scan_time_t active
```

active scan time per channel, units: millisecond.

5.106.2.2 passive

```
uint32_t passive
```

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

5.107 wifi_sta_config_t Struct Reference

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

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

Data Fields

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

5.107.1 Detailed Description

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

5.107.2 Field Documentation

5.107.2.1 bssid

```
uint8_t bssid[WIFI_MAC_ADDRESS_LENGTH]
```

The MAC address of the target AP.

5.107.2.2 bssid_present

```
uint8_t bssid_present
```

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

5.107.2.3 password

```
uint8_t password[WIFI_LENGTH_PASSPHRASE]
```

The password of the target AP.

5.107.2.4 password_length

uint8_t password_length

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

5.107.2.5 scan_method

wifi_scan_method_t scan_method

do all channel scan or fast scan

5.107.2.6 sort_method

wifi_sort_method_t sort_method

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

5.107.2.7 ssid

uint8_t ssid[WIFI_MAX_LENGTH_OF_SSID]

The SSID of the target AP.

5.107.2.8 ssid_length

uint8_t ssid_length

The length of the SSID.

5.107.2.9 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.

5.108 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](#)

5.108.1 Detailed Description

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

5.108.2 Field Documentation

5.108.2.1 capabilities

```
int capabilities
```

5.108.2.2 group_cipher

```
int group_cipher
```

5.108.2.3 key_mgmt

```
int key_mgmt
```

5.108.2.4 mgmt_group_cipher

```
int mgmt_group_cipher
```

5.108.2.5 num_pmkid

uint32_t num_pmkid

5.108.2.6 pairwise_cipher

int pairwise_cipher

5.108.2.7 pmkid

const uint8_t* pmkid

5.108.2.8 proto

int proto

Index

- [_wpa_ie_data, 147](#)
 - [capabilities, 147](#)
 - [group_cipher, 147](#)
 - [key_mgmt, 147](#)
 - [mgmt_group_cipher, 148](#)
 - [num_pmkid, 148](#)
 - [pairwise_cipher, 148](#)
 - [pmkid, 148](#)
 - [proto, 148](#)
- [action](#)
 - [LE_SMP_MSG_PAIRING_ACTION_IND_T, 216](#)
- [active](#)
 - [wifi_scan_time_t, 253](#)
- [addr](#)
 - [LE_BT_ADDR_T, 157](#)
 - [LE_CM_MSG_ADVERTISE_REPORT_IND_T, 160](#)
- [addr_type](#)
 - [LE_CM_MSG_ADVERTISE_REPORT_IND_T, 160](#)
- [ap_channel](#)
 - [auto_conn_info_t, 151](#)
 - [mw_wifi_auto_connect_ap_info_t, 223](#)
 - [scan_info_t, 230](#)
 - [wifi_auto_connect_info_t, 239](#)
- [ap_config](#)
 - [wifi_config_t, 242](#)
- [ap_record](#)
 - [wifi_scan_list_t, 252](#)
- [arg1](#)
 - [wifi_cmd_t, 241](#)
- [asso_data, 148](#)
 - [eap_workaround, 149](#)
 - [eapol_flags, 149](#)
 - [group_cipher, 149](#)
 - [key_mgmt, 149](#)
 - [leap, 149](#)
 - [mgmt_group_cipher, 149](#)
 - [non_leap, 150](#)
 - [pairwise_cipher, 150](#)
 - [passphrase, 150](#)
 - [proto, 150](#)
 - [psk, 150](#)
 - [psk_set, 150](#)
- [att_err](#)
 - [LE_GATT_MSG_CHARACTERISTIC_VAL_IND_T, 188](#)
 - [LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM_T, 191](#)
- [LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM_T, 192](#)
- [LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_CFM_T, 193](#)
- [LE_GATT_MSG_FIND_CHARACTERISTIC_CFM_T, 194](#)
- [LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM_T, 195](#)
- [LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM_T, 196](#)
- [LE_GATT_MSG_PREPARE_WRITE_RELIABLE_CFM_T, 202](#)
- [LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_CFM_T, 203](#)
- [LE_GATT_MSG_READ_CHARACTERISTIC_VALUE_CFM_T, 204](#)
- [LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM_T, 205](#)
- [LE_GATT_MSG_READ_MULTIPLE_CHAR_VALUES_CFM_T, 206](#)
- [LE_GATT_MSG_WRITE_CHAR_VAL_RELIABLE_CFM_T, 210](#)
- [LE_GATT_MSG_WRITE_CHAR_VALUE_CFM_T, 211](#)
- [LE_GATT_MSG_WRITE_LONG_CHAR_VALUE_CFM_T, 212](#)
- [att_op](#)
 - [LE_GATT_MSG_OPERATION_TIMEOUT_T, 201](#)
- [au8Dot11MACAddress](#)
 - [mw_wifi_sta_info_t, 226](#)
- [auth_mode](#)
 - [wifi_ap_config_t, 237](#)
 - [wifi_scan_info_t, 251](#)
- [authenticated](#)
 - [LE_SMP_MSG_PAIRING_COMPLETE_IND_T, 217](#)
- [authmode](#)
 - [wifi_event_sta_connected_t, 244](#)
 - [wifi_fast_scan_threshold_t, 248](#)
- [auto_conn_info_t, 150](#)
 - [ap_channel, 151](#)
 - [beacon_interval, 151](#)
 - [bssid, 151](#)
 - [capabilities, 151](#)
 - [dtim_prod, 151](#)
 - [fast_connect, 152](#)
 - [free_ocpy, 152](#)
 - [hid_ssid, 152](#)
 - [hid_ssid_len, 152](#)

- latest_beacon_rx_time, 152
- passphrase, 152
- psk, 152
- rsn_ie, 152
- rsni, 153
- ssid, 153
- ssid_len, 153
- supported_rates, 153
- wpa_data, 153
- wpa_ie, 153
- auto_connect_cfg_t, 153
 - flag, 154
 - front, 154
 - max_save_num, 154
 - pFCInfo, 154
 - rear, 154
 - retryCount, 154
 - targetIdx, 155
 - uFCAppNum, 155
- BLE ALL APIs, 9
 - LeSmpGetBondIdFromAddr, 9
- BLE CM APIs, 10
 - LE_CM_MSG_ADD_TO_RESOLVING_LIST_C←
FM_T, 11
 - LE_CM_MSG_ADD_TO_WHITE_LIST_CFM_T, 11
 - LE_CM_MSG_CANCEL_CONNECTION_CFM_T, 11
 - LE_CM_MSG_CLEAR_RESOLVING_LIST_C←
M_T, 12
 - LE_CM_MSG_CLEAR_WHITE_LIST_CFM_T, 12
 - LE_CM_MSG_CREATE_CONNECTION_CFM_T, 12
 - LE_CM_MSG_ENTER_ADVERTISING_CFM_T, 12
 - LE_CM_MSG_ENTER_SCANNING_CFM_T, 12
 - LE_CM_MSG_EXIT_ADVERTISING_CFM_T, 12
 - LE_CM_MSG_EXIT_SCANNING_CFM_T, 12
 - LE_CM_MSG_PHY_UPDATE_COMPLETE_IN←
D_T, 12
 - LE_CM_MSG_REMOVE_FROM_RESOLVING←
LIST_CFM_T, 13
 - LE_CM_MSG_REMOVE_FROM_WHITE_LIST←
_CFM_T, 13
 - LE_CM_MSG_SET_ADVERTISING_DATA_C←
M_T, 13
 - LE_CM_MSG_SET_ADVERTISING_PARAMS←
CFM_T, 13
 - LE_CM_MSG_SET_CHANNEL_MAP_CFM_T, 13
 - LE_CM_MSG_SET_DEFAULT_PHY_CFM_T, 13
 - LE_CM_MSG_SET_RANDOM_ADDRESS_C←
M_T, 13
 - LE_CM_MSG_SET_RPA_TIMEOUT_CFM_T, 13
 - LE_CM_MSG_SET_SCAN_PARAMS_CFM_T, 14
 - LE_CM_MSG_SET_SCAN_RSP_DATA_CFM_T, 14
 - LeCmInit, 15
- BLE GAP APIs, 17
 - GAP_ADTYPE_128BIT_COMPLETE, 19
 - GAP_ADTYPE_128BIT_MORE, 19
 - GAP_ADTYPE_16BIT_COMPLETE, 19
 - GAP_ADTYPE_16BIT_MORE, 20
 - GAP_ADTYPE_32BIT_COMPLETE, 20
 - GAP_ADTYPE_32BIT_MORE, 20
 - GAP_ADTYPE_3D_INFO_DATA, 20
 - GAP_ADTYPE_ADV_INTERVAL, 20
 - GAP_ADTYPE_APPEARANCE, 20
 - GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPO←
RTED, 20
 - GAP_ADTYPE_FLAGS_GENERAL, 21
 - GAP_ADTYPE_FLAGS_LIMITED, 21
 - GAP_ADTYPE_FLAGS, 20
 - GAP_ADTYPE_LE_BD_ADDR, 21
 - GAP_ADTYPE_LE_ROLE, 21
 - GAP_ADTYPE_LOCAL_NAME_COMPLETE, 21
 - GAP_ADTYPE_LOCAL_NAME_SHORT, 21
 - GAP_ADTYPE_MANUFACTURER_SPECIFIC, 21
 - GAP_ADTYPE_OOB_CLASS_OF_DEVICE, 21
 - GAP_ADTYPE_OOB_SIMPLE_PAIRING_HAS←
HC, 22
 - GAP_ADTYPE_OOB_SIMPLE_PAIRING_RAN←
DR, 22
 - GAP_ADTYPE_POWER_LEVEL, 22
 - GAP_ADTYPE_PUBLIC_TARGET_ADDR, 22
 - GAP_ADTYPE_RANDOM_TARGET_ADDR, 22
 - GAP_ADTYPE_SERVICE_DATA_128BIT, 22
 - GAP_ADTYPE_SERVICE_DATA_32BIT, 22
 - GAP_ADTYPE_SERVICE_DATA, 22
 - GAP_ADTYPE_SERVICES_LIST_128BIT, 23
 - GAP_ADTYPE_SERVICES_LIST_16BIT, 23
 - GAP_ADTYPE_SIGNED_DATA, 23
 - GAP_ADTYPE_SIMPLE_PAIRING_HASHC_256, 23
 - GAP_ADTYPE_SIMPLE_PAIRING_RANDR_256, 23
 - GAP_ADTYPE_SLAVE_CONN_INTERVAL_RA←
NGE, 23
 - GAP_ADTYPE_SM_OOB_FLAG, 23
 - GAP_ADTYPE_SM_TK, 23
 - GAP_PUBLIC_ADDR, 24
 - GAP_RAND_ADDR_NRPA, 24
 - GAP_RAND_ADDR_RPA, 24
 - GAP_RAND_ADDR_STATIC, 24
 - GAP_SCAN_TYPE_ACTIVE, 24
 - GAP_SCAN_TYPE_PASSIVE, 24
 - GAP_TX_PWR_CURR_VAL, 24
 - GAP_TX_PWR_MAX_VAL, 24
 - GAPBOND_IO_CAP_DISPLAY_ONLY, 25
 - GAPBOND_IO_CAP_DISPLAY_YES_NO, 25
 - GAPBOND_IO_CAP_KEYBOARD_DISPLAY, 25
 - GAPBOND_IO_CAP_KEYBOARD_ONLY, 25
 - GAPBOND_IO_CAP_NO_INPUT_NO_OUTPUT, 25
 - GAPBOND_PAIRING_MODE_INITIATE, 25
 - GAPBOND_PAIRING_MODE_NO_PAIRING, 25

- GAPBOND_PAIRING_MODE_WAIT_FOR_REQ, 25
- LE_GAP_ADV_MAX_SIZE, 26
- LeGapAddToResolvingList, 26
- LeGapAddToWhiteList, 26
- LeGapAdvertisingEnable, 27
- LeGapCentralConnectReq, 27
- LeGapCentralSetDataChannel, 27
- LeGapClearResolvingList, 29
- LeGapClearWhiteList, 29
- LeGapConnParaRequestRsp, 29
- LeGapConnUpdateRequest, 30
- LeGapConnUpdateResponse, 30
- LeGapConnectCancelReq, 29
- LeGapDisconnectReq, 31
- LeGapGenRandAddr, 31
- LeGapGetBtAddr, 31
- LeGapReadAdvChannelTxPower, 31
- LeGapReadChannelMap, 32
- LeGapReadPhy, 32
- LeGapReadResolvingListSize, 32
- LeGapReadRssi, 32
- LeGapReadTxPower, 33
- LeGapReadWhiteListSize, 33
- LeGapRemoveFromWhiteList, 33
- LeGapScanningReq, 34
- LeGapSetAdvData, 34
- LeGapSetAdvParameter, 35
- LeGapSetConnParameter, 35
- LeGapSetDataChannelPduLen, 35
- LeGapSetDefaultPhy, 36
- LeGapSetPhy, 36
- LeGapSetRandAddr, 36
- LeGapSetRpaTimeout, 37
- LeGapSetStaticAddr, 37
- LeSetScanParameter, 37
- LeSetScanRspData, 38
- BLE GATT APIs, 39
 - CHAR_AGGREGATE_DESCRIPTOR, 43
 - CHAR_CLIENT_CONFIG_DESCRIPTOR, 43
 - CHAR_DECL_UUID16_ATTR_VAL, 44
 - CHAR_EXT_PROP_DESCRIPTOR, 44
 - CHAR_PRESENT_FORMAT_DESCRIPTOR, 44
 - CHAR_SERVER_CONFIG_DESCRIPTOR, 44
 - CHAR_USER_DESC_DESCRIPTOR, 44
 - CHARACTERISTIC_DECL_UUID128, 44
 - CHARACTERISTIC_DECL_UUID16, 45
 - CHARACTERISTIC_UUID128, 45
 - CHARACTERISTIC_UUID16, 45
 - GATT_CHAR_AGG_FORMAT_UUID, 45
 - GATT_CHAR_EXT_PROPS_UUID, 45
 - GATT_CHAR_FORMAT_UUID, 45
 - GATT_CHAR_USER_DESC_UUID, 46
 - GATT_CHARACTERISTIC_UUID, 46
 - GATT_CLIENT_CHAR_CFG_UUID, 46
 - GATT_EXT_REPORT_REF_UUID, 46
 - GATT_INCLUDE_UUID, 46
 - GATT_PRIMARY_SERVICE_UUID, 46
 - GATT_REPORT_REF_UUID, 46
 - GATT_SECONDARY_SERVICE_UUID, 46
 - GATT_SERV_CHAR_CFG_UUID, 47
 - GATT_VALID_RANGE_UUID, 47
 - gcCharAggregateUuid, 70
 - gcCharExtPropUuid, 70
 - gcCharFormatUuid, 71
 - gcCharUserDescUuid, 71
 - gcCharacteristicUuid, 70
 - gcClientCharConfigUuid, 71
 - gcExtReportRefUuid, 71
 - gcIncludeUuid, 71
 - gcPrimaryServiceUuid, 71
 - gcReportRefUuid, 71
 - gcSecondaryServiceUuid, 71
 - gcServerCharConfigUuid, 72
 - gcValidRangeUuid, 72
 - INCLUDE_DECL_UUID128, 47
 - INCLUDE_DECL_UUID128_ATTR_VAL, 47
 - INCLUDE_DECL_UUID16_ATTR_VAL, 47
 - INCLUDE_DECL_UUINT16, 47
 - LE_ATT_UUID_SIZE, 47
 - LE_GATT_CHAR_PROP_AUTH, 48
 - LE_GATT_CHAR_PROP_BCAST, 48
 - LE_GATT_CHAR_PROP_EXT_PROP, 48
 - LE_GATT_CHAR_PROP_IND, 48
 - LE_GATT_CHAR_PROP_NTF, 48
 - LE_GATT_CHAR_PROP_RD, 48
 - LE_GATT_CHAR_PROP_WR_NO_RESP, 49
 - LE_GATT_CHAR_PROP_WR, 48
 - LE_GATT_CLIENT_CFG_INDICATION, 49
 - LE_GATT_CLIENT_CFG_NOTIFICATION, 49
 - LE_GATT_EXT_PROP_RELIABLE_WR, 49
 - LE_GATT_EXT_PROP_WR_AUX, 49
 - LE_GATT_FLAG_PREPARE_WRITE, 49
 - LE_GATT_FLAG_WRITE_CMD, 49
 - LE_GATT_FLAG_WRITE_REQ, 49
 - LE_GATT_PERM_AUTH_READABLE, 50
 - LE_GATT_PERM_AUTH_WRITABLE, 50
 - LE_GATT_PERM_NONE, 50
 - LE_GATT_PERM_READ, 50
 - LE_GATT_PERM_RELIABLE_WRITE, 50
 - LE_GATT_PERM_WRITE_CMD, 50
 - LE_GATT_PERM_WRITE_REQ, 50
 - LE_GATT_PERMIT_AUTHEN_READ, 50
 - LE_GATT_PERMIT_AUTHEN_WRITE, 51
 - LE_GATT_PERMIT_AUTHOR_READ, 51
 - LE_GATT_PERMIT_AUTHOR_WRITE, 51
 - LE_GATT_PERMIT_ENCRYPT_READ, 51
 - LE_GATT_PERMIT_ENCRYPT_WRITE, 51
 - LE_GATT_PERMIT_READABLE, 51
 - LE_GATT_PERMIT_READ, 51
 - LE_GATT_PERMIT_SC_AUTHEN_READ, 51
 - LE_GATT_PERMIT_SC_AUTHEN_WRITE, 52
 - LE_GATT_PERMIT_WRITABLE, 52
 - LE_GATT_PERMIT_WRITE, 52
 - LeGattAccessReadRsp, 54
 - LeGattAccessWriteRsp, 54

- LeGattChangeAttrVal, 55
- LeGattCharValConfirmation, 55
- LeGattCharValIndicate, 56
- LeGattCharValNotify, 56
- LeGattExchangeMtuReq, 57
- LeGattExchangeMtuRsp, 57
- LeGattExecuteWriteCharValReliable, 57
- LeGattFindAllCharDescriptor, 58
- LeGattFindAllCharacteristic, 58
- LeGattFindAllPrimaryService, 59
- LeGattFindCharacteristicByUuid, 59
- LeGattFindIncludedService, 60
- LeGattFindPrimaryServiceByUuid, 60
- LeGattGetAttrHandle, 60
- LeGattGetAttrVal, 61
- LeGattGetAttrValLen, 61
- LeGattGetAttrValMaxLen, 63
- LeGattInit, 63
- LeGattModifyAttrVal, 64
- LeGattPrepareWriteCharValReliable, 64
- LeGattReadCharValByUuid, 65
- LeGattReadCharValue, 65
- LeGattReadLongCharVal, 66
- LeGattReadMultipleCharVal, 66
- LeGattRegisterIncludeService, 66
- LeGattRegisterService, 67
- LeGattSignedWriteNoRsp, 67
- LeGattStopCurrentProcedure, 68
- LeGattWriteCharVal, 68
- LeGattWriteCharValReliable, 69
- LeGattWriteLongCharVal, 69
- LeGattWriteNoRsp, 70
- PRIMARY_SERVICE_DECL_UUID128, 52
- PRIMARY_SERVICE_DECL_UUID16, 52
- SECONDARY_SERVICE_DECL_UUID128, 52
- SECONDARY_SERVICE_DECL_UUID16, 52
- BLE MSG APIs, 73
 - LE_ATT_MSG_BASE, 74
 - LE_CM_MSG_BASE, 74
 - LE_GATT_MSG_BASE, 74
 - LE_HCI_MSG_BASE, 75
 - LE_L2CAP_MSG_BASE, 75
 - LE_SMP_MSG_BASE, 75
 - LE_SYS_MSG_BASE, 75
 - LeCancelAllMessage, 78
 - LeCancelAllSubMessage, 79
 - LeCancelFirstMessage, 79
 - LeCancelFirstSubMessage, 79
 - LeGetSubMsgId, 80
 - LeHostCreateTask, 80
 - LeHostMessageLoop, 81
 - LeSendMessage, 81
 - LeSendMessageAfter, 81
 - LeSendMessageUnlock, 82
 - LeSendSubMessage, 82
 - LeSendSubMessageAfter, 83
 - LeSendSubMessageUnlock, 83
 - MESSAGE_ALLOCATE, 75
 - MESSAGE_BULID, 75
 - MESSAGE_DATA_BULID, 75
 - MESSAGE_OFFSET, 76
 - MESSAGEID, 76
 - MESSAGE, 76
 - MSGLOCK, 77
 - MSGSUBID, 77
 - MSGTIMER, 77
 - MsgData, 77
 - MsgLock, 77
 - T_HOUR, 76
 - T_MIN, 76
 - T_SEC, 76
 - TASKHANDLER, 77
 - TASKPACK, 78
 - TASK, 77
 - Task, 77
- BLE SMP APIs, 85
 - LE_MAX_BOND_COUNT, 86
 - LE_SM_IO_CAP_DISP_ONLY, 86
 - LE_SM_IO_CAP_DISP_YES_NO, 86
 - LE_SM_IO_CAP_KEYBOARD_DISP, 86
 - LE_SM_IO_CAP_KEYBOARD_ONLY, 87
 - LE_SM_IO_CAP_NO_IO, 87
 - LE_SM_PAIR_MITM_NO, 87
 - LE_SM_PAIR_MITM_YES, 87
 - LE_SM_PAIR_OOB_NO, 87
 - LE_SM_PAIR_OOB_YES, 87
 - LE_SM_PAIR_SC_NO, 87
 - LE_SM_PAIR_SC_YES, 87
 - LeSmpInit, 89
 - LeSmpOobAuthDataRsp, 89
 - LeSmpOobPresent, 89
 - LeSmpPasskeyInput, 90
 - LeSmpScOobComputeConfirmVal, 90
 - LeSmpScOobDataRsp, 90
 - LeSmpSecurityReq, 91
 - LeSmpSecurityRsp, 91
 - LeSmpSetDefaultConfig, 92
 - LeSmpUserConfirmRsp, 92
- bd_addr
 - LE_CM_MSG_READ_BD_ADDR_CFM_T, 169
- beacon_interval
 - auto_conn_info_t, 151
 - mw_wifi_auto_connect_ap_info_t, 223
 - scan_info_t, 230
 - wifi_ap_config_t, 237
 - wifi_auto_connect_info_t, 239
 - wifi_scan_info_t, 251
- bondable
 - LE_SMP_MSG_SLAVE_SECURITY_REQUEST_IND_T, 219
- bonded
 - LE_SMP_MSG_PAIRING_COMPLETE_IND_T, 217
- bssid
 - auto_conn_info_t, 151
 - mw_wifi_auto_connect_ap_info_t, 223

- scan_info_t, 230
- wifi_auto_connect_info_t, 239
- wifi_event_sta_connected_t, 244
- wifi_event_sta_disconnected_t, 245
- wifi_scan_config_t, 249
- wifi_scan_info_t, 251
- wifi_sta_config_t, 254
- bssid_present
 - wifi_sta_config_t, 254
- CHARAggregate_DESCRIPTOR
 - BLE GATT APIs, 43
- CHAR_CLIENT_CONFIG_DESCRIPTOR
 - BLE GATT APIs, 43
- CHAR_DECL_UUID16_ATTR_VAL
 - BLE GATT APIs, 44
- CHAR_EXT_PROP_DESCRIPTOR
 - BLE GATT APIs, 44
- CHAR_PRESENT_FORMAT_DESCRIPTOR
 - BLE GATT APIs, 44
- CHAR_SERVER_CONFIG_DESCRIPTOR
 - BLE GATT APIs, 44
- CHAR_USER_DESC_DESCRIPTOR
 - BLE GATT APIs, 44
- CHARACTERISTIC_DECL_UUID128
 - BLE GATT APIs, 44
- CHARACTERISTIC_DECL_UUID16
 - BLE GATT APIs, 45
- CHARACTERISTIC_UUID128
 - BLE GATT APIs, 45
- CHARACTERISTIC_UUID16
 - BLE GATT APIs, 45
- capabilities
 - _wpa_ie_data, 147
 - auto_conn_info_t, 151
 - mw_wifi_auto_connect_ap_info_t, 223
 - scan_info_t, 230
 - wifi_auto_connect_info_t, 239
 - wifi_wpa_ie_data_t, 256
- capability_info
 - wifi_scan_info_t, 251
- ch_map
 - LE_CM_MSG_READ_CHANNEL_MAP_CFM_T, 170
- channel
 - wifi_ap_config_t, 237
 - wifi_event_sta_connected_t, 244
 - wifi_scan_config_t, 249
 - wifi_scan_info_t, 251
- channel_map
 - LE_GAP_ADVERTISING_PARAM_T, 178
- client_rx_mtu
 - LE_GATT_MSG_EXCHANGE_MTU_IND_T, 190
- cmd_type
 - wifi_cmd_t, 241
- confirm
 - LE_SMP_SC_OOB_DATA_T, 221
- confirm_num
 - LE_SMP_MSG_USER_CONFIRM_IND_T, 220
- conn_hdl
 - LE_CM_CONNECTION_COMPLETE_IND_T, 158
 - LE_CM_MSG_CONN_PARA_REQ_T, 161
 - LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T, 162
 - LE_CM_MSG_DATA_LEN_CHANGE_IND_T, 163
 - LE_CM_MSG_DISCONNECT_COMPLETE_IND_T, 165
 - LE_CM_MSG_ENCRYPTION_CHANGE_IND_T, 165
 - LE_CM_MSG_ENCRYPTION_REFRESH_IND_T, 166
 - LE_CM_MSG_LTK_REQ_IND_T, 168
 - LE_CM_MSG_READ_CHANNEL_MAP_CFM_T, 170
 - LE_CM_MSG_READ_PHY_CFM_T, 171
 - LE_CM_MSG_READ_RSSI_CFM_T, 172
 - LE_CM_MSG_READ_TX_POWER_CFM_T, 173
 - LE_CM_MSG_SET_DATA_LENGTH_CFM_T, 174
 - LE_CM_MSG_SET_PHY_CFM_T, 175
 - LE_CM_MSG_SIGNAL_UPDATE_REQ_T, 176
 - LE_GATT_MSG_ACCESS_READ_IND_T, 183
 - LE_GATT_MSG_ACCESS_WRITE_IND_T, 184
 - LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND_T, 185
 - LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND_T, 186
 - LE_GATT_MSG_CHARACTERISTIC_VAL_IND_T, 188
 - LE_GATT_MSG_CONFIRMATION_CFM_T, 189
 - LE_GATT_MSG_EXCHANGE_MTU_CFM_T, 190
 - LE_GATT_MSG_EXCHANGE_MTU_IND_T, 190
 - LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM_T, 191
 - LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM_T, 192
 - LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_CFM_T, 193
 - LE_GATT_MSG_FIND_CHARACTERISTIC_CFM_T, 194
 - LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM_T, 195
 - LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM_T, 196
 - LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T, 197
 - LE_GATT_MSG_INDICATE_IND_T, 198
 - LE_GATT_MSG_NOTIFY_CFM_T, 199
 - LE_GATT_MSG_NOTIFY_IND_T, 200
 - LE_GATT_MSG_OPERATION_TIMEOUT_T, 201
 - LE_GATT_MSG_PREPARE_WRITE_RELIABLE_CFM_T, 202
 - LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_CFM_T, 203
 - LE_GATT_MSG_READ_CHARACTERISTIC_VALUE_CFM_T, 204
 - LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM_T, 205

- FM_T, 205
- LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL↔
L_CFM_T, 206
- LE_GATT_MSG_SERVICE_INFO_IND_T, 208
- LE_GATT_MSG_SIGNED_WRITE_CFM_T, 209
- LE_GATT_MSG_WRITE_CHAR_VAL_RELIAB↔
LE_CFM_T, 210
- LE_GATT_MSG_WRITE_CHAR_VALUE_CFM↔
_T, 211
- LE_GATT_MSG_WRITE_LONG_CHAR_VALU↔
E_CFM_T, 212
- LE_GATT_MSG_WRITE_NO_RSP_CFM_T, 213
- LE_SMP_MSG_ENCRYPTION_CHANGE_IND↔
_T, 214
- LE_SMP_MSG_ENCRYPTION_REFRESH_IND↔
_T, 215
- LE_SMP_MSG_OOB_DATA_REQUEST_IND_T,
215
- LE_SMP_MSG_PAIRING_ACTION_IND_T, 216
- LE_SMP_MSG_PAIRING_COMPLETE_IND_T,
217
- LE_SMP_MSG_PASSKEY_DISPLAY_IND_T, 218
- LE_SMP_MSG_PASSKEY_INPUT_IND_T, 218
- LE_SMP_MSG_SC_OOB_DATA_REQUEST_I↔
ND_T, 219
- LE_SMP_MSG_SLAVE_SECURITY_REQUES↔
T_IND_T, 220
- LE_SMP_MSG_USER_CONFIRM_IND_T, 221
- LE_SYS_MSG_BUF_OVERFLOW_T, 222
- conn_interval
 - LE_CM_CONNECTION_COMPLETE_IND_T, 158
- conn_latency
 - LE_CM_CONNECTION_COMPLETE_IND_T, 158
- connected
 - wifi_event_info_t, 243
- current_rx_mtu
 - LE_GATT_MSG_EXCHANGE_MTU_CFM_T, 190
- data
 - LE_CM_MSG_ADVERTISE_REPORT_IND↔
T, 160
- dev_id
 - LE_CM_CONNECTION_COMPLETE_IND_T, 158
- devid
 - LE_CM_MSG_ENCRYPTION_CHANGE_IND_T,
166
 - LE_CM_MSG_ENCRYPTION_REFRESH_IND_T,
166
 - LE_CM_MSG_LTK_REQ_IND_T, 168
 - LE_GATT_MSG_ACCESS_READ_IND_T, 183
 - LE_GATT_MSG_ACCESS_WRITE_IND_T, 184
 - LE_GATT_MSG_CHAR_DESCRIPTOR_INFO↔
IND_T, 185
 - LE_GATT_MSG_CHARACTERISTIC_DECL_IN↔
FO_IND_T, 186
 - LE_GATT_MSG_CHARACTERISTIC_VAL_IND↔
_T, 188
 - LE_GATT_MSG_CONFIRMATION_CFM_T, 189
 - LE_GATT_MSG_EXCHANGE_MTU_CFM_T, 190
 - LE_GATT_MSG_EXCHANGE_MTU_IND_T, 191
 - LE_GATT_MSG_EXECUTE_WRITE_RELIABL↔
E_CFM_T, 191
 - LE_GATT_MSG_FIND_ALL_CHAR_DESC_CF↔
M_T, 192
 - LE_GATT_MSG_FIND_ALL_PRIMARY_SERVI↔
CE_CFM_T, 193
 - LE_GATT_MSG_FIND_CHARACTERISTIC_CF↔
M_T, 194
 - LE_GATT_MSG_FIND_INCLUDED_SERVICE↔
CFM_T, 195
 - LE_GATT_MSG_FIND_PRIMARY_SERVICE_B↔
Y_UUID_CFM_T, 196
 - LE_GATT_MSG_INCLUDE_SERVICE_INFO_I↔
ND_T, 197
 - LE_GATT_MSG_INDICATE_IND_T, 199
 - LE_GATT_MSG_NOTIFY_CFM_T, 200
 - LE_GATT_MSG_NOTIFY_IND_T, 200
 - LE_GATT_MSG_OPERATION_TIMEOUT_T, 202
 - LE_GATT_MSG_PREPARE_WRITE_RELIABL↔
E_CFM_T, 202
 - LE_GATT_MSG_READ_CHAR_VAL_BY_UUID↔
_CFM_T, 203
 - LE_GATT_MSG_READ_CHARACTERISTIC_V↔
ALUE_CFM_T, 204
 - LE_GATT_MSG_READ_LONG_CHAR_VAL_C↔
FM_T, 205
 - LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL↔
L_CFM_T, 206
 - LE_GATT_MSG_SERVICE_INFO_IND_T, 208
 - LE_GATT_MSG_SIGNED_WRITE_CFM_T, 209
 - LE_GATT_MSG_WRITE_CHAR_VAL_RELIAB↔
LE_CFM_T, 210
 - LE_GATT_MSG_WRITE_CHAR_VALUE_CFM↔
_T, 211
 - LE_GATT_MSG_WRITE_LONG_CHAR_VALU↔
E_CFM_T, 212
 - LE_GATT_MSG_WRITE_NO_RSP_CFM_T, 213
 - direct_addr
 - LE_CM_MSG_DIRECT_ADV_REPORT_IND_T,
164
 - direct_addr_type
 - LE_CM_MSG_DIRECT_ADV_REPORT_IND_T,
164
 - disconnected
 - wifi_event_info_t, 243
 - dtim_period
 - wifi_scan_info_t, 251
 - dtim_prod
 - auto_conn_info_t, 151
 - mw_wifi_auto_connect_ap_info_t, 223
 - scan_info_t, 230
 - wifi_auto_connect_info_t, 239
 - eap_workaround
 - asso_data, 149
 - eapol_flags
 - asso_data, 149
 - ediv

- LE_CM_MSG_LTK_REQ_IND_T, 168
- enable
 - LE_SMP_MSG_ENCRYPTION_CHANGE_IND_T, 214
- enabled
 - LE_CM_MSG_ENCRYPTION_CHANGE_IND_T, 166
 - LE_CM_MSG_ENCRYPTION_REFRESH_IND_T, 167
- encrypt_type
 - wifi_ap_config_t, 237
- end_hdl
 - LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T, 197
 - LE_GATT_MSG_SERVICE_INFO_IND_T, 208
- endHdl
 - LE_GATT_SERVICE_T, 213
- Enumeration, 140
 - wifi_auth_mode_t, 140
 - wifi_bandwidth_t, 142
 - wifi_cipher_type_t, 142
 - wifi_event_t, 142
 - wifi_mode_t, 143
 - wifi_reason_code_t, 143
 - wifi_scan_method_t, 144
 - wifi_scan_type_t, 144
 - wifi_sort_method_t, 145
- err_hdl
 - LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM_T, 191
 - LE_GATT_MSG_READ_MULTIPLE_CHAR_VALUES_CFM_T, 207
- event
 - event_msg_t, 155
- event_handler
 - wifi_init_config_t, 248
- event_msg_t, 155
 - event, 155
 - length, 155
 - param, 156
- event_type
 - LE_CM_MSG_ADVERTISE_REPORT_IND_T, 160
- evt_type
 - wifi_evt_t, 247
- fast_connect
 - auto_conn_info_t, 152
 - mw_wifi_auto_connect_ap_info_t, 224
 - wifi_auto_connect_info_t, 239
- filter_policy
 - LE_GAP_ADVERTISING_PARAM_T, 178
 - LE_GAP_SCAN_PARAM_T, 181
- flag
 - auto_connect_cfg_t, 154
 - LE_GATT_MSG_ACCESS_WRITE_IND_T, 184
 - MwFimAutoConnectCFG_t, 226
- format
 - LE_GATT_ATTR_T, 182
- LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND_T, 185
- LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND_T, 186
- LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T, 198
- LE_GATT_MSG_SERVICE_INFO_IND_T, 208
- frame_buffer
 - rx_eapol_data, 227
- frame_length
 - rx_eapol_data, 227
- free_ocpy
 - auto_conn_info_t, 152
 - mw_wifi_auto_connect_ap_info_t, 224
 - scan_info_t, 231
- front
 - auto_connect_cfg_t, 154
 - MwFimAutoConnectCFG_t, 226
- GAP_ADTYPE_128BIT_COMPLETE
 - BLE GAP APIs, 19
- GAP_ADTYPE_128BIT_MORE
 - BLE GAP APIs, 19
- GAP_ADTYPE_16BIT_COMPLETE
 - BLE GAP APIs, 19
- GAP_ADTYPE_16BIT_MORE
 - BLE GAP APIs, 20
- GAP_ADTYPE_32BIT_COMPLETE
 - BLE GAP APIs, 20
- GAP_ADTYPE_32BIT_MORE
 - BLE GAP APIs, 20
- GAP_ADTYPE_3D_INFO_DATA
 - BLE GAP APIs, 20
- GAP_ADTYPE_ADV_INTERVAL
 - BLE GAP APIs, 20
- GAP_ADTYPE_APPEARANCE
 - BLE GAP APIs, 20
- GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED
 - BLE GAP APIs, 20
- GAP_ADTYPE_FLAGS_GENERAL
 - BLE GAP APIs, 21
- GAP_ADTYPE_FLAGS_LIMITED
 - BLE GAP APIs, 21
- GAP_ADTYPE_FLAGS
 - BLE GAP APIs, 20
- GAP_ADTYPE_LE_BD_ADDR
 - BLE GAP APIs, 21
- GAP_ADTYPE_LE_ROLE
 - BLE GAP APIs, 21
- GAP_ADTYPE_LOCAL_NAME_COMPLETE
 - BLE GAP APIs, 21
- GAP_ADTYPE_LOCAL_NAME_SHORT
 - BLE GAP APIs, 21
- GAP_ADTYPE_MANUFACTURER_SPECIFIC
 - BLE GAP APIs, 21
- GAP_ADTYPE_OOB_CLASS_OF_DEVICE
 - BLE GAP APIs, 21
- GAP_ADTYPE_OOB_SIMPLE_PAIRING_HASHC
 - BLE GAP APIs, 22

- GAP_ADTYPE_OOB_SIMPLE_PAIRING_RANDR
 - BLE GAP APIs, [22](#)
- GAP_ADTYPE_POWER_LEVEL
 - BLE GAP APIs, [22](#)
- GAP_ADTYPE_PUBLIC_TARGET_ADDR
 - BLE GAP APIs, [22](#)
- GAP_ADTYPE_RANDOM_TARGET_ADDR
 - BLE GAP APIs, [22](#)
- GAP_ADTYPE_SERVICE_DATA_128BIT
 - BLE GAP APIs, [22](#)
- GAP_ADTYPE_SERVICE_DATA_32BIT
 - BLE GAP APIs, [22](#)
- GAP_ADTYPE_SERVICE_DATA
 - BLE GAP APIs, [22](#)
- GAP_ADTYPE_SERVICES_LIST_128BIT
 - BLE GAP APIs, [23](#)
- GAP_ADTYPE_SERVICES_LIST_16BIT
 - BLE GAP APIs, [23](#)
- GAP_ADTYPE_SIGNED_DATA
 - BLE GAP APIs, [23](#)
- GAP_ADTYPE_SIMPLE_PAIRING_HASHC_256
 - BLE GAP APIs, [23](#)
- GAP_ADTYPE_SIMPLE_PAIRING_RANDR_256
 - BLE GAP APIs, [23](#)
- GAP_ADTYPE_SLAVE_CONN_INTERVAL_RANGE
 - BLE GAP APIs, [23](#)
- GAP_ADTYPE_SM_OOB_FLAG
 - BLE GAP APIs, [23](#)
- GAP_ADTYPE_SM_TK
 - BLE GAP APIs, [23](#)
- GAP_PUBLIC_ADDR
 - BLE GAP APIs, [24](#)
- GAP_RAND_ADDR_NRPA
 - BLE GAP APIs, [24](#)
- GAP_RAND_ADDR_RPA
 - BLE GAP APIs, [24](#)
- GAP_RAND_ADDR_STATIC
 - BLE GAP APIs, [24](#)
- GAP_SCAN_TYPE_ACTIVE
 - BLE GAP APIs, [24](#)
- GAP_SCAN_TYPE_PASSIVE
 - BLE GAP APIs, [24](#)
- GAP_TX_PWR_CURR_VAL
 - BLE GAP APIs, [24](#)
- GAP_TX_PWR_MAX_VAL
 - BLE GAP APIs, [24](#)
- GAPBOND_IO_CAP_DISPLAY_ONLY
 - BLE GAP APIs, [25](#)
- GAPBOND_IO_CAP_DISPLAY_YES_NO
 - BLE GAP APIs, [25](#)
- GAPBOND_IO_CAP_KEYBOARD_DISPLAY
 - BLE GAP APIs, [25](#)
- GAPBOND_IO_CAP_KEYBOARD_ONLY
 - BLE GAP APIs, [25](#)
- GAPBOND_IO_CAP_NO_INPUT_NO_OUTPUT
 - BLE GAP APIs, [25](#)
- GAPBOND_PAIRING_MODE_INITIATE
 - BLE GAP APIs, [25](#)
- GAPBOND_PAIRING_MODE_NO_PAIRING
 - BLE GAP APIs, [25](#)
- GAPBOND_PAIRING_MODE_WAIT_FOR_REQ
 - BLE GAP APIs, [25](#)
- GATT_CHAR_AGG_FORMAT_UUID
 - BLE GATT APIs, [45](#)
- GATT_CHAR_EXT_PROPS_UUID
 - BLE GATT APIs, [45](#)
- GATT_CHAR_FORMAT_UUID
 - BLE GATT APIs, [45](#)
- GATT_CHAR_USER_DESC_UUID
 - BLE GATT APIs, [46](#)
- GATT_CHARACTERISTIC_UUID
 - BLE GATT APIs, [46](#)
- GATT_CLIENT_CHAR_CFG_UUID
 - BLE GATT APIs, [46](#)
- GATT_EXT_REPORT_REF_UUID
 - BLE GATT APIs, [46](#)
- GATT_INCLUDE_UUID
 - BLE GATT APIs, [46](#)
- GATT_PRIMARY_SERVICE_UUID
 - BLE GATT APIs, [46](#)
- GATT_REPORT_REF_UUID
 - BLE GATT APIs, [46](#)
- GATT_SECONDARY_SERVICE_UUID
 - BLE GATT APIs, [46](#)
- GATT_SERV_CHAR_CFG_UUID
 - BLE GATT APIs, [47](#)
- GATT_VALID_RANGE_UUID
 - BLE GATT APIs, [47](#)
- gcCharAggregateUuid
 - BLE GATT APIs, [70](#)
- gcCharExtPropUuid
 - BLE GATT APIs, [70](#)
- gcCharFormatUuid
 - BLE GATT APIs, [71](#)
- gcCharUserDescUuid
 - BLE GATT APIs, [71](#)
- gcCharacteristicUuid
 - BLE GATT APIs, [70](#)
- gcClientCharConfigUuid
 - BLE GATT APIs, [71](#)
- gcExtReportRefUuid
 - BLE GATT APIs, [71](#)
- gcIncludeUuid
 - BLE GATT APIs, [71](#)
- gcPrimaryServiceUuid
 - BLE GATT APIs, [71](#)
- gcReportRefUuid
 - BLE GATT APIs, [71](#)
- gcSecondaryServiceUuid
 - BLE GATT APIs, [71](#)
- gcServerCharConfigUuid
 - BLE GATT APIs, [72](#)
- gcValidRangeUuid
 - BLE GATT APIs, [72](#)
- got_ip
 - wifi_event_info_t, [243](#)

- group_cipher
 - _wpa_ie_data, 147
 - asso_data, 149
 - wifi_scan_info_t, 251
 - wifi_wpa_ie_data_t, 256
- handle
 - LE_CM_MSG_SET_DISCONNECT_CFM_T, 175
 - LE_GATT_ATTR_T, 182
 - LE_GATT_MSG_ACCESS_READ_IND_T, 183
 - LE_GATT_MSG_ACCESS_WRITE_IND_T, 184
 - LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_↔
IND_T, 185
 - LE_GATT_MSG_CHARACTERISTIC_DECL_IN↔
FO_IND_T, 187
 - LE_GATT_MSG_CHARACTERISTIC_VAL_IN↔
T, 188
 - LE_GATT_MSG_CONFIRMATION_CFM_T, 189
 - LE_GATT_MSG_FIND_ALL_CHAR_DESC_CF↔
M_T, 192
 - LE_GATT_MSG_FIND_ALL_PRIMARY_SERVI↔
CE_CFM_T, 193
 - LE_GATT_MSG_FIND_CHARACTERISTIC_CF↔
M_T, 194
 - LE_GATT_MSG_FIND_INCLUDED_SERVICE_↔
CFM_T, 195
 - LE_GATT_MSG_FIND_PRIMARY_SERVICE_B↔
Y_UUID_CFM_T, 196
 - LE_GATT_MSG_INCLUDE_SERVICE_INFO_I↔
ND_T, 198
 - LE_GATT_MSG_INDICATE_IND_T, 199
 - LE_GATT_MSG_NOTIFY_CFM_T, 200
 - LE_GATT_MSG_NOTIFY_IND_T, 201
 - LE_GATT_MSG_PREPARE_WRITE_RELIABL↔
E_CFM_T, 202
 - LE_GATT_MSG_READ_CHAR_VAL_BY_UUID↔
_CFM_T, 203
 - LE_GATT_MSG_READ_CHARACTERISTIC_V↔
ALUE_CFM_T, 204
 - LE_GATT_MSG_READ_LONG_CHAR_VAL_C↔
FM_T, 205
 - LE_GATT_MSG_SIGNED_WRITE_CFM_T, 209
 - LE_GATT_MSG_WRITE_CHAR_VAL_RELIAB↔
LE_CFM_T, 210
 - LE_GATT_MSG_WRITE_CHAR_VALUE_CFM↔
T, 211
 - LE_GATT_MSG_WRITE_LONG_CHAR_VALU↔
E_CFM_T, 212
 - LE_GATT_MSG_WRITE_NO_RSP_CFM_T, 213
- hap_ap_info
 - hap_control_t, 156
- hap_bitvector
 - hap_control_t, 156
- hap_control_t, 156
 - hap_ap_info, 156
 - hap_bitvector, 156
 - hap_en, 156
 - hap_final_index, 156
 - hap_index, 157
 - hap_ssid, 157
- hap_en
 - hap_control_t, 156
- hap_final_index
 - hap_control_t, 156
- hap_index
 - hap_control_t, 157
- hap_ssid
 - hap_control_t, 157
- hid_ssid
 - auto_conn_info_t, 152
 - mw_wifi_auto_connect_ap_info_t, 224
 - wifi_auto_connect_info_t, 239
- hid_ssid_len
 - auto_conn_info_t, 152
 - mw_wifi_auto_connect_ap_info_t, 224
- iArgc
 - T_RfCmd, 233
- INCLUDE_DECL_UUID128
 - BLE GATT APIs, 47
- INCLUDE_DECL_UUID128_ATTR_VAL
 - BLE GATT APIs, 47
- INCLUDE_DECL_UUID16_ATTR_VAL
 - BLE GATT APIs, 47
- INCLUDE_DECL_UINT16
 - BLE GATT APIs, 47
- identifier
 - LE_CM_MSG_SIGNAL_UPDATE_REQ_T, 176
- interval
 - LE_CM_MSG_CONN_UPDATE_COMPLETE_I↔
ND_T, 162
 - LE_GAP_SCAN_PARAM_T, 181
- interval_max
 - LE_CM_MSG_SIGNAL_UPDATE_REQ_T, 176
 - LE_GAP_ADVERTISING_PARAM_T, 179
 - LE_GAP_CONN_PARAM_T, 180
- interval_min
 - LE_CM_MSG_SIGNAL_UPDATE_REQ_T, 176
 - LE_GAP_ADVERTISING_PARAM_T, 179
 - LE_GAP_CONN_PARAM_T, 180
- ip_changed
 - wifi_event_sta_got_ip_t, 246
- itv_max
 - LE_CM_MSG_CONN_PARA_REQ_T, 161
 - LE_CONN_PARA_T, 177
- itv_min
 - LE_CM_MSG_CONN_PARA_REQ_T, 161
 - LE_CONN_PARA_T, 177
- key_mgmt
 - _wpa_ie_data, 147
 - asso_data, 149
 - wifi_wpa_ie_data_t, 256
- keypress
 - LE_SMP_MSG_SLAVE_SECURITY_REQUES↔
T_IND_T, 220
- LE_ATT_MSG_BASE

- BLE MSG APIs, 74
- LE_ATT_UUID_SIZE
 - BLE GATT APIs, 47
- LE_BT_ADDR_T, 157
 - addr, 157
 - type, 157
- LE_CM_CONNECTION_COMPLETE_IND_T, 158
 - conn_hdl, 158
 - conn_interval, 158
 - conn_latency, 158
 - dev_id, 158
 - peer_addr, 158
 - peer_addr_type, 159
 - role, 159
 - status, 159
 - supervision_timeout, 159
- LE_CM_MSG_ADD_TO_RESOLVING_LIST_CFM_T
 - BLE CM APIs, 11
- LE_CM_MSG_ADD_TO_WHITE_LIST_CFM_T
 - BLE CM APIs, 11
- LE_CM_MSG_ADVERTISE_REPORT_IND_T, 159
 - addr, 160
 - addr_type, 160
 - data, 160
 - event_type, 160
 - len, 160
 - rss_i, 160
- LE_CM_MSG_BASE
 - BLE MSG APIs, 74
- LE_CM_MSG_CANCEL_CONNECTION_CFM_T
 - BLE CM APIs, 11
- LE_CM_MSG_CLEAR_RESOLVING_LIST_CFM_T
 - BLE CM APIs, 12
- LE_CM_MSG_CLEAR_WHITE_LIST_CFM_T
 - BLE CM APIs, 12
- LE_CM_MSG_CONN_PARA_REQ_T, 160
 - conn_hdl, 161
 - itv_max, 161
 - itv_min, 161
 - latency, 161
 - sv_tmo, 161
- LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T, 161
 - conn_hdl, 162
 - interval, 162
 - latency, 162
 - status, 162
 - supervision_timeout, 162
- LE_CM_MSG_CREATE_CONNECTION_CFM_T
 - BLE CM APIs, 12
- LE_CM_MSG_DATA_LEN_CHANGE_IND_T, 162
 - conn_hdl, 163
 - max_rx_octets, 163
 - max_rx_time, 163
 - max_tx_octets, 163
 - max_tx_time, 163
- LE_CM_MSG_DIRECT_ADV_REPORT_IND_T, 163
 - direct_addr, 164
 - direct_addr_type, 164
 - peer_addr, 164
 - peer_addr_type, 164
 - rss_i, 164
- LE_CM_MSG_DISCONNECT_COMPLETE_IND_T, 164
 - conn_hdl, 165
 - reason, 165
 - status, 165
- LE_CM_MSG_ENCRYPTION_CHANGE_IND_T, 165
 - conn_hdl, 165
 - devid, 166
 - enabled, 166
 - status, 166
- LE_CM_MSG_ENCRYPTION_REFRESH_IND_T, 166
 - conn_hdl, 166
 - devid, 166
 - enabled, 167
 - status, 167
- LE_CM_MSG_ENTER_ADVERTISING_CFM_T
 - BLE CM APIs, 12
- LE_CM_MSG_ENTER_SCANNING_CFM_T
 - BLE CM APIs, 12
- LE_CM_MSG_EXIT_ADVERTISING_CFM_T
 - BLE CM APIs, 12
- LE_CM_MSG_EXIT_SCANNING_CFM_T
 - BLE CM APIs, 12
- LE_CM_MSG_INIT_COMPLETE_CFM_T, 167
 - status, 167
- LE_CM_MSG_LTK_REQ_IND_T, 167
 - conn_hdl, 168
 - devid, 168
 - ediv, 168
 - rand, 168
- LE_CM_MSG_PHY_UPDATE_COMPLETE_IND_T
 - BLE CM APIs, 12
- LE_CM_MSG_READ_ADV_TX_POWER_CFM_T, 168
 - pwr_level, 169
 - status, 169
- LE_CM_MSG_READ_BD_ADDR_CFM_T, 169
 - bd_addr, 169
 - status, 169
- LE_CM_MSG_READ_CHANNEL_MAP_CFM_T, 170
 - ch_map, 170
 - conn_hdl, 170
 - status, 170
- LE_CM_MSG_READ_PHY_CFM_T, 170
 - conn_hdl, 171
 - rx_phy, 171
 - status, 171
 - tx_phy, 171
- LE_CM_MSG_READ_RESOLVING_LIST_SIZE_CFM_T, 171
 - size, 171
 - status, 171
- LE_CM_MSG_READ_RSSI_CFM_T, 172
 - conn_hdl, 172
 - rss_i, 172

- status, [172](#)
- LE_CM_MSG_READ_TX_POWER_CFM_T, [172](#)
 - conn_hdl, [173](#)
 - status, [173](#)
 - tx_power, [173](#)
- LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM_T, [173](#)
 - size, [173](#)
 - status, [174](#)
- LE_CM_MSG_REMOVE_FROM_RESOLVING_LIST_CFM_T, [13](#)
 - BLE CM APIs, [13](#)
- LE_CM_MSG_REMOVE_FROM_WHITE_LIST_CFM_T, [13](#)
 - BLE CM APIs, [13](#)
- LE_CM_MSG_SET_ADVERTISING_DATA_CFM_T, [13](#)
 - BLE CM APIs, [13](#)
- LE_CM_MSG_SET_ADVERTISING_PARAMS_CFM_T, [13](#)
 - BLE CM APIs, [13](#)
- LE_CM_MSG_SET_CHANNEL_MAP_CFM_T, [13](#)
 - BLE CM APIs, [13](#)
- LE_CM_MSG_SET_DATA_LENGTH_CFM_T, [174](#)
 - conn_hdl, [174](#)
 - status, [174](#)
- LE_CM_MSG_SET_DEFAULT_PHY_CFM_T, [13](#)
 - BLE CM APIs, [13](#)
- LE_CM_MSG_SET_DISCONNECT_CFM_T, [174](#)
 - handle, [175](#)
 - status, [175](#)
- LE_CM_MSG_SET_PHY_CFM_T, [175](#)
 - conn_hdl, [175](#)
 - status, [175](#)
- LE_CM_MSG_SET_RANDOM_ADDRESS_CFM_T, [13](#)
 - BLE CM APIs, [13](#)
- LE_CM_MSG_SET_RPA_TIMEOUT_CFM_T, [13](#)
 - BLE CM APIs, [13](#)
- LE_CM_MSG_SET_SCAN_PARAMS_CFM_T, [14](#)
 - BLE CM APIs, [14](#)
- LE_CM_MSG_SET_SCAN_RSP_DATA_CFM_T, [14](#)
 - BLE CM APIs, [14](#)
- LE_CM_MSG_SIGNAL_UPDATE_REQ_T, [176](#)
 - conn_hdl, [176](#)
 - identifier, [176](#)
 - interval_max, [176](#)
 - interval_min, [176](#)
 - slave_latency, [176](#)
 - timeout_multiplier, [176](#)
- LE_CM_REQ_STATUS_T, [177](#)
 - status, [177](#)
- LE_CONN_PARA_T, [177](#)
 - itv_max, [177](#)
 - itv_min, [177](#)
 - latency, [178](#)
 - sv_timeout, [178](#)
- LE_GAP_ADV_MAX_SIZE, [26](#)
 - BLE GAP APIs, [26](#)
- LE_GAP_ADVERTISING_PARAM_T, [178](#)
 - channel_map, [178](#)
 - filter_policy, [178](#)
 - interval_max, [179](#)
 - interval_min, [179](#)
 - own_addr_type, [179](#)
 - peer_addr, [179](#)
 - peer_addr_type, [179](#)
 - type, [179](#)
- LE_GAP_CONN_PARAM_T, [179](#)
 - interval_max, [180](#)
 - interval_min, [180](#)
 - latency, [180](#)
 - supervision_timeout, [180](#)
- LE_GAP_SCAN_PARAM_T, [180](#)
 - filter_policy, [181](#)
 - interval, [181](#)
 - own_addr_type, [181](#)
 - type, [181](#)
 - window, [181](#)
- LE_GATT_ATTR_T, [181](#)
 - format, [182](#)
 - handle, [182](#)
 - len, [182](#)
 - maxLen, [182](#)
 - pUuid, [182](#)
 - pVal, [182](#)
 - permit, [182](#)
- LE_GATT_CHAR_PROP_AUTH, [48](#)
 - BLE GATT APIs, [48](#)
- LE_GATT_CHAR_PROP_BCAST, [48](#)
 - BLE GATT APIs, [48](#)
- LE_GATT_CHAR_PROP_EXT_PROP, [48](#)
 - BLE GATT APIs, [48](#)
- LE_GATT_CHAR_PROP_IND, [48](#)
 - BLE GATT APIs, [48](#)
- LE_GATT_CHAR_PROP_NTF, [48](#)
 - BLE GATT APIs, [48](#)
- LE_GATT_CHAR_PROP_RD, [48](#)
 - BLE GATT APIs, [48](#)
- LE_GATT_CHAR_PROP_WR_NO_RESP, [49](#)
 - BLE GATT APIs, [49](#)
- LE_GATT_CHAR_PROP_WR, [48](#)
 - BLE GATT APIs, [48](#)
- LE_GATT_CLIENT_CFG_INDICATION, [49](#)
 - BLE GATT APIs, [49](#)
- LE_GATT_CLIENT_CFG_NOTIFICATION, [49](#)
 - BLE GATT APIs, [49](#)
- LE_GATT_EXT_PROP_RELIABLE_WR, [49](#)
 - BLE GATT APIs, [49](#)
- LE_GATT_EXT_PROP_WR_AUX, [49](#)
 - BLE GATT APIs, [49](#)
- LE_GATT_FLAG_PREPARE_WRITE, [49](#)
 - BLE GATT APIs, [49](#)
- LE_GATT_FLAG_WRITE_CMD, [49](#)
 - BLE GATT APIs, [49](#)
- LE_GATT_FLAG_WRITE_REQ, [49](#)
 - BLE GATT APIs, [49](#)
- LE_GATT_MSG_ACCESS_READ_IND_T, [183](#)

- conn_hdl, 183
- devid, 183
- handle, 183
- offset, 183
- LE_GATT_MSG_ACCESS_WRITE_IND_T, 183
 - conn_hdl, 184
 - devid, 184
 - flag, 184
 - handle, 184
 - len, 184
 - offset, 184
 - pVal, 185
- LE_GATT_MSG_BASE
 - BLE MSG APIs, 74
- LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_IND_T, 185
 - conn_hdl, 185
 - devid, 185
 - format, 185
 - handle, 185
 - uuid, 186
- LE_GATT_MSG_CHARACTERISTIC_DECL_INFO_IND_T, 186
 - conn_hdl, 186
 - devid, 186
 - format, 186
 - handle, 187
 - property, 187
 - uuid, 187
 - val_hdl, 187
- LE_GATT_MSG_CHARACTERISTIC_VAL_IND_T, 187
 - att_err, 188
 - conn_hdl, 188
 - devid, 188
 - handle, 188
 - len, 188
 - offset, 188
 - val, 188
- LE_GATT_MSG_CONFIRMATION_CFM_T, 189
 - conn_hdl, 189
 - devid, 189
 - handle, 189
- LE_GATT_MSG_EXCHANGE_MTU_CFM_T, 189
 - conn_hdl, 190
 - current_rx_mtu, 190
 - devid, 190
- LE_GATT_MSG_EXCHANGE_MTU_IND_T, 190
 - client_rx_mtu, 190
 - conn_hdl, 190
 - devid, 191
- LE_GATT_MSG_EXECUTE_WRITE_RELIABLE_CFM_T, 191
 - att_err, 191
 - conn_hdl, 191
 - devid, 191
 - err_hdl, 191
 - status, 192
- LE_GATT_MSG_FIND_ALL_CHAR_DESC_CFM_T, 192
 - att_err, 192
 - conn_hdl, 192
 - devid, 192
 - handle, 192
 - status, 193
- LE_GATT_MSG_FIND_ALL_PRIMARY_SERVICE_CFM_T, 193
 - att_err, 193
 - conn_hdl, 193
 - devid, 193
 - handle, 193
 - status, 194
- LE_GATT_MSG_FIND_CHARACTERISTIC_CFM_T, 194
 - att_err, 194
 - conn_hdl, 194
 - devid, 194
 - handle, 194
 - status, 195
- LE_GATT_MSG_FIND_INCLUDED_SERVICE_CFM_T, 195
 - att_err, 195
 - conn_hdl, 195
 - devid, 195
 - handle, 195
 - status, 196
- LE_GATT_MSG_FIND_PRIMARY_SERVICE_BY_UUID_CFM_T, 196
 - att_err, 196
 - conn_hdl, 196
 - devid, 196
 - handle, 196
 - status, 197
- LE_GATT_MSG_INCLUDE_SERVICE_INFO_IND_T, 197
 - conn_hdl, 197
 - devid, 197
 - end_hdl, 197
 - format, 198
 - handle, 198
 - start_hdl, 198
 - uuid, 198
- LE_GATT_MSG_INDICATE_IND_T, 198
 - conn_hdl, 198
 - devid, 199
 - handle, 199
 - len, 199
 - val, 199
- LE_GATT_MSG_NOTIFY_CFM_T, 199
 - conn_hdl, 199
 - devid, 200
 - handle, 200
 - status, 200
- LE_GATT_MSG_NOTIFY_IND_T, 200
 - conn_hdl, 200
 - devid, 200

- handle, 201
- len, 201
- val, 201
- LE_GATT_MSG_OPERATION_TIMEOUT_T, 201
 - att_op, 201
 - conn_hdl, 201
 - devid, 202
- LE_GATT_MSG_PREPARE_WRITE_RELIABLE_CF↔
 - M_T, 202
 - att_err, 202
 - conn_hdl, 202
 - devid, 202
 - handle, 202
 - status, 203
- LE_GATT_MSG_READ_CHAR_VAL_BY_UUID_CF↔
 - M_T, 203
 - att_err, 203
 - conn_hdl, 203
 - devid, 203
 - handle, 203
 - status, 204
- LE_GATT_MSG_READ_CHARACTERISTIC_VALU↔
 - E_CFM_T, 204
 - att_err, 204
 - conn_hdl, 204
 - devid, 204
 - handle, 204
 - status, 205
- LE_GATT_MSG_READ_LONG_CHAR_VAL_CFM_T, 205
 - att_err, 205
 - conn_hdl, 205
 - devid, 205
 - handle, 205
 - status, 206
- LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL_C↔
 - FM_T, 206
 - att_err, 206
 - conn_hdl, 206
 - devid, 206
 - err_hdl, 207
 - len, 207
 - status, 207
 - val, 207
- LE_GATT_MSG_SERVICE_INFO_IND_T, 207
 - conn_hdl, 208
 - devid, 208
 - end_hdl, 208
 - format, 208
 - start_hdl, 208
 - uuid, 208
- LE_GATT_MSG_SIGNED_WRITE_CFM_T, 208
 - conn_hdl, 209
 - devid, 209
 - handle, 209
 - status, 209
- LE_GATT_MSG_WRITE_CHAR_VAL_RELIABLE_C↔
 - FM_T, 209
 - att_err, 210
 - conn_hdl, 210
 - devid, 210
 - handle, 210
 - status, 210
- LE_GATT_MSG_WRITE_CHAR_VALUE_CFM_T, 210
 - att_err, 211
 - conn_hdl, 211
 - devid, 211
 - handle, 211
 - status, 211
- LE_GATT_MSG_WRITE_LONG_CHAR_VALUE_C↔
 - M_T, 211
 - att_err, 212
 - conn_hdl, 212
 - devid, 212
 - handle, 212
 - status, 212
- LE_GATT_MSG_WRITE_NO_RSP_CFM_T, 212
 - conn_hdl, 213
 - devid, 213
 - handle, 213
 - status, 213
- LE_GATT_PERM_AUTH_READABLE
 - BLE GATT APIs, 50
- LE_GATT_PERM_AUTH_WRITABLE
 - BLE GATT APIs, 50
- LE_GATT_PERM_NONE
 - BLE GATT APIs, 50
- LE_GATT_PERM_READ
 - BLE GATT APIs, 50
- LE_GATT_PERM_RELIABLE_WRITE
 - BLE GATT APIs, 50
- LE_GATT_PERM_WRITE_CMD
 - BLE GATT APIs, 50
- LE_GATT_PERM_WRITE_REQ
 - BLE GATT APIs, 50
- LE_GATT_PERMIT_AUTHEN_READ
 - BLE GATT APIs, 50
- LE_GATT_PERMIT_AUTHEN_WRITE
 - BLE GATT APIs, 51
- LE_GATT_PERMIT_AUTHOR_READ
 - BLE GATT APIs, 51
- LE_GATT_PERMIT_AUTHOR_WRITE
 - BLE GATT APIs, 51
- LE_GATT_PERMIT_ENCRYPT_READ
 - BLE GATT APIs, 51
- LE_GATT_PERMIT_ENCRYPT_WRITE
 - BLE GATT APIs, 51
- LE_GATT_PERMIT_READABLE
 - BLE GATT APIs, 51
- LE_GATT_PERMIT_READ
 - BLE GATT APIs, 51
- LE_GATT_PERMIT_SC_AUTHEN_READ
 - BLE GATT APIs, 51
- LE_GATT_PERMIT_SC_AUTHEN_WRITE
 - BLE GATT APIs, 52
- LE_GATT_PERMIT_WRITABLE

- BLE GATT APIs, 52
- LE_GATT_PERMIT_WRITE
 - BLE GATT APIs, 52
- LE_GATT_SERVICE_T, 213
 - endHdl, 213
 - pAttr, 214
 - startHdl, 214
 - svc_id, 214
- LE_HCI_MSG_BASE
 - BLE MSG APIs, 75
- LE_L2CAP_MSG_BASE
 - BLE MSG APIs, 75
- LE_MAX_BOND_COUNT
 - BLE SMP APIs, 86
- LE_SM_IO_CAP_DISP_ONLY
 - BLE SMP APIs, 86
- LE_SM_IO_CAP_DISP_YES_NO
 - BLE SMP APIs, 86
- LE_SM_IO_CAP_KEYBOARD_DISP
 - BLE SMP APIs, 86
- LE_SM_IO_CAP_KEYBOARD_ONLY
 - BLE SMP APIs, 87
- LE_SM_IO_CAP_NO_IO
 - BLE SMP APIs, 87
- LE_SM_PAIR_MITM_NO
 - BLE SMP APIs, 87
- LE_SM_PAIR_MITM_YES
 - BLE SMP APIs, 87
- LE_SM_PAIR_OOB_NO
 - BLE SMP APIs, 87
- LE_SM_PAIR_OOB_YES
 - BLE SMP APIs, 87
- LE_SM_PAIR_SC_NO
 - BLE SMP APIs, 87
- LE_SM_PAIR_SC_YES
 - BLE SMP APIs, 87
- LE_SMP_MSG_BASE
 - BLE MSG APIs, 75
- LE_SMP_MSG_ENCRYPTION_CHANGE_IND_T, 214
 - conn_hdl, 214
 - enable, 214
- LE_SMP_MSG_ENCRYPTION_REFRESH_IND_T, 215
 - conn_hdl, 215
 - status, 215
- LE_SMP_MSG_OOB_DATA_REQUEST_IND_T, 215
 - conn_hdl, 215
- LE_SMP_MSG_PAIRING_ACTION_IND_T, 216
 - action, 216
 - conn_hdl, 216
 - lost_bond, 216
 - sc, 216
- LE_SMP_MSG_PAIRING_COMPLETE_IND_T, 217
 - authenticated, 217
 - bonded, 217
 - conn_hdl, 217
 - peer_id_addr, 217
 - sc, 217
 - status, 217
- LE_SMP_MSG_PASSKEY_DISPLAY_IND_T, 218
 - conn_hdl, 218
 - passkey, 218
- LE_SMP_MSG_PASSKEY_INPUT_IND_T, 218
 - conn_hdl, 218
- LE_SMP_MSG_SC_OOB_DATA_REQUEST_IND_T, 219
 - conn_hdl, 219
- LE_SMP_MSG_SLAVE_SECURITY_REQUEST_IND_T, 219
 - bondable, 219
 - conn_hdl, 220
 - keypress, 220
 - mitm, 220
 - sc, 220
- LE_SMP_MSG_USER_CONFIRM_IND_T, 220
 - confirm_num, 220
 - conn_hdl, 221
- LE_SMP_SC_OOB_DATA_T, 221
 - confirm, 221
 - rand, 221
- LE_SYS_MSG_BASE
 - BLE MSG APIs, 75
- LE_SYS_MSG_BUF_OVERFLOW_T, 221
 - conn_hdl, 222
- latency
 - LE_CM_MSG_CONN_PARA_REQ_T, 161
 - LE_CM_MSG_CONN_UPDATE_COMPLETE_IND_T, 162
 - LE_CONN_PARA_T, 178
 - LE_GAP_CONN_PARAM_T, 180
- latest_beacon_rx_time
 - auto_conn_info_t, 152
 - mw_wifi_auto_connect_ap_info_t, 224
 - scan_info_t, 231
 - wifi_auto_connect_info_t, 240
- LeCancelAllMessage
 - BLE MSG APIs, 78
- LeCancelAllSubMessage
 - BLE MSG APIs, 79
- LeCancelFirstMessage
 - BLE MSG APIs, 79
- LeCancelFirstSubMessage
 - BLE MSG APIs, 79
- LeCmInit
 - BLE CM APIs, 15
- LeGapAddToResolvingList
 - BLE GAP APIs, 26
- LeGapAddToWhiteList
 - BLE GAP APIs, 26
- LeGapAdvertisingEnable
 - BLE GAP APIs, 27
- LeGapCentralConnectReq
 - BLE GAP APIs, 27
- LeGapCentralSetDataChannel
 - BLE GAP APIs, 27
- LeGapClearResolvingList

- BLE GAP APIs, [29](#)
- LeGapClearWhiteList
 - BLE GAP APIs, [29](#)
- LeGapConnParaRequestRsp
 - BLE GAP APIs, [29](#)
- LeGapConnUpdateRequest
 - BLE GAP APIs, [30](#)
- LeGapConnUpdateResponse
 - BLE GAP APIs, [30](#)
- LeGapConnectCancelReq
 - BLE GAP APIs, [29](#)
- LeGapDisconnectReq
 - BLE GAP APIs, [31](#)
- LeGapGenRandAddr
 - BLE GAP APIs, [31](#)
- LeGapGetBtAddr
 - BLE GAP APIs, [31](#)
- LeGapReadAdvChannelTxPower
 - BLE GAP APIs, [31](#)
- LeGapReadChannelMap
 - BLE GAP APIs, [32](#)
- LeGapReadPhy
 - BLE GAP APIs, [32](#)
- LeGapReadResolvingListSize
 - BLE GAP APIs, [32](#)
- LeGapReadRssi
 - BLE GAP APIs, [32](#)
- LeGapReadTxPower
 - BLE GAP APIs, [33](#)
- LeGapReadWhiteListSize
 - BLE GAP APIs, [33](#)
- LeGapRemoveFromWhiteList
 - BLE GAP APIs, [33](#)
- LeGapScanningReq
 - BLE GAP APIs, [34](#)
- LeGapSetAdvData
 - BLE GAP APIs, [34](#)
- LeGapSetAdvParameter
 - BLE GAP APIs, [35](#)
- LeGapSetConnParameter
 - BLE GAP APIs, [35](#)
- LeGapSetDataChannelPduLen
 - BLE GAP APIs, [35](#)
- LeGapSetDefaultPhy
 - BLE GAP APIs, [36](#)
- LeGapSetPhy
 - BLE GAP APIs, [36](#)
- LeGapSetRandAddr
 - BLE GAP APIs, [36](#)
- LeGapSetRpaTimeout
 - BLE GAP APIs, [37](#)
- LeGapSetStaticAddr
 - BLE GAP APIs, [37](#)
- LeGattAccessReadRsp
 - BLE GATT APIs, [54](#)
- LeGattAccessWriteRsp
 - BLE GATT APIs, [54](#)
- LeGattChangeAttrVal
 - BLE GATT APIs, [55](#)
- LeGattCharValConfirmation
 - BLE GATT APIs, [55](#)
- LeGattCharValIndicate
 - BLE GATT APIs, [56](#)
- LeGattCharValNotify
 - BLE GATT APIs, [56](#)
- LeGattExchangeMtuReq
 - BLE GATT APIs, [57](#)
- LeGattExchangeMtuRsp
 - BLE GATT APIs, [57](#)
- LeGattExecuteWriteCharValReliable
 - BLE GATT APIs, [57](#)
- LeGattFindAllCharDescriptor
 - BLE GATT APIs, [58](#)
- LeGattFindAllCharacteristic
 - BLE GATT APIs, [58](#)
- LeGattFindAllPrimaryService
 - BLE GATT APIs, [59](#)
- LeGattFindCharacteristicByUuid
 - BLE GATT APIs, [59](#)
- LeGattFindIncludedService
 - BLE GATT APIs, [60](#)
- LeGattFindPrimaryServiceByUuid
 - BLE GATT APIs, [60](#)
- LeGattGetAttrHandle
 - BLE GATT APIs, [60](#)
- LeGattGetAttrVal
 - BLE GATT APIs, [61](#)
- LeGattGetAttrValLen
 - BLE GATT APIs, [61](#)
- LeGattGetAttrValMaxLen
 - BLE GATT APIs, [63](#)
- LeGattInit
 - BLE GATT APIs, [63](#)
- LeGattModifyAttrVal
 - BLE GATT APIs, [64](#)
- LeGattPrepareWriteCharValReliable
 - BLE GATT APIs, [64](#)
- LeGattReadCharValByUuid
 - BLE GATT APIs, [65](#)
- LeGattReadCharValue
 - BLE GATT APIs, [65](#)
- LeGattReadLongCharVal
 - BLE GATT APIs, [66](#)
- LeGattReadMultipleCharVal
 - BLE GATT APIs, [66](#)
- LeGattRegisterIncludeService
 - BLE GATT APIs, [66](#)
- LeGattRegisterService
 - BLE GATT APIs, [67](#)
- LeGattSignedWriteNoRsp
 - BLE GATT APIs, [67](#)
- LeGattStopCurrentProcedure
 - BLE GATT APIs, [68](#)
- LeGattWriteCharVal
 - BLE GATT APIs, [68](#)
- LeGattWriteCharValReliable

- BLE GATT APIs, 69
- LeGattWriteLongCharVal
 - BLE GATT APIs, 69
- LeGattWriteNoRsp
 - BLE GATT APIs, 70
- LeGetSubMsgId
 - BLE MSG APIs, 80
- LeHostCreateTask
 - BLE MSG APIs, 80
- LeHostMessageLoop
 - BLE MSG APIs, 81
- LeSendMessage
 - BLE MSG APIs, 81
- LeSendMessageAfter
 - BLE MSG APIs, 81
- LeSendMessageUnlock
 - BLE MSG APIs, 82
- LeSendSubMessage
 - BLE MSG APIs, 82
- LeSendSubMessageAfter
 - BLE MSG APIs, 83
- LeSendSubMessageUnlock
 - BLE MSG APIs, 83
- LeSetScanParameter
 - BLE GAP APIs, 37
- LeSetScanRspData
 - BLE GAP APIs, 38
- LeSmpGetBondIdFromAddr
 - BLE ALL APIs, 9
- LeSmpInit
 - BLE SMP APIs, 89
- LeSmpOobAuthDataRsp
 - BLE SMP APIs, 89
- LeSmpOobPresent
 - BLE SMP APIs, 89
- LeSmpPasskeyInput
 - BLE SMP APIs, 90
- LeSmpScOobComputeConfirmVal
 - BLE SMP APIs, 90
- LeSmpScOobDataRsp
 - BLE SMP APIs, 90
- LeSmpSecurityReq
 - BLE SMP APIs, 91
- LeSmpSecurityRsp
 - BLE SMP APIs, 91
- LeSmpSetDefaultConfig
 - BLE SMP APIs, 92
- LeSmpUserConfirmRsp
 - BLE SMP APIs, 92
- leap
 - asso_data, 149
- len
 - LE_CM_MSG_ADVERTISE_REPORT_IND_↔
T, 160
 - LE_GATT_ATTR_T, 182
 - LE_GATT_MSG_ACCESS_WRITE_IND_T, 184
 - LE_GATT_MSG_CHARACTERISTIC_VAL_IND_↔
T, 188
 - LE_GATT_MSG_INDICATE_IND_T, 199
 - LE_GATT_MSG_NOTIFY_IND_T, 201
 - LE_GATT_MSG_READ_MULTIPLE_CHAR_VAL_↔
L_CFM_T, 207
- length
 - event_msg_t, 155
- lost_bond
 - LE_SMP_MSG_PAIRING_ACTION_IND_T, 216
- MESSAGE_ALLOCATE
 - BLE MSG APIs, 75
- MESSAGE_BULID
 - BLE MSG APIs, 75
- MESSAGE_DATA_BULID
 - BLE MSG APIs, 75
- MESSAGE_OFFSET
 - BLE MSG APIs, 76
- MESSAGEID
 - BLE MSG APIs, 76
- MESSAGE
 - BLE MSG APIs, 76
- MSGLOCK
 - BLE MSG APIs, 77
- MSGSUBID
 - BLE MSG APIs, 77
- MSGTIMER
 - BLE MSG APIs, 77
- magic
 - wifi_init_config_t, 249
- manufacture_name
 - mw_blewifi_cbs_store_t, 222
- max
 - wifi_active_scan_time_t, 236
- max_connection
 - wifi_ap_config_t, 237
- max_rx_octets
 - LE_CM_MSG_DATA_LEN_CHANGE_IND_T, 163
- max_rx_time
 - LE_CM_MSG_DATA_LEN_CHANGE_IND_T, 163
- max_save_num
 - auto_connect_cfg_t, 154
 - MwFimAutoConnectCFG_t, 227
- max_tx_octets
 - LE_CM_MSG_DATA_LEN_CHANGE_IND_T, 163
- max_tx_time
 - LE_CM_MSG_DATA_LEN_CHANGE_IND_T, 163
- maxLen
 - LE_GATT_ATTR_T, 182
- mgmt_group_cipher
 - _wpa_ie_data, 148
 - asso_data, 149
 - wifi_wpa_ie_data_t, 256
- min
 - wifi_active_scan_time_t, 236
- mitm
 - LE_SMP_MSG_SLAVE_SECURITY_REQUEST_↔
T_IND_T, 220
- MsgData
 - BLE MSG APIs, 77

- MsgLock
 - BLE MSG APIs, 77
- mw_blewifi_cbs_store_t, 222
 - manufacture_name, 222
- mw_wifi_auto_connect_ap_info_t, 222
 - ap_channel, 223
 - beacon_interval, 223
 - bssid, 223
 - capabilities, 223
 - dtim_prod, 223
 - fast_connect, 224
 - free_ocpy, 224
 - hid_ssid, 224
 - hid_ssid_len, 224
 - latest_beacon_rx_time, 224
 - passphrase, 224
 - psk, 224
 - rsn_ie, 224
 - rssi, 225
 - ssid, 225
 - ssid_len, 225
 - supported_rates, 225
 - wpa_data, 225
 - wpa_ie, 225
- mw_wifi_sta_info_t, 225
 - au8Dot11MACAddress, 226
 - u8SkipDtimPeriods, 226
- MwFimAutoConnectCFG_t, 226
 - flag, 226
 - front, 226
 - max_save_num, 227
 - rear, 227
 - targetIdx, 227
- non_leap
 - asso_data, 150
- num
 - wifi_scan_list_t, 253
- num_pmkid
 - _wpa_ie_data, 148
 - wifi_wpa_ie_data_t, 256
- number
 - wifi_event_sta_scan_done_t, 246
- offset
 - LE_GATT_MSG_ACCESS_READ_IND_T, 183
 - LE_GATT_MSG_ACCESS_WRITE_IND_T, 184
 - LE_GATT_MSG_CHARACTERISTIC_VAL_IND_T, 188
- own_addr_type
 - LE_GAP_ADVERTISING_PARAM_T, 179
 - LE_GAP_SCAN_PARAM_T, 181
- pAttr
 - LE_GATT_SERVICE_T, 214
- pFCInfo
 - auto_connect_cfg_t, 154
- pParam
 - T_RfEvt, 233
- PRIMARY_SERVICE_DECL_UUID128
 - BLE GATT APIs, 52
- PRIMARY_SERVICE_DECL_UUID16
 - BLE GATT APIs, 52
- pScanInfo
 - scan_report_t, 232
- pUuid
 - LE_GATT_ATTR_T, 182
- pVal
 - LE_GATT_ATTR_T, 182
 - LE_GATT_MSG_ACCESS_WRITE_IND_T, 185
- pairwise_cipher
 - _wpa_ie_data, 148
 - asso_data, 150
 - wifi_scan_info_t, 251
 - wifi_wpa_ie_data_t, 257
- param
 - event_msg_t, 156
- passive
 - wifi_scan_time_t, 253
- passkey
 - LE_SMP_MSG_PASSKEY_DISPLAY_IND_T, 218
- passphrase
 - asso_data, 150
 - auto_conn_info_t, 152
 - mw_wifi_auto_connect_ap_info_t, 224
 - wifi_auto_connect_info_t, 240
- password
 - wifi_ap_config_t, 237
 - wifi_sta_config_t, 254
- password_length
 - wifi_ap_config_t, 237
 - wifi_sta_config_t, 254
- peer_addr
 - LE_CM_CONNECTION_COMPLETE_IND_T, 158
 - LE_CM_MSG_DIRECT_ADV_REPORT_IND_T, 164
 - LE_GAP_ADVERTISING_PARAM_T, 179
- peer_addr_type
 - LE_CM_CONNECTION_COMPLETE_IND_T, 159
 - LE_CM_MSG_DIRECT_ADV_REPORT_IND_T, 164
 - LE_GAP_ADVERTISING_PARAM_T, 179
- peer_id_addr
 - LE_SMP_MSG_PAIRING_COMPLETE_IND_T, 217
- permit
 - LE_GATT_ATTR_T, 182
- pmkid
 - _wpa_ie_data, 148
 - wifi_wpa_ie_data_t, 257
- property
 - LE_GATT_MSG_CHARACTERISTIC_DECL_IND_T, 187
- proto
 - _wpa_ie_data, 148
 - asso_data, 150
 - wifi_wpa_ie_data_t, 257

- prvData
 - wifi_cmd_t, 241
 - wifi_evt_t, 247
- psk
 - asso_data, 150
 - auto_conn_info_t, 152
 - mw_wifi_auto_connect_ap_info_t, 224
 - wifi_auto_connect_info_t, 240
- psk_set
 - asso_data, 150
- ptScanReport
 - S_WIFI_MLME_SCAN_CFG, 228
- pwr_level
 - LE_CM_MSG_READ_ADV_TX_POWER_CFM←_T, 169
- rand
 - LE_CM_MSG_LTK_REQ_IND_T, 168
 - LE_SMP_SC_OOB_DATA_T, 221
- rear
 - auto_connect_cfg_t, 154
 - MwFimAutoConnectCFG_t, 227
- reason
 - LE_CM_MSG_DISCONNECT_COMPLETE_IN←D_T, 165
 - wifi_event_sta_disconnected_t, 245
- reserved
 - wifi_cmd_t, 241
- retryCount
 - auto_connect_cfg_t, 154
- role
 - LE_CM_CONNECTION_COMPLETE_IND_T, 159
- rsn_ie
 - auto_conn_info_t, 152
 - mw_wifi_auto_connect_ap_info_t, 224
 - scan_info_t, 231
 - wifi_auto_connect_info_t, 240
- rsni
 - auto_conn_info_t, 153
 - LE_CM_MSG_ADVERTISE_REPORT_IND←T, 160
 - LE_CM_MSG_DIRECT_ADV_REPORT_IND_T, 164
 - LE_CM_MSG_READ_RSSI_CFM_T, 172
 - mw_wifi_auto_connect_ap_info_t, 225
 - scan_info_t, 231
 - wifi_auto_connect_info_t, 240
 - wifi_fast_scan_threshold_t, 248
 - wifi_scan_info_t, 252
- rx_eapol_data, 227
 - frame_buffer, 227
 - frame_length, 227
- rx_phy
 - LE_CM_MSG_READ_PHY_CFM_T, 171
- S_WIFI_MLME_SCAN_CFG, 228
 - ptScanReport, 228
 - tScanType, 228
 - u32ActiveScanDur, 228
 - u32PassiveScanDur, 228
 - u8Channel, 229
 - u8MaxScanApNum, 229
 - u8ResendCnt, 229
 - u8aBssid, 229
 - u8aSsid, 229
- SECONDARY_SERVICE_DECL_UUID128
 - BLE GATT APIs, 52
- SECONDARY_SERVICE_DECL_UUID16
 - BLE GATT APIs, 52
- saArgv
 - T_RfCmd, 233
- sc
 - LE_SMP_MSG_PAIRING_ACTION_IND_T, 216
 - LE_SMP_MSG_PAIRING_COMPLETE_IND_T, 217
 - LE_SMP_MSG_SLAVE_SECURITY_REQUEST←T_IND_T, 220
- scan_done
 - wifi_event_info_t, 243
- scan_id
 - wifi_event_sta_scan_done_t, 246
- scan_info_t, 229
 - ap_channel, 230
 - beacon_interval, 230
 - bssid, 230
 - capabilities, 230
 - dtim_prod, 230
 - free_ocpy, 231
 - latest_beacon_rx_time, 231
 - rsn_ie, 231
 - rssi, 231
 - ssid, 231
 - ssid_len, 231
 - supported_rates, 231
 - wpa_data, 231
 - wpa_ie, 232
- scan_method
 - wifi_sta_config_t, 255
- scan_report_t, 232
 - pScanInfo, 232
 - uScanApNum, 232
- scan_time
 - wifi_scan_config_t, 249
- scan_type
 - wifi_scan_config_t, 250
- show_hidden
 - wifi_scan_config_t, 250
- size
 - LE_CM_MSG_READ_RESOLVING_LIST_SIZE←_CFM_T, 171
 - LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM←_T, 173
- slave_latency
 - LE_CM_MSG_SIGNAL_UPDATE_REQ_T, 176
- sort_method
 - wifi_sta_config_t, 255
- ssid

- auto_conn_info_t, 153
- mw_wifi_auto_connect_ap_info_t, 225
- scan_info_t, 231
- wifi_ap_config_t, 238
- wifi_auto_connect_info_t, 240
- wifi_event_sta_connected_t, 244
- wifi_event_sta_disconnected_t, 245
- wifi_scan_config_t, 250
- wifi_scan_info_t, 252
- wifi_sta_config_t, 255
- ssid_hidden
 - wifi_ap_config_t, 238
- ssid_len
 - auto_conn_info_t, 153
 - mw_wifi_auto_connect_ap_info_t, 225
 - scan_info_t, 231
 - wifi_event_sta_connected_t, 244
 - wifi_event_sta_disconnected_t, 245
- ssid_length
 - wifi_ap_config_t, 238
 - wifi_scan_info_t, 252
 - wifi_sta_config_t, 255
- sta_config
 - wifi_config_t, 242
- start_hdl
 - LE_GATT_MSG_INCLUDE_SERVICE_INFO_I↔ND_T, 198
 - LE_GATT_MSG_SERVICE_INFO_IND_T, 208
- startHdl
 - LE_GATT_SERVICE_T, 214
- status
 - LE_CM_CONNECTION_COMPLETE_IND_T, 159
 - LE_CM_MSG_CONN_UPDATE_COMPLETE_I↔ND_T, 162
 - LE_CM_MSG_DISCONNECT_COMPLETE_I↔D_T, 165
 - LE_CM_MSG_ENCRYPTION_CHANGE_IND_T, 166
 - LE_CM_MSG_ENCRYPTION_REFRESH_IND_T, 167
 - LE_CM_MSG_INIT_COMPLETE_CFM_T, 167
 - LE_CM_MSG_READ_ADV_TX_POWER_CFM↔_T, 169
 - LE_CM_MSG_READ_BD_ADDR_CFM_T, 169
 - LE_CM_MSG_READ_CHANNEL_MAP_CFM_T, 170
 - LE_CM_MSG_READ_PHY_CFM_T, 171
 - LE_CM_MSG_READ_RESOLVING_LIST_SIZE↔_CFM_T, 171
 - LE_CM_MSG_READ_RSSI_CFM_T, 172
 - LE_CM_MSG_READ_TX_POWER_CFM_T, 173
 - LE_CM_MSG_READ_WHITE_LIST_SIZE_CFM↔_T, 174
 - LE_CM_MSG_SET_DATA_LENGTH_CFM_T, 174
 - LE_CM_MSG_SET_DISCONNECT_CFM_T, 175
 - LE_CM_MSG_SET_PHY_CFM_T, 175
 - LE_CM_REQ_STATUS_T, 177
 - LE_GATT_MSG_EXECUTE_WRITE_RELIABL↔E_CFM_T, 192
 - LE_GATT_MSG_FIND_ALL_CHAR_DESC_CF↔M_T, 193
 - LE_GATT_MSG_FIND_ALL_PRIMARY_SERVI↔CE_CFM_T, 194
 - LE_GATT_MSG_FIND_CHARACTERISTIC_CF↔M_T, 195
 - LE_GATT_MSG_FIND_INCLUDED_SERVICE↔CFM_T, 196
 - LE_GATT_MSG_FIND_PRIMARY_SERVICE_B↔Y_UUID_CFM_T, 197
 - LE_GATT_MSG_NOTIFY_CFM_T, 200
 - LE_GATT_MSG_PREPARE_WRITE_RELIABL↔E_CFM_T, 203
 - LE_GATT_MSG_READ_CHAR_VAL_BY_UUID↔_CFM_T, 204
 - LE_GATT_MSG_READ_CHARACTERISTIC_V↔ALUE_CFM_T, 205
 - LE_GATT_MSG_READ_LONG_CHAR_VAL_C↔FM_T, 206
 - LE_GATT_MSG_READ_MULTIPLE_CHAR_VA↔L_CFM_T, 207
 - LE_GATT_MSG_SIGNED_WRITE_CFM_T, 209
 - LE_GATT_MSG_WRITE_CHAR_VAL_RELIAB↔LE_CFM_T, 210
 - LE_GATT_MSG_WRITE_CHAR_VALUE_CFM↔_T, 211
 - LE_GATT_MSG_WRITE_LONG_CHAR_VALU↔E_CFM_T, 212
 - LE_GATT_MSG_WRITE_NO_RSP_CFM_T, 213
 - LE_SMP_MSG_ENCRYPTION_REFRESH_IND↔_T, 215
 - LE_SMP_MSG_PAIRING_COMPLETE_IND_T, 217
 - wifi_event_sta_scan_done_t, 247
- supervision_timeout
 - LE_CM_MSG_CONN_UPDATE_COMPLETE_I↔ND_T, 162
 - LE_GAP_CONN_PARAM_T, 180
- supervision_timeout
 - LE_CM_CONNECTION_COMPLETE_IND_T, 159
- supported_rates
 - auto_conn_info_t, 153
 - mw_wifi_auto_connect_ap_info_t, 225
 - scan_info_t, 231
 - wifi_auto_connect_info_t, 240
- sv_timeout
 - LE_CONN_PARA_T, 178
- sv_tmo
 - LE_CM_MSG_CONN_PARA_REQ_T, 161
- svc_id
 - LE_GATT_SERVICE_T, 214
- T_HOUR
 - BLE MSG APIs, 76
- T_MIN
 - BLE MSG APIs, 76
- T_RfCmd, 232

- iArgc, [233](#)
- saArgv, [233](#)
- u32Type, [233](#)
- T_RfEvt, [233](#)
 - pParam, [233](#)
 - u16RfMode, [234](#)
 - u16RxCnt, [234](#)
 - u16RxCrcOkCnt, [234](#)
 - u32Freq, [234](#)
 - u32Mode, [234](#)
 - u32RfChannel, [234](#)
 - u32Type, [234](#)
 - u8Freq, [234](#)
 - u8IpcEnable, [235](#)
 - u8Len, [235](#)
 - u8Pkt, [235](#)
 - u8Reserved, [235](#)
 - u8Status, [235](#)
 - u8Unicast, [235](#)
- T_SEC
 - BLE MSG APIs, [76](#)
- TASKHANDLER
 - BLE MSG APIs, [77](#)
- TASKPACK
 - BLE MSG APIs, [78](#)
- TASK
 - BLE MSG APIs, [77](#)
- tScanType
 - S_WIFI_MLME_SCAN_CFG, [228](#)
- targetIdx
 - auto_connect_cfg_t, [155](#)
 - MwFimAutoConnectCFG_t, [227](#)
- Task
 - BLE MSG APIs, [77](#)
- threshold
 - wifi_sta_config_t, [255](#)
- timeout_multiplier
 - LE_CM_MSG_SIGNAL_UPDATE_REQ_T, [176](#)
- tx_phy
 - LE_CM_MSG_READ_PHY_CFM_T, [171](#)
- tx_power
 - LE_CM_MSG_READ_TX_POWER_CFM_T, [173](#)
- type
 - LE_BT_ADDR_T, [157](#)
 - LE_GAP_ADVERTISING_PARAM_T, [179](#)
 - LE_GAP_SCAN_PARAM_T, [181](#)
- u16RfMode
 - T_RfEvt, [234](#)
- u16RxCnt
 - T_RfEvt, [234](#)
- u16RxCrcOkCnt
 - T_RfEvt, [234](#)
- u32ActiveScanDur
 - S_WIFI_MLME_SCAN_CFG, [228](#)
- u32Freq
 - T_RfEvt, [234](#)
- u32Mode
 - T_RfEvt, [234](#)
- u32PassiveScanDur
 - S_WIFI_MLME_SCAN_CFG, [228](#)
- u32RfChannel
 - T_RfEvt, [234](#)
- u32Type
 - T_RfCmd, [233](#)
 - T_RfEvt, [234](#)
- u8Channel
 - S_WIFI_MLME_SCAN_CFG, [229](#)
- u8Freq
 - T_RfEvt, [234](#)
- u8IpcEnable
 - T_RfEvt, [235](#)
- u8Len
 - T_RfEvt, [235](#)
- u8MaxScanApNum
 - S_WIFI_MLME_SCAN_CFG, [229](#)
- u8Pkt
 - T_RfEvt, [235](#)
- u8ResendCnt
 - S_WIFI_MLME_SCAN_CFG, [229](#)
- u8Reserved
 - T_RfEvt, [235](#)
- u8SkipDtimPeriods
 - mw_wifi_sta_info_t, [226](#)
- u8Status
 - T_RfEvt, [235](#)
- u8Unicast
 - T_RfEvt, [235](#)
- u8aBssid
 - S_WIFI_MLME_SCAN_CFG, [229](#)
- u8aSsid
 - S_WIFI_MLME_SCAN_CFG, [229](#)
- uFCAPNum
 - auto_connect_cfg_t, [155](#)
- uScanApNum
 - scan_report_t, [232](#)
- uuid
 - LE_GATT_MSG_CHAR_DESCRIPTOR_INFO_↔
IND_T, [186](#)
 - LE_GATT_MSG_CHARACTERISTIC_DECL_IN↔
FO_IND_T, [187](#)
 - LE_GATT_MSG_INCLUDE_SERVICE_INFO_I↔
ND_T, [198](#)
 - LE_GATT_MSG_SERVICE_INFO_IND_T, [208](#)
- val
 - LE_GATT_MSG_CHARACTERISTIC_VAL_IND↔
_T, [188](#)
 - LE_GATT_MSG_INDICATE_IND_T, [199](#)
 - LE_GATT_MSG_NOTIFY_IND_T, [201](#)
 - LE_GATT_MSG_READ_MULTIPLE_CHAR_VA↔
L_CFM_T, [207](#)
- val_hdl
 - LE_GATT_MSG_CHARACTERISTIC_DECL_IN↔
FO_IND_T, [187](#)
- WIFI APIs, [93](#)
 - WIFI_BEACON_INTERVAL_LENGTH, [94](#)

- WIFI_CAPABILITY_INFO_LENGTH, 94
- WIFI_LENGTH_802_11, 95
- WIFI_LENGTH_PASSPHRASE, 95
- WIFI_MAC_ADDRESS_LENGTH, 95
- WIFI_MAX_LENGTH_OF_SSID, 95
- WIFI_MAX_SCAN_AP_NUM, 95
- WIFI_MAX_SUPPORTED_RATES, 95
- wifi_ap_record_t, 96
- wifi_auto_connect_mode_e, 96
- wifi_event_notify_cb_t, 96
- wifi_event_process_handler, 96
- wifi_install_default_event_handlers, 97
- wifi_register_event_handler, 97
- WIFI Common APIs, 99
 - wifi_event_cb_t, 99
 - wifi_event_loop_init, 99
 - wifi_event_loop_send, 100
 - wifi_event_loop_set_cb, 100
 - wifi_event_process_handler, 101
- WIFI STA APIs, 102
 - WIFI_READY_TIME, 106
 - wifi_auto_connect_clear_ap_info, 113
 - wifi_auto_connect_clear_ap_info_api, 133
 - wifi_auto_connect_clear_ap_info_fp_t, 106
 - wifi_auto_connect_get_ap_info, 114
 - wifi_auto_connect_get_ap_info_api, 134
 - wifi_auto_connect_get_ap_info_fp_t, 106
 - wifi_auto_connect_get_ap_num, 114
 - wifi_auto_connect_get_ap_num_api, 134
 - wifi_auto_connect_get_ap_num_fp_t, 106
 - wifi_auto_connect_get_mode, 115
 - wifi_auto_connect_get_mode_api, 134
 - wifi_auto_connect_get_mode_fp_t, 106
 - wifi_auto_connect_init, 115
 - wifi_auto_connect_init_api, 134
 - wifi_auto_connect_init_fp_t, 106
 - wifi_auto_connect_reset, 115
 - wifi_auto_connect_reset_api, 134
 - wifi_auto_connect_reset_fp_t, 106
 - wifi_auto_connect_set_ap_num, 116
 - wifi_auto_connect_set_ap_num_api, 134
 - wifi_auto_connect_set_ap_num_fp_t, 107
 - wifi_auto_connect_set_mode, 116
 - wifi_auto_connect_set_mode_api, 134
 - wifi_auto_connect_set_mode_fp_t, 107
 - wifi_auto_connect_start, 117
 - wifi_auto_connect_start_api, 134
 - wifi_auto_connect_start_fp_t, 107
 - wifi_config_get_bandwidth, 117
 - wifi_config_get_bandwidth_api, 135
 - wifi_config_get_bandwidth_fp_t, 107
 - wifi_config_get_bssid, 118
 - wifi_config_get_bssid_api, 135
 - wifi_config_get_bssid_fp_t, 107
 - wifi_config_get_channel, 118
 - wifi_config_get_channel_api, 135
 - wifi_config_get_channel_fp_t, 107
 - wifi_config_get_dtim_interval, 119
 - wifi_config_get_dtim_interval_api, 135
 - wifi_config_get_dtim_interval_fp_t, 107
 - wifi_config_get_listen_interval, 119
 - wifi_config_get_listen_interval_api, 135
 - wifi_config_get_listen_interval_fp_t, 107
 - wifi_config_get_mac_address, 119
 - wifi_config_get_mac_address_api, 135
 - wifi_config_get_mac_address_fp_t, 108
 - wifi_config_get_opmode, 120
 - wifi_config_get_opmode_api, 135
 - wifi_config_get_opmode_fp_t, 108
 - wifi_config_get_skip_dtim, 120
 - wifi_config_get_ssid, 120
 - wifi_config_get_ssid_api, 135
 - wifi_config_get_ssid_fp_t, 108
 - wifi_config_set_bandwidth, 121
 - wifi_config_set_bandwidth_api, 136
 - wifi_config_set_bandwidth_fp_t, 108
 - wifi_config_set_bssid, 121
 - wifi_config_set_bssid_api, 136
 - wifi_config_set_bssid_fp_t, 108
 - wifi_config_set_channel, 121
 - wifi_config_set_channel_api, 136
 - wifi_config_set_channel_fp_t, 108
 - wifi_config_set_dtim_interval, 122
 - wifi_config_set_dtim_interval_api, 136
 - wifi_config_set_dtim_interval_fp_t, 108
 - wifi_config_set_listen_interval, 122
 - wifi_config_set_listen_interval_api, 136
 - wifi_config_set_listen_interval_fp_t, 108
 - wifi_config_set_mac_address, 122
 - wifi_config_set_mac_address_api, 136
 - wifi_config_set_mac_address_fp_t, 109
 - wifi_config_set_opmode, 123
 - wifi_config_set_opmode_api, 136
 - wifi_config_set_opmode_fp_t, 109
 - wifi_config_set_skip_dtim, 123
 - wifi_config_set_ssid, 123
 - wifi_config_set_ssid_api, 136
 - wifi_config_set_ssid_fp_t, 109
 - wifi_connection_connect, 124
 - wifi_connection_connect_api, 137
 - wifi_connection_connect_fp_t, 109
 - wifi_connection_disconnect_ap, 124
 - wifi_connection_disconnect_ap_api, 137
 - wifi_connection_disconnect_ap_fp_t, 109
 - wifi_connection_disconnect_sta, 125
 - wifi_connection_disconnect_sta_api, 137
 - wifi_connection_disconnect_sta_fp_t, 109
 - wifi_connection_get_rssi, 125
 - wifi_connection_get_rssi_api, 137
 - wifi_connection_get_rssi_fp_t, 109
 - wifi_connection_register_event_handler, 126
 - wifi_connection_register_event_handler_api, 137
 - wifi_connection_register_event_handler_fp_t, 110
 - wifi_connection_scan_start, 126
 - wifi_connection_scan_start_api, 137
 - wifi_connection_scan_start_fp_t, 110

- wifi_connection_unregister_event_handler, 126
- wifi_connection_unregister_event_handler_api, 137
- wifi_connection_unregister_event_handler_fp_t, 110
- wifi_convert_auth_mode, 127
- wifi_convert_auth_mode_api, 137
- wifi_convert_auth_mode_fp_t, 110
- wifi_deinit, 127
- wifi_deinit_api, 138
- wifi_deinit_fp_t, 110
- wifi_event_handler_t, 110
- wifi_fast_connect_get_mode, 127
- wifi_fast_connect_get_mode_api, 138
- wifi_fast_connect_get_mode_fp_t, 111
- wifi_fast_connect_set_mode, 128
- wifi_fast_connect_set_mode_api, 138
- wifi_fast_connect_set_mode_fp_t, 111
- wifi_fast_connect_start, 128
- wifi_fast_connect_start_api, 138
- wifi_fast_connect_start_fp_t, 111
- wifi_get_config, 129
- wifi_get_config_api, 138
- wifi_get_config_fp_t, 111
- wifi_init, 129
- wifi_init_api, 138
- wifi_init_complete_cb_t, 111
- wifi_init_fp_t, 112
- wifi_result_t, 112
- wifi_scan_get_ap_list, 130
- wifi_scan_get_ap_list_api, 138
- wifi_scan_get_ap_list_fp_t, 112
- wifi_scan_get_ap_num, 130
- wifi_scan_get_ap_num_api, 138
- wifi_scan_get_ap_num_fp_t, 112
- wifi_scan_get_ap_records, 131
- wifi_scan_get_ap_records_api, 139
- wifi_scan_get_ap_records_fp_t, 112
- wifi_scan_scan_stop, 131
- wifi_scan_start, 131
- wifi_scan_start_api, 139
- wifi_scan_start_fp_t, 112
- wifi_scan_stop, 139
- wifi_scan_stop_fp_t, 112
- wifi_set_config, 132
- wifi_set_config_api, 139
- wifi_set_config_fp_t, 113
- wifi_sta_get_ap_info, 132
- wifi_sta_get_ap_info_api, 139
- wifi_sta_get_ap_info_fp_t, 113
- wifi_start, 133
- wifi_start_api, 139
- wifi_start_fp_t, 113
- wifi_stop, 133
- wifi_stop_api, 139
- wifi_stop_fp_t, 113
- WIFI_BEACON_INTERVAL_LENGTH
 - WIFI APs, 94
- WIFI_CAPABILITY_INFO_LENGTH
 - WIFI APs, 94
- WIFI_LENGTH_802_11
 - WIFI APs, 95
- WIFI_LENGTH_PASSPHRASE
 - WIFI APs, 95
- WIFI_MAC_ADDRESS_LENGTH
 - WIFI APs, 95
- WIFI_MAX_LENGTH_OF_SSID
 - WIFI APs, 95
- WIFI_MAX_SCAN_AP_NUM
 - WIFI APs, 95
- WIFI_MAX_SUPPORTED_RATES
 - WIFI APs, 95
- WIFI_READY_TIME
 - WIFI STA APs, 106
- wifi_active_scan_time_t, 235
 - max, 236
 - min, 236
- wifi_ap_config_t, 236
 - auth_mode, 237
 - beacon_interval, 237
 - channel, 237
 - encrypt_type, 237
 - max_connection, 237
 - password, 237
 - password_length, 237
 - ssid, 238
 - ssid_hidden, 238
 - ssid_length, 238
- wifi_ap_record_t
 - WIFI APs, 96
- wifi_auth_mode_t
 - Enumeration, 140
- wifi_auto_connect_clear_ap_info
 - WIFI STA APs, 113
- wifi_auto_connect_clear_ap_info_api
 - WIFI STA APs, 133
- wifi_auto_connect_clear_ap_info_fp_t
 - WIFI STA APs, 106
- wifi_auto_connect_get_ap_info
 - WIFI STA APs, 114
- wifi_auto_connect_get_ap_info_api
 - WIFI STA APs, 134
- wifi_auto_connect_get_ap_info_fp_t
 - WIFI STA APs, 106
- wifi_auto_connect_get_ap_num
 - WIFI STA APs, 114
- wifi_auto_connect_get_ap_num_api
 - WIFI STA APs, 134
- wifi_auto_connect_get_ap_num_fp_t
 - WIFI STA APs, 106
- wifi_auto_connect_get_mode
 - WIFI STA APs, 115
- wifi_auto_connect_get_mode_api
 - WIFI STA APs, 134
- wifi_auto_connect_get_mode_fp_t
 - WIFI STA APs, 106

- wifi_auto_connect_info_t, 238
 - ap_channel, 239
 - beacon_interval, 239
 - bssid, 239
 - capabilities, 239
 - dtim_prod, 239
 - fast_connect, 239
 - hid_ssid, 239
 - latest_beacon_rx_time, 240
 - passphrase, 240
 - psk, 240
 - rsn_ie, 240
 - rsni, 240
 - ssid, 240
 - supported_rates, 240
 - wpa_data, 240
 - wpa_ie, 241
- wifi_auto_connect_init
 - WIFI STA APIs, 115
- wifi_auto_connect_init_api
 - WIFI STA APIs, 134
- wifi_auto_connect_init_fp_t
 - WIFI STA APIs, 106
- wifi_auto_connect_reset
 - WIFI STA APIs, 115
- wifi_auto_connect_reset_api
 - WIFI STA APIs, 134
- wifi_auto_connect_reset_fp_t
 - WIFI STA APIs, 106
- wifi_auto_connect_set_ap_num
 - WIFI STA APIs, 116
- wifi_auto_connect_set_ap_num_api
 - WIFI STA APIs, 134
- wifi_auto_connect_set_ap_num_fp_t
 - WIFI STA APIs, 107
- wifi_auto_connect_set_mode
 - WIFI STA APIs, 116
- wifi_auto_connect_set_mode_api
 - WIFI STA APIs, 134
- wifi_auto_connect_set_mode_fp_t
 - WIFI STA APIs, 107
- wifi_auto_connect_start
 - WIFI STA APIs, 117
- wifi_auto_connect_start_api
 - WIFI STA APIs, 134
- wifi_auto_connect_start_fp_t
 - WIFI STA APIs, 107
- wifi_auto_connet_mode_e
 - WIFI APIs, 96
- wifi_bandwidth_t
 - Enumeration, 142
- wifi_cipher_type_t
 - Enumeration, 142
- wifi_cmd_t, 241
 - arg1, 241
 - cmd_type, 241
 - prvData, 241
 - reserved, 241
- wifi_config_get_bandwidth
 - WIFI STA APIs, 117
- wifi_config_get_bandwidth_api
 - WIFI STA APIs, 135
- wifi_config_get_bandwidth_fp_t
 - WIFI STA APIs, 107
- wifi_config_get_bssid
 - WIFI STA APIs, 118
- wifi_config_get_bssid_api
 - WIFI STA APIs, 135
- wifi_config_get_bssid_fp_t
 - WIFI STA APIs, 107
- wifi_config_get_channel
 - WIFI STA APIs, 118
- wifi_config_get_channel_api
 - WIFI STA APIs, 135
- wifi_config_get_channel_fp_t
 - WIFI STA APIs, 107
- wifi_config_get_dtim_interval
 - WIFI STA APIs, 119
- wifi_config_get_dtim_interval_api
 - WIFI STA APIs, 135
- wifi_config_get_dtim_interval_fp_t
 - WIFI STA APIs, 107
- wifi_config_get_listen_interval
 - WIFI STA APIs, 119
- wifi_config_get_listen_interval_api
 - WIFI STA APIs, 135
- wifi_config_get_listen_interval_fp_t
 - WIFI STA APIs, 107
- wifi_config_get_mac_address
 - WIFI STA APIs, 119
- wifi_config_get_mac_address_api
 - WIFI STA APIs, 135
- wifi_config_get_mac_address_fp_t
 - WIFI STA APIs, 108
- wifi_config_get_opmode
 - WIFI STA APIs, 120
- wifi_config_get_opmode_api
 - WIFI STA APIs, 135
- wifi_config_get_opmode_fp_t
 - WIFI STA APIs, 108
- wifi_config_get_skip_dtim
 - WIFI STA APIs, 120
- wifi_config_get_ssid
 - WIFI STA APIs, 120
- wifi_config_get_ssid_api
 - WIFI STA APIs, 135
- wifi_config_get_ssid_fp_t
 - WIFI STA APIs, 108
- wifi_config_set_bandwidth
 - WIFI STA APIs, 121
- wifi_config_set_bandwidth_api
 - WIFI STA APIs, 136
- wifi_config_set_bandwidth_fp_t
 - WIFI STA APIs, 108
- wifi_config_set_bssid
 - WIFI STA APIs, 121

- wifi_config_set_bssid_api
 - WIFI STA APIs, [136](#)
- wifi_config_set_bssid_fp_t
 - WIFI STA APIs, [108](#)
- wifi_config_set_channel
 - WIFI STA APIs, [121](#)
- wifi_config_set_channel_api
 - WIFI STA APIs, [136](#)
- wifi_config_set_channel_fp_t
 - WIFI STA APIs, [108](#)
- wifi_config_set_dtim_interval
 - WIFI STA APIs, [122](#)
- wifi_config_set_dtim_interval_api
 - WIFI STA APIs, [136](#)
- wifi_config_set_dtim_interval_fp_t
 - WIFI STA APIs, [108](#)
- wifi_config_set_listen_interval
 - WIFI STA APIs, [122](#)
- wifi_config_set_listen_interval_api
 - WIFI STA APIs, [136](#)
- wifi_config_set_listen_interval_fp_t
 - WIFI STA APIs, [108](#)
- wifi_config_set_mac_address
 - WIFI STA APIs, [122](#)
- wifi_config_set_mac_address_api
 - WIFI STA APIs, [136](#)
- wifi_config_set_mac_address_fp_t
 - WIFI STA APIs, [109](#)
- wifi_config_set_opmode
 - WIFI STA APIs, [123](#)
- wifi_config_set_opmode_api
 - WIFI STA APIs, [136](#)
- wifi_config_set_opmode_fp_t
 - WIFI STA APIs, [109](#)
- wifi_config_set_skip_dtim
 - WIFI STA APIs, [123](#)
- wifi_config_set_ssid
 - WIFI STA APIs, [123](#)
- wifi_config_set_ssid_api
 - WIFI STA APIs, [136](#)
- wifi_config_set_ssid_fp_t
 - WIFI STA APIs, [109](#)
- wifi_config_t, [242](#)
 - ap_config, [242](#)
 - sta_config, [242](#)
- wifi_connection_connect
 - WIFI STA APIs, [124](#)
- wifi_connection_connect_api
 - WIFI STA APIs, [137](#)
- wifi_connection_connect_fp_t
 - WIFI STA APIs, [109](#)
- wifi_connection_disconnect_ap
 - WIFI STA APIs, [124](#)
- wifi_connection_disconnect_ap_api
 - WIFI STA APIs, [137](#)
- wifi_connection_disconnect_ap_fp_t
 - WIFI STA APIs, [109](#)
- wifi_connection_disconnect_sta
 - WIFI STA APIs, [125](#)
- wifi_connection_disconnect_sta_api
 - WIFI STA APIs, [137](#)
- wifi_connection_disconnect_sta_fp_t
 - WIFI STA APIs, [109](#)
- wifi_connection_get_rssi
 - WIFI STA APIs, [125](#)
- wifi_connection_get_rssi_api
 - WIFI STA APIs, [137](#)
- wifi_connection_get_rssi_fp_t
 - WIFI STA APIs, [109](#)
- wifi_connection_register_event_handler
 - WIFI STA APIs, [126](#)
- wifi_connection_register_event_handler_api
 - WIFI STA APIs, [137](#)
- wifi_connection_register_event_handler_fp_t
 - WIFI STA APIs, [110](#)
- wifi_connection_scan_start
 - WIFI STA APIs, [126](#)
- wifi_connection_scan_start_api
 - WIFI STA APIs, [137](#)
- wifi_connection_scan_start_fp_t
 - WIFI STA APIs, [110](#)
- wifi_connection_unregister_event_handler
 - WIFI STA APIs, [126](#)
- wifi_connection_unregister_event_handler_api
 - WIFI STA APIs, [137](#)
- wifi_connection_unregister_event_handler_fp_t
 - WIFI STA APIs, [110](#)
- wifi_convert_auth_mode
 - WIFI STA APIs, [127](#)
- wifi_convert_auth_mode_api
 - WIFI STA APIs, [137](#)
- wifi_convert_auth_mode_fp_t
 - WIFI STA APIs, [110](#)
- wifi_deinit
 - WIFI STA APIs, [127](#)
- wifi_deinit_api
 - WIFI STA APIs, [138](#)
- wifi_deinit_fp_t
 - WIFI STA APIs, [110](#)
- wifi_event_cb_t
 - WIFI Common APIs, [99](#)
- wifi_event_handler_t
 - WIFI STA APIs, [110](#)
- wifi_event_info_t, [242](#)
 - connected, [243](#)
 - disconnected, [243](#)
 - got_ip, [243](#)
 - scan_done, [243](#)
- wifi_event_loop_init
 - WIFI Common APIs, [99](#)
- wifi_event_loop_send
 - WIFI Common APIs, [100](#)
- wifi_event_loop_set_cb
 - WIFI Common APIs, [100](#)
- wifi_event_notify_cb_t
 - WIFI APIs, [96](#)

- wifi_event_process_handler
 - WIFI APIs, 96
 - WIFI Common APIs, 101
- wifi_event_sta_connected_t, 243
 - authmode, 244
 - bssid, 244
 - channel, 244
 - ssid, 244
 - ssid_len, 244
- wifi_event_sta_disconnected_t, 245
 - bssid, 245
 - reason, 245
 - ssid, 245
 - ssid_len, 245
- wifi_event_sta_got_ip_t, 246
 - ip_changed, 246
- wifi_event_sta_scan_done_t, 246
 - number, 246
 - scan_id, 246
 - status, 247
- wifi_event_t
 - Enumeration, 142
- wifi_evt_t, 247
 - evt_type, 247
 - privData, 247
- wifi_fast_connect_get_mode
 - WIFI STA APIs, 127
- wifi_fast_connect_get_mode_api
 - WIFI STA APIs, 138
- wifi_fast_connect_get_mode_fp_t
 - WIFI STA APIs, 111
- wifi_fast_connect_set_mode
 - WIFI STA APIs, 128
- wifi_fast_connect_set_mode_api
 - WIFI STA APIs, 138
- wifi_fast_connect_set_mode_fp_t
 - WIFI STA APIs, 111
- wifi_fast_connect_start
 - WIFI STA APIs, 128
- wifi_fast_connect_start_api
 - WIFI STA APIs, 138
- wifi_fast_connect_start_fp_t
 - WIFI STA APIs, 111
- wifi_fast_scan_threshold_t, 247
 - authmode, 248
 - rsi, 248
- wifi_get_config
 - WIFI STA APIs, 129
- wifi_get_config_api
 - WIFI STA APIs, 138
- wifi_get_config_fp_t
 - WIFI STA APIs, 111
- wifi_init
 - WIFI STA APIs, 129
- wifi_init_api
 - WIFI STA APIs, 138
- wifi_init_complete_cb_t
 - WIFI STA APIs, 111
- wifi_init_config_t, 248
 - event_handler, 248
 - magic, 249
- wifi_init_fp_t
 - WIFI STA APIs, 112
- wifi_install_default_event_handlers
 - WIFI APIs, 97
- wifi_mode_t
 - Enumeration, 143
- wifi_reason_code_t
 - Enumeration, 143
- wifi_register_event_handler
 - WIFI APIs, 97
- wifi_result_t
 - WIFI STA APIs, 112
- wifi_scan_config_t, 249
 - bssid, 249
 - channel, 249
 - scan_time, 249
 - scan_type, 250
 - show_hidden, 250
 - ssid, 250
- wifi_scan_get_ap_list
 - WIFI STA APIs, 130
- wifi_scan_get_ap_list_api
 - WIFI STA APIs, 138
- wifi_scan_get_ap_list_fp_t
 - WIFI STA APIs, 112
- wifi_scan_get_ap_num
 - WIFI STA APIs, 130
- wifi_scan_get_ap_num_api
 - WIFI STA APIs, 138
- wifi_scan_get_ap_num_fp_t
 - WIFI STA APIs, 112
- wifi_scan_get_ap_records
 - WIFI STA APIs, 131
- wifi_scan_get_ap_records_api
 - WIFI STA APIs, 139
- wifi_scan_get_ap_records_fp_t
 - WIFI STA APIs, 112
- wifi_scan_info_t, 250
 - auth_mode, 251
 - beacon_interval, 251
 - bssid, 251
 - capability_info, 251
 - channel, 251
 - dtim_period, 251
 - group_cipher, 251
 - pairwise_cipher, 251
 - rsi, 252
 - ssid, 252
 - ssid_length, 252
- wifi_scan_list_t, 252
 - ap_record, 252
 - num, 253
- wifi_scan_method_t
 - Enumeration, 144
- wifi_scan_scan_stop

- WIFI STA APIs, 131
- wifi_scan_start
 - WIFI STA APIs, 131
- wifi_scan_start_api
 - WIFI STA APIs, 139
- wifi_scan_start_fp_t
 - WIFI STA APIs, 112
- wifi_scan_stop_api
 - WIFI STA APIs, 139
- wifi_scan_stop_fp_t
 - WIFI STA APIs, 112
- wifi_scan_time_t, 253
 - active, 253
 - passive, 253
- wifi_scan_type_t
 - Enumeration, 144
- wifi_set_config
 - WIFI STA APIs, 132
- wifi_set_config_api
 - WIFI STA APIs, 139
- wifi_set_config_fp_t
 - WIFI STA APIs, 113
- wifi_sort_method_t
 - Enumeration, 145
- wifi_sta_config_t, 254
 - bssid, 254
 - bssid_present, 254
 - password, 254
 - password_length, 254
 - scan_method, 255
 - sort_method, 255
 - ssid, 255
 - ssid_length, 255
 - threshold, 255
- wifi_sta_get_ap_info
 - WIFI STA APIs, 132
- wifi_sta_get_ap_info_api
 - WIFI STA APIs, 139
- wifi_sta_get_ap_info_fp_t
 - WIFI STA APIs, 113
- wifi_start
 - WIFI STA APIs, 133
- wifi_start_api
 - WIFI STA APIs, 139
- wifi_start_fp_t
 - WIFI STA APIs, 113
- wifi_stop
 - WIFI STA APIs, 133
- wifi_stop_api
 - WIFI STA APIs, 139
- wifi_stop_fp_t
 - WIFI STA APIs, 113
- wifi_wpa_ie_data_t, 255
 - capabilities, 256
 - group_cipher, 256
 - key_mgmt, 256
 - mgmt_group_cipher, 256
 - num_pmkid, 256
 - pairwise_cipher, 257
 - pmkid, 257
 - proto, 257
- window
 - LE_GAP_SCAN_PARAM_T, 181
- wpa_data
 - auto_conn_info_t, 153
 - mw_wifi_auto_connect_ap_info_t, 225
 - scan_info_t, 231
 - wifi_auto_connect_info_t, 240
- wpa_ie
 - auto_conn_info_t, 153
 - mw_wifi_auto_connect_ap_info_t, 225
 - scan_info_t, 232
 - wifi_auto_connect_info_t, 241