



**BLE スタック
ユーザーズガイド**

第 6 版 イー・フォース株式会社

改版記録

第 1 版 新規作成

第 2 版で改訂された項目

ページ／章	内容
7／2	・誤記を修正しました。
87／6	・タスク:ID_TSK_BLE_HCI_SDIO_WR のスタックサイズを変更しました。
89～94／7	・サンプルプログラムの使用方法について、説明を追加しました。

第 3 版で改訂された項目

ページ／章	内容
7／2	<ul style="list-style-type: none"> ・カレントフォルダの名称を変更。 (変更前)BT → (変更後)BLE ・ハードウェア固有のフォルダの名称を変更。 (変更前)marvell → (変更後)88W8887 ・以下のファイルを削除。 uC3¥BLE¥88W8887¥bt_marvell.c uC3¥BLE¥88W8887¥bt_marvell.h uC3¥BLE¥sample¥ble_srv_uart.h ・ファイル名の誤記訂正。 (誤)bel_hci_drv_sdio.c → (正)ble_hci_drv_sdio.c
16／5	<ul style="list-style-type: none"> ・デバイス API に ble_ext を追加。 ・プロファイル API の ble_chg_gatt_srv を削除。 ・L2CAP API の l2cap_cmd_reject を削除。 ・GATT プロシージャ API の誤記訂正。 (誤)ble_disc_all_pri_src → (正)ble_disc_all_pri_svc (誤)ble_disc_all_pri_src_by_uuid → (正)ble_disc_all_pri_svc_by_uuid (誤)ble_disc_inc_src → (正)ble_disc_inc_svc
21／5.1	・ble_ext(BLE スタック終了)を追加。
22／5.1	・ble_get_dev_addr、ble_add_whitelist、ble_rmv_whitelist の書式誤記訂正。 (誤)T_BEL_ADDR → (正)T_BLE_ADDR
26／5.2	・ble_cfg_adv の書式およびパラメータ誤記訂正。 (誤)T_BEL_ADV_CFG → (正)T_BLE_ADV_CFG
67／5.8	・ble_disc_all_char_by_uuid のパラメータ誤記訂正。 (誤)T_BEL_UUID → (正)T_BLE_UUID
70／5.8	・ble_read_long_char の API 名の誤記訂正。また、パラメータ offset を追加。
76／5.8	・ble_read_long_char_desc の API 名の誤記訂正。また、パラメータ offset を追加。
81～83／5.9	<ul style="list-style-type: none"> ・BLE_APP_EVG_DEV イベントの誤記訂正。 (誤)BLE_APP_ADV_DEV → (正)BLE_APP_ADV_EVT ・T_BLE_LE_ADV、T_BLE_LE_DIRECT_ADV パラメータの rssi を UB→B 型に変更。 ・T_BLE_LE_CON_COMP、T_BLE_LE_DIS_COMP パラメータのフィールドに status を追加。 ・T_BLE_LE_DIS_COMP パラメータの誤記訂正。 (誤)T_BLE_DIS_COMP パラメータ → (正)T_BLE_LE_DIS_COMP パラメータ
92～93／7.1	<ul style="list-style-type: none"> ・ble ini、ble ext のコマンド追加。 ・ble cfm → ble nfy コマンドに名称変更。

第 4 版で改訂された項目

ページ／章	内容
8～9／2	<ul style="list-style-type: none"> hw フォルダを追加して、88W8887 フォルダの記載削除。 ble_smp.h、ble_smp.c の注釈削除。
10～11／3	<ul style="list-style-type: none"> SMP (Security Manager Protocol) の記載追加。
17／5	<ul style="list-style-type: none"> L2CAP API の名称変更。 (変更前) l2cap_cmd_con_upd_req l2cap_cmd_con_upd_res (変更後) ble_l2cap_cmd_con_upd_req ble_l2cap_cmd_con_upd_res
48～49／5.6	<ul style="list-style-type: none"> L2CAP API の名称変更。 T_BLE_L2CAP_CON_UPD_REQ 構造体の名称変更。
75,79／5.8	<ul style="list-style-type: none"> ble_write_long_char にパラメータ offset を追加。 ble_write_long_char_desc にパラメータ offset を追加。
85～86／5.9	<ul style="list-style-type: none"> BLE_APP_EVG_GATT イベントの誤記訂正および記載漏れを追記。
88～89／6	<ul style="list-style-type: none"> BLE スタックのタスク構成変更に伴い、OS 資源を更新。

第 5 版で改訂された項目

ページ／章	内容
17～19／5	<ul style="list-style-type: none"> イニシエータ API の ble_upd_con_res を削除。 GATT サービス API に ble_add_char_desc_cus を追加。 GATT プロシージャ API の ble_read_char_desc を削除。 GATT プロシージャ API の ble_read_long_char_desc を削除。 GATT プロシージャ API の ble_write_char_desc を削除。 GATT プロシージャ API の ble_write_long_char_desc を削除。
61／5.7	<ul style="list-style-type: none"> ble_add_char_desc_cus (キャラクターリスティックカスタムディスクリプタ追加) を追加。
78,82／5.9	<ul style="list-style-type: none"> イベントグループに、BLE_APP_EVG_L2CAP を追加して、BLE_APP_EVG_UNKWOUN を削除。 BLE_APP_EVG_DEV イベントに、BLE_APP_ADV_TER_EVT の記載漏れを追記。 BLE_APP_EVG_L2CAP イベントの説明追加。

第 6 版で改訂された項目

ページ／章	内容
全体	<ul style="list-style-type: none"> ・「SM」と「SMP」が同じ意味で混在使用されているものについて、文脈に応じて適切な方を使用。 ・「接続」と「コネクション」が同じ意味で使用されているものについて、すべて「コネクション」に統一。 ・「発見」と「検出」が同じ意味で使用されているものについて、すべて「検出」に統一。 ・「コネクションタイムアウト」と「スーパービジョンタイムアウト」が同じ意味で使用されているものについて、すべて「スーパービジョンタイムアウト」に統一。 ・その他、誤字脱字を修正。
12～13／2	<ul style="list-style-type: none"> ・uC3/BLE/lib 以下の構成を変更。 ・ble_sec.c を追加。 ・ble_time.c を ble_tim.c に変更。 ・ble_strlib.c、ble_strlib.h を追加。
14～15／3	<ul style="list-style-type: none"> ・各プロトコルの説明を修正。 ・BLE スタックが SM に対応していない旨の記載を削除。
16～25／4	<ul style="list-style-type: none"> ・「BLE の基礎知識」項を追加。
26～36／5	<ul style="list-style-type: none"> ・「BLE アプリケーションプロファイル」項を追加。 ・旧版の「4.用語」の内容は本項に統合。
37～61／6	<ul style="list-style-type: none"> ・「機能概要」項を追加。
62～140／7	<ul style="list-style-type: none"> ・API の追加に伴い、API 一覧を更新。 ・各 API の説明に項番をアサイン。
68／7.1.3	<ul style="list-style-type: none"> ・デバイス API の ble_get_dev_addr の機能追加。ランダムスタティックアドレスの取得に対応。
69／7.1.4	<ul style="list-style-type: none"> ・デバイス API に、ble_set_dev_addr を追加。
70／7.1.5	<ul style="list-style-type: none"> ・デバイス API の ble_add_whitelist の説明を修正。
71／7.1.6	<ul style="list-style-type: none"> ・デバイス API の ble_rmv_whitelist の説明を修正。
72／7.1.7	<ul style="list-style-type: none"> ・デバイス API の ble_clr_whitelist の説明を修正。
73／7.1.8	<ul style="list-style-type: none"> ・デバイス API に、ble_gen_rnd を追加。
74～77／7.2.1	<ul style="list-style-type: none"> ・アドバタイズ API の ble_cfg_adv のパラメータ全般に関して、説明の追加と修正。
78／7.2.2	<ul style="list-style-type: none"> ・アドバタイズ API の ble_ena_adv のタイムアウトに関する説明を追加。
82～84／7.3.1	<ul style="list-style-type: none"> ・スキャン API の ble_ena_scan のパラメータ全般に関して、説明の追加と修正。
86／7.4.1	<ul style="list-style-type: none"> ・GAP、GATT プロファイル API の ble_set_gap_srv_device_name の引数に関する説明の修正。
87／7.4.2	<ul style="list-style-type: none"> ・GAP、GATT プロファイル API の ble_set_gap_srv_appearance の引数に関する説明の修正。
88～90／7.5.1	<ul style="list-style-type: none"> ・イニシエータ API の ble_cre_con のパラメータ全般に関して、説明の追加と修正。
94～96／7.5	<ul style="list-style-type: none"> ・イニシエータ API に、以下の API を追加。 ble_enc_con ble_enc_con_res ble_get_enc_state
97／7.6.1	<ul style="list-style-type: none"> ・L2CAP API の、ble_l2cap_cmd_con_upd_req に関する説明の追加と修正。
101～102／7.7.3	<ul style="list-style-type: none"> ・GATT サービス API の、ble_add_char の property パラメータに関する説明の追加。
103～104／7.7.4	<ul style="list-style-type: none"> ・GATT サービス API の、ble_add_char_val に渡すコールバック関数の仕様変更。
109～110／7.7.9	<ul style="list-style-type: none"> ・GATT サービス API の、ble_add_char_desc_app の、T_BLE_CHAR_APP_DESC パラメータの説明を修正。
114／7.8.1	<ul style="list-style-type: none"> ・GATT プロシージャ API の ble_exchange_mtu の解説を、より詳細な内容に変更。
129～131／7.9	<ul style="list-style-type: none"> ・プライバシーAPIを追加
132～140／7.10	<ul style="list-style-type: none"> ・ペアリング API を追加
141～146／7.11	<ul style="list-style-type: none"> ・イベントグループに BLE_APP_EVG_SM を追加 ・BLE_APP_EVG_DEV イベントグループに、BLE_APP_CON_ENC_REQ_EVT、BLE_APP_CON_ENC_CHG_EVT、BLE_APP_CON_ENC_KEY_REF_EVT を追

ページ／章	内容
	加。 ・ BLE_APP_EVG_L2CAP イベントグループから BLE_APP_GATT_SYS_ERR_EVT イベントを削除
147～149／7.12	・エラーコードに、BLE スタックが定義するマエラーコードクロの一覧を追加。
150／7.13	・「グローバル UUID 変数」項を追加
151／8	・各 OS 資源のパラメータに関する説明を追記。
151／8.1	・ SM 対応に伴い、タスク：ID_BLE_SM_TSK 及びイベントフラグ：ID_BLE_SM_FLG を追加。
153～161／9	・コンフィグレータの BLE スタックへの対応に伴い、「コンフィグレーション」項を追加。
N/A	「サンプルプログラム」項をチュートリアルガイドに移行。

目次

1 はじめに	11
2 ファイル構成	12
3 機能.....	14
3.1 HCI (HOST CONTROLLER INTERFACE)	15
3.2 L2CAP (LOGIC LINK CONTROL AND ADAPTATION PROTOCOL).....	15
3.3 ATT (ATTRIBUTE PROTOCOL)	15
3.4 GAP (GENERAL ACCESS PROFILE)	15
3.5 GATT (GENERAL ATTRIBUTE PROFILE).....	15
3.6 SM (SECURITY MANAGER).....	15
4 BLE の基礎知識.....	16
4.1 チャンネル.....	16
4.2 コネクション非確立状態とコネクション確立状態におけるデータ転送	16
4.3 デバイスアドレス	17
4.3.1. パブリックデバイスアドレス	17
4.3.2. スタティックデバイスアドレス	17
4.3.3. Non-Resolvable プライベートアドレス	18
4.3.4. Resolvable プライベートアドレス.....	18
4.4 デバイスの状態	19
4.4.1. スタンバイ状態.....	19
4.4.2. アドバタイジング状態.....	19
4.4.3. スキャンング状態	20
4.4.4. イニシエーティング状態	21
4.4.5. コネクション状態	21
4.5 コネクションパラメータ	22
4.5.1. コネクション間隔	22
4.5.2. スレーブレイテンシ	22
4.5.3. スーパービジョンタイムアウト	22
4.5.4. コネクションパラメータ更新.....	23
4.6 ホワイトリスト	24
4.7 セキュリティ	24
4.7.1. プライバシー	24
4.7.2. 暗号化	24
4.7.3. ペアリングとボンディング	25
5 BLE アプリケーションプロファイル.....	26

5.1 GATT プロファイル	26
5.1.1. アトリビュート	26
5.1.2. アトリビュート UUID	26
5.1.3. アトリビュート階層構造	27
5.1.4. サービス	27
5.1.5. インクルード宣言	28
5.1.6. キャラクタリスティック	29
5.1.7. アトリビュート検出	30
5.1.8. ATT MTU	31
5.1.9. ロングアトリビュート値	31
5.1.10. Notification と Indication	31
5.2 GAP プロファイル	32
5.2.1. デバイスロール	32
5.2.2. AD タイプ	33
5.2.3. 検出モードと手順	34
5.2.4. コネクションモードと手順	36
6 機能概要	37
6.1 イベントハンドラ	37
6.2 アイデンティティアドレス	37
6.3 ブロードキャスターロール	38
6.4 オブザーバーロール	39
6.5 ペリフェラルロール	40
6.6 GATT サーバ	42
6.6.1. GATT サービスの登録	42
6.6.2. アトリビュートの read / write	43
6.7 セントラルロール	46
6.7.1. ペリフェラルデバイスの検出	46
6.7.2. コネクション	47
6.8 GATT クライアント	48
6.8.1. サービス検出	48
6.8.2. キャラクタリスティック検出	49
6.8.3. キャラクタリスティックディスクリプタ検出	50
6.8.4. アトリビュートへの read / write	51
6.8.5. MTU 交換	52
6.9 プライバシー	53
6.9.1. Resolvable プライベートアドレス (RPA)	53
6.9.2. Non-Resolvable プライベートアドレス (Non-RPA)	55

6.1 0 暗号化	56
6.1 0.1. セントラル	56
6.1 0.2. ペリフェラル	57
6.1 1 ペアリング	58
6.1 1.1. セントラル	58
6.1 1.2. ペリフェラル	60
6.1 1.3. ペリフェラルデバイスからのセキュリティ要求	61
6.1 1.4. ボンディング	61
7 API	62
7.1 デバイス API	65
7.1.1. ble_ini (BLE スタック初期化)	65
7.1.2. ble_ext (BLE スタック終了)	67
7.1.3. ble_get_dev_addr (デバイスアドレス取得)	68
7.1.4. ble_set_dev_addr (ランダムスタティックデバイスアドレス設定)	69
7.1.5. ble_add_whitelist (ホワイトリスト追加)	70
7.1.6. ble_rmwhitelist (ホワイトリスト削除)	71
7.1.7. ble_clr_whitelist (ホワイトリスト全削除)	72
7.1.8. ble_gen_rnd (乱数生成)	73
7.2 アドバタイズ API	74
7.2.1. ble_cfg_adv (アドバタイズパラメータ設定)	74
7.2.2. ble_ena_adv (アドバタイズパケット送信開始)	78
7.2.3. ble_dis_adv (アドバタイズパケット送信終了)	79
7.2.4. ble_set_adv_dat (アドバタイズデータ設定)	80
7.2.5. ble_set_adv_scan_rsp_dat (スキャン応答データ設定)	81
7.3 スキャン API	82
7.3.1. ble_ena_scan (スキャン開始)	82
7.3.2. ble_dis_scan (スキャン終了)	85
7.4 GAP, GATT プロファイル API	86
7.4.1. ble_set_gap_srv_device_name (デバイス名設定)	86
7.4.2. ble_set_gap_srv_appearance (アピアランス設定)	87
7.5 イニシエータ API	88
7.5.1. ble_cre_con (コネクション確立)	88
7.5.2. ble_cls_con (コネクション切断)	91
7.5.3. ble_upd_con (コネクションパラメータ更新)	92
7.5.4. ble_enc_con (コネクション暗号化)	94
7.5.5. ble_enc_con_res (コネクション暗号化応答)	95
7.5.6. ble_get_enc_state (コネクション暗号化確認)	96

7.6 L2CAP API.....	97
7.6.1. ble_l2cap_cmd_con_upd_req (コネクションパラメータ更新要求コマンド).....	97
7.6.2. ble_l2cap_cmd_con_upd_res (コネクションパラメータ更新要求への応答コマンド).....	98
7.7 GATT サービス API.....	99
7.7.1. ble_add_srv (サービス追加).....	99
7.7.2. ble_add_inc_srv (インクルードサービス定義追加).....	100
7.7.3. ble_add_char (キャラクターリスティック定義追加).....	101
7.7.4. ble_add_char_val (キャラクターリスティック値宣言追加).....	103
7.7.5. ble_add_char_desc_ext (キャラクターリスティック拡張プロパティディスクリプタ追加).....	105
7.7.6. ble_add_char_desc_user (キャラクターリスティックユーザーディスクリプタ追加).....	106
7.7.7. ble_add_char_desc_cli (クライアントキャラクターリスティック構成ディスクリプタ追加).....	107
7.7.8. ble_add_char_desc_ser (サーバキャラクターリスティック構成ディスクリプタ追加).....	108
7.7.9. ble_add_char_desc_app (キャラクターリスティック表示フォーマットディスクリプタ追加).....	109
7.7.10. ble_add_char_desc_agg (キャラクターリスティックアグリゲートフォーマットディスクリプタ追加).....	111
7.7.11. ble_add_char_desc_cus (キャラクターリスティックカスタムディスクリプタ追加).....	112
7.7.12. ble_reg_srv (サービス定義登録).....	113
7.8 GATT プロシージャ API.....	114
7.8.1. ble_exchange_mtu (MTU 交換).....	114
7.8.2. ble_disc_all_pri_svc (プライマリサービス検出).....	115
7.8.3. ble_disc_all_pri_svc_by_uuid (特定プライマリサービス検出).....	116
7.8.4. ble_disc_inc_svc (インクルードサービス検出).....	117
7.8.5. ble_disc_all_char (キャラクターリスティック検出).....	118
7.8.6. ble_disc_all_char_by_uuid (特定キャラクターリスティック検出).....	119
7.8.7. ble_disc_all_char_desc (ディスクリプタ検出).....	120
7.8.8. ble_read_char (キャラクターリスティック値読み取り).....	121
7.8.9. ble_read_long_char (キャラクターリスティック値読み取り(大きいサイズ)).....	122
7.8.10. ble_read_char_by_uuid (キャラクターリスティック値読み取り(UUID 指定)).....	123
7.8.11. ble_write_char (キャラクターリスティック値書き込み).....	124
7.8.12. ble_write_cmd_char (キャラクターリスティック値書き込み(応答なし)).....	125
7.8.13. ble_write_long_char (キャラクターリスティック値書き込み(大きいサイズ)).....	126

7.8.1 4. ble_notify (キャラクタースティック値 notify).....	127
7.8.1 5. ble_indicate (キャラクタースティック値 indicate).....	128
7.9 プライバシーAPI.....	129
7.9.1. ble_gen_irk (IRK 生成).....	129
7.9.2. ble_cfg_resolve (Resolving リストの設定).....	130
7.9.3. ble_chk_addr_resolvable (受信 RPA が既知デバイスのものか確認)	131
7.10 ペ어링 API.....	132
7.10.1. ble_pair_master_req (セントラルデバイスからのペ어링開始要求).....	132
7.10.2. ble_pair_slave_res (ペリフェラルデバイスのペ어링応答)	135
7.10.3. ble_pair_slave_req (ペリフェラルデバイスからのペ어링開始要求)	136
7.10.4. ble_pair_user_input (ペ어링入力方法イベントに対する応答).....	137
7.10.5. ble_gen_oob_dat (OOB データ生成).....	139
7.10.6. ble_get_bond_by_con (ボンディング情報の検索).....	140
7.11 コールバックイベント	141
7.12 エラーコード	147
7.12.1. API 戻り値	147
7.12.2. BLE エラーコード	147
7.12.3. BLE ATT エラーコード.....	149
7.12.4. BLE ペ어링エラーコード.....	149
7.13 グローバル UUID 変数.....	150
8 OS 資源	151
8.1 タスク	151
8.2 セマフォ	151
8.3 イベントフラグ	151
8.4 メールボックス	151
8.5 固定長メモリプール	152
8.6 周期ハンドラ	152
9 コンフィグレーション	153
9.1 コンフィグレータの起動.....	153
9.2 BLE スタックの設定.....	157
9.3 プロジェクトファイルの保存.....	160
9.4 ソース生成.....	161

1 はじめに

BLE スタックは、弊社 リアルタイムオペレーティングシステム μ C3(マイクロ・シー・キューブ)向けに実装された Bluetooth Low Energy プロトコルスタックです。BLE スタックは、Bluetooth Core Specification 4.2 で規定された仕様に従って実装されています。

本書は BLE スタックの API、OS リソース、ファイル構成、サンプルプログラムの動作について説明します。

2 ファイル構成

BLE スタックのファイル構成は以下の通りです。

表 2-1 ファイル一覧

uC3¥BLE¥	
hw	ハードウェア固有のソースコード(HCI)
inc	BLE スタックヘッダファイル
lib	BLE スタックライブラリ
sample	BLE サンプルアプリケーションのソースコード
src	BLE スタックソースコード

inc	
ble_evt.h	BLE イベント関連ヘッダファイル
ble_gatt_db.h	GATT データベース関連ヘッダファイル
ble_gatt_proc.h	GATT プロシージャ API 関連ヘッダファイル
ble_hci.h	HCI 関連ヘッダファイル
ble_hci_cmd.h	HCI コマンド関連ヘッダファイル
ble_hci_drv.h	HCI ドライバ関連ヘッダファイル
ble_l2cap.h	L2CAP API 関連ヘッダファイル
ble_scan.h	スキャン関連ヘッダファイル
ble_smp.h	SMP 関連ヘッダファイル
ble_tim.h	BLE 時間管理関連ヘッダファイル
ble_adv.h	アドバタイズ関連ヘッダファイル
ble_api.h	デバイス API、GAP、GATT API 関連ヘッダファイル
ble_att.h	ATT 関連ヘッダファイル
ble_cfg.h	BLE コンフィグ用マクロ
ble_con.h	イニシエータ API 関連ヘッダファイル
ble_con_mng.h	
ble_err.h	Bluetooth Core Specification で定義されたエラーコード

lib¥M4¥EWARM¥vX(※)	
BLEcortexm4.eww	EWARM ワークスペースファイル
BLEcortexm4l.a	BLE スタックビルド済みライブラリファイル
BLEcortexm4l.ewp	EWARM コンフィギュレーションファイル

※本 BLE スタックは EWARM の v7 系もしくは v8 系をサポートしています。コンフィグレータのソース生成操作時、ソース生成先フォルダに適切な.a ファイルがコピーされます。

sample	
ble_ad_type.c	アドバタイズパケットの設定に使用するサンプルアプリケーション
ble_ad_type.h	
ble_cfg.c	BLE スタックコンフィギュレーション用ファイル
ble_cus_srv.c	仮想 UART サービスのサンプルアプリケーション
ble_srv_uart.c	
ble_uart_peripheral.c	
ble_err_str.c	サンプルアプリケーション用エラーコード
ble_shell.c	サンプルアプリケーションを動作させるためのシェル
ble_strlib.c	文字列操作関連ソースファイル
ble_strlib.h	

src	
ble_adv.c	アドバタイズ関連ソースファイル
ble_att_cli.c	ATT 関連ソースファイル
ble_att_ser.c	
ble_con.c	イニシエータ API 関連ソースファイル
ble_con_mng.c	
ble_dev.c	デバイス API 関連ソースファイル
ble_gatt_db.c	GATT サービス API 関連ソースファイル
ble_gatt_proc.c	GATT プロシージャ API 関連ソースファイル
ble_hci.c	HCI 関連ソースファイル
ble_hci_cmd.c	HCI コマンド関連ソースファイル
ble_hci_l2cap.c	L2CAP API 関連ソースファイル
ble_scan.c	スキャン API 関連ソースファイル
ble_sec.c	セキュリティ関連ソースファイル
ble_smp.c	SMP 関連ソースファイル
ble_srv_gap.c	GAP プロファイル API 関連ソースファイル
ble_srv_gatt.c	GATT プロファイル API 関連ソースファイル
ble_tim.c	BLE 時間管理関連ソースファイル

3 機能

BLE スタックの機能を説明します。

BLE プロトコルは下図のような構成となっており、赤字部分が、BLE スタックが提供する機能を示しています。

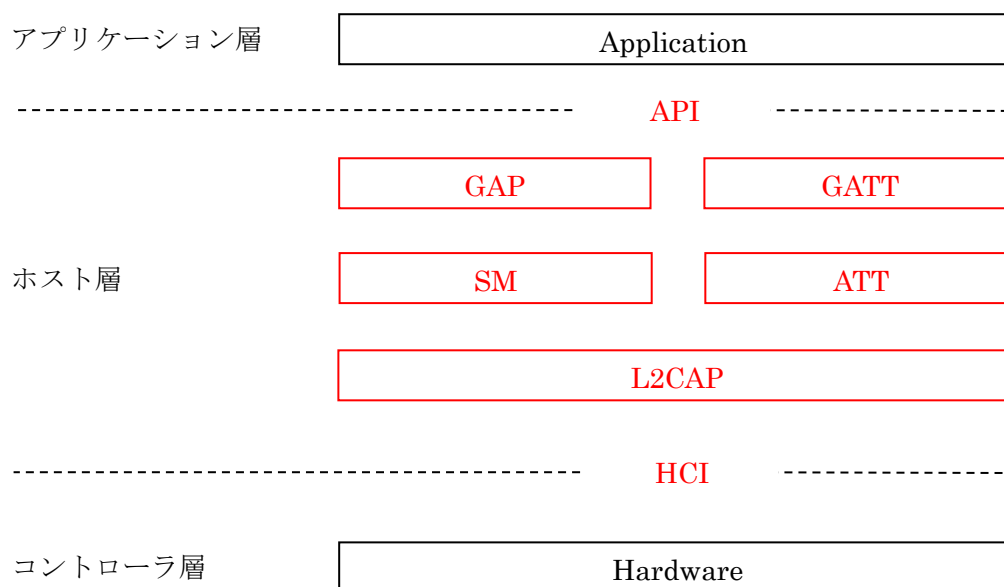


図 3-1 BLE スタックの構成図

BLE ホスト層として、GAP、GATT、ATT、SM、L2CAP を実装しており、アプリケーションプログラムから各機能を使用するために API を用意しています。また、BLE コントローラ層(ハードウェア)とのインターフェースとして、HCI を実装しています。

3.1 HCI (Host Controller Interface)

HCI は、ホストとコントローラ間で通信を行うためのプロトコルです。BLE スタックでは、HCI トランスポートとして SDIO を使用します。

3.2 L2CAP (Logic Link Control and Adaptation Protocol)

L2CAP は、デバイス間通信における論理チャンネルを確保し、上位層から受け取ったプロトコルを、下位層に渡すためのパケットフォーマットに変換、分割するためのプロトコルです。また、下位層から受け取ったプロトコルを、上位層に渡すための変換、結合を行います。

3.3 ATT (Attribute Protocol)

ATT は、デバイス間のクライアント/サーバアーキテクチャを実現するためのプロトコルです。ATT では、アトリビュートという形式で構造化したデータのやり取りが行われます。GATT サーバは自身が保持するアトリビュートのセットを公開し、GATT クライアントはアトリビュートにアクセスします。アトリビュートのアクセスに、ATT を使用します。

3.4 GAP (General Access Profile)

GAP は、デバイスが他デバイスの検索を行う、データのブロードキャストを行う、接続の確立を開始するなど、デバイス同士の運用を行うための、役割や通信手順の規定です。デバイスの役割には、ブロードキャスター、オブザーバー、セントラル、ペリフェラルの 4 つがあります。

3.5 GATT (General Attribute Profile)

GATT は、接続したデバイス間で、データのやり取りを行うための役割、データ構造、アクセス手法などの規定です。ATT を使用して、GATT で定義されたデータのやり取りを行います。

3.6 SM (Security Manager)

SM は、デバイス間のペアリング、認証及び暗号化に関するプロトコル(SMP: Security Manager Protocol)と、これらを実現するためのデバイスの動作に関する規定です。

4 BLE の基礎知識

BLE の基礎知識について説明します。詳細については、Bluetooth Core Specification Version4.2 を参照してください。

<https://www.bluetooth.com/ja-jp/specifications/archived-specifications/>

4.1 チャンネル

BLE は、2.4GHz ISM バンドの 2400 – 2483.5MHz で動作し、この周波数帯を 40 個の RF チャンネル(0 - 39)に分割して使用します。この内、37, 38, 39 チャンネルは、アドバタイジングチャンネルと呼ばれ、アドバタイジングパケットに使用されます。残りの 0 から 36 チャンネルは、データチャンネルと呼ばれ、データパケットに使用されます。

4.2 コネクション非確立状態とコネクション確立状態におけるデータ転送

BLE では、ターゲットデバイスとのリンクが確立している状態(コネクション確立状態)もしくは確立していない状態(コネクション非確立状態)の両方において、アプリケーションデータの転送を行うことができます。

コネクション非確立状態においては

- ・データ転送にはアドバタイジングパケットを使用します。
- ・パケットは 1 つのデバイスから複数のデバイスにブロードキャストされます。
- ・データ転送は 1 方向性です。

コネクション確立状態においては、

- ・データ転送にはデータパケットを使用します。
- ・データ転送は 1 対 1 で行われます。
- ・データ転送は双方向性です。
- ・パケットは、ターゲットから応答があるまで再送されるため、信頼性があります。

一般的に、アドバタイジングパケットはデバイス固有の情報をブロードキャストするために使用され、データパケットはリアルタイムのアプリケーションデータ転送のために使用されます。

4.3 デバイスアドレス

BLE では、48bit (6byte)のアドレスを使用してターゲットデバイスを識別します。各 BLE デバイスは、パブリックデバイスアドレスと呼ばれるユニークなアドレスを持ちます。

また、セキュリティのため、ランダムデバイスアドレスを持つことができます。このアドレスはホストによって設定可能です。セキュリティ要件に応じて、パブリックアドレスを使用するか、ランダムアドレスを使用するか選択できます。

ランダムアドレスは、スタティックアドレスとプライベートアドレスに分けられます。更にプライベートアドレスは、Resolvable プライベートアドレスと Non-Resolvable プライベートアドレスに分けられます。

4.3.1. パブリックデバイスアドレス

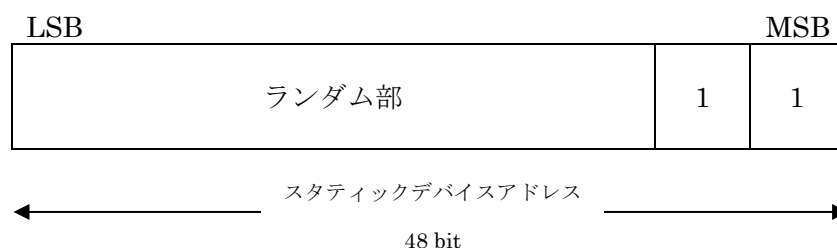
パブリックデバイスアドレスは、BLE コントローラごとに固有のアドレスです。上位 24bit は company_id と呼ばれ、Bluetooth SIG が各メーカーに割り振った値が使用されます。下位 24bit は、company_assigned と呼ばれ、各メーカーが割り振った値が使用されます。

4.3.2. スタティックデバイスアドレス

デバイスやアプリケーションは、電源サイクル毎にスタティックデバイスアドレスを生成することができます。スタティックデバイスアドレスは、次の電源サイクルまで変更してはいけません。

スタティックアドレスの条件は以下の通りです。

- 48bit のうち、MSB 及び 1 つ下の bit が 1。
- 残り 46bit(ランダム部)の全 bit が 0 または 1 であってはいけない。

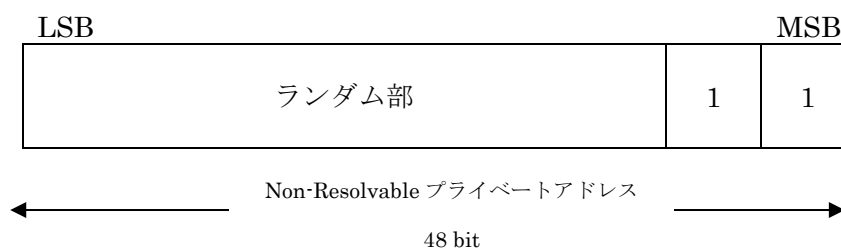


4.3.3. Non-Resolvable プライベートアドレス

Non-Resolvable プライベートアドレスはコネクション非確立状態においてのみ使用します。このアドレスは、15 分毎もしくは設定した時間毎に変更する必要があります。

Non-Resolvable プライベートアドレスの条件は以下の通りです。

- 48bit のうち、MSB 及び 1 つ下の bit は 0。
- 残り 46bit(ランダム部)の全 bit が 0 または 1 であってははいけない。
- パブリックアドレスと一致してはいけない。



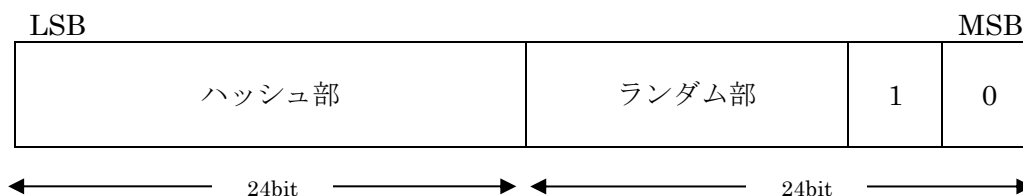
4.3.4. Resolvable プライベートアドレス

Resolvable プライベートアドレスは、IRK(Identity Resolving Key)の値を使用して生成されるセキュアなアドレスです。同じ IRK を持つデバイスのみが、Resolvable プライベートアドレスを解決できます。このアドレスは、15 分毎もしくは設定した時間毎に変更する必要があります。

Resolvable プライベートアドレスは、上位 24bit のランダム部(prand)と、下位 24bit のハッシュ部(hash)に分割されます。hash は、Bluetooth core specification [Vol 3] Part H, Section 2.2.2 で定義されている、ランダム・アドレス・ハッシュ関数 $ah(hash = ah(IRK, prand))$ を使用して生成される値です。

Resolvable プライベートアドレスの条件は以下の通りです。

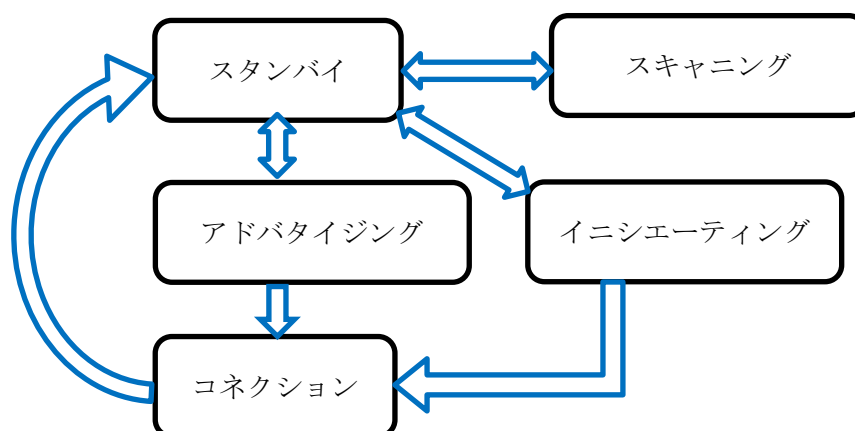
- prand の MSB は 0。その 1 つ下の bit は 1。
- prand の全 bit が 0 または 1 であってははいけない。



本 BLE スタックでは、スタティックアドレスを設定する場合、API : `ble_set_dev_addr()` を使用します。パブリックアドレスやスタティックアドレスを取得する場合、API : `ble_get_dev_addr()` を使用します。

4.4 デバイスの状態

BLE デバイスは、スタンバイ状態、アドバタイジング状態、スキャンニング状態、イニシエーティング状態、コネクション状態のいずれかになります。状態遷移図を示します。



4.4.1. スタンバイ状態

スタンバイ状態にあるデバイスは、パケットの送受信はできません。

本 BLE スタックでは、API : ble_ini0を実行した後、デバイスがスタンバイ状態に遷移します。

4.4.2. アドバタイジング状態

アドバタイジング状態では、一定間隔でアドバタイジングパケットをブロードキャストで送信します。この感覚は、20ms から 10.24s の範囲で設定可能です。アドバタイジングパケットには、最大で 31 バイトのアプリケーションデータを含めることができます。また、アドバタイズパケットを送信しているデバイスが、スキャンしているデバイスからスキャンリクエストを受信した場合、スキャンレスポンスパケットと呼ばれるパケットを使用して、更に 31 バイトの追加データを転送することができます。

規定では、アドバタイジングパケットは 3 つのアドバタイジングチャンネルすべてを使って転送されますが、選択したチャンネルのみを使用するように設定できます。

アプリケーションは、アドバタイジングパケットにパブリックアドレスを使用するか、ランダムアドレスを使用するか設定できます。

アドバタイジングパケットには 4 つのタイプがあり、アプリケーションはこれらのタイプを選択することができます。

アドバタイジングパケットタイプ	説明
ADV_IND (コネクション可、無向)	デバイスがコネクションの確立を受け入れることを示します。 スキャンレスポンスパケットを使用して、追加のデータを送信することができます。
ADV_NONCONN_IND	デバイスがコネクションの確立を受け入れないこと

(コネクション不可、無向)	を示します。
ADV_SCAN_IND (スキャン可、無向)	デバイスがスキャンレスポンスパケットを使用して、追加のデータを送信できることを示します。 このタイプでは、アプリケーションデータは送信できません。
ADV_DIRECT_IND (コネクション可、有向)	ターゲットデバイスのアドレスが既に分かっている場合に、デバイスがコネクションの確立を受け入れることを示します。 このタイプでは、アプリケーションデータは送信できません。

本 BLE スタックでは、アドバタイジングパケットの送信を開始するために、API : `ble_ena_adv()`を使用します。また、送信を停止するためには、API : `ble_dis_adv()`を使用します。アドバタイジングパケットのタイプ、使用チャンネル、送信間隔等を設定する際は、API : `ble_cfg_adv()`を使用します。アプリケーションデータを設定する際は、API : `ble_set_adv_dat()`を使用します。スキャンレスポンスデータを設定する際は、API : `ble_set_adv_scan_rsp_dat()`を使用します。

4.4.3. スキャン状態

スキャン状態では、アドバタイズパケットを受信することができます。デバイスは一定間隔でアドバタイジングパケットをスキャンします。この間隔は、2.5ms から 10.24s の範囲で設定可能です。規定では、スキャナーは 3 つのアドバタイジングチャンネルすべてをスキャンしますが、選択したチャンネルのみをスキャンするように設定可能です。

スキャンには 2 つのタイプがあります。

スキャンタイプ	説明
Passive	アドバタイジングパケットを受信します。
Active	アドバタイジングパケットを受信します。 また、アドバタイズパケットを送信しているデバイスに、スキャンリクエストパケットを送信し、その応答としてスキャンレスポンスパケットを受信します。

本 BLE スタックでは、スキャンを開始するために、API : `ble_ena_scan()`を使用します。また、スキャンを停止するためには、API : `ble_dis_scan()`を使用します。

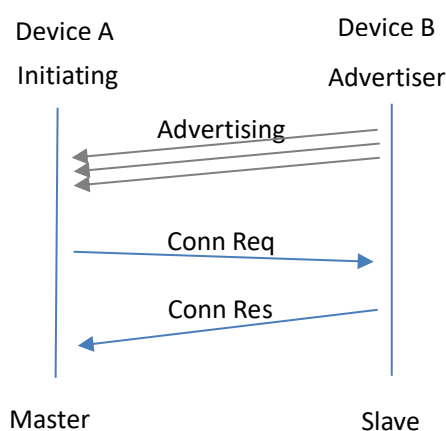
受信したアドバタイジングパケットは、コールバックイベント : `BLE_APP_ADV_EVT` によって、アプリケーションに通知されます。

4.4.4. イニシエーティング状態

イニシエーティング状態は、スキャン状態と同様に、他デバイスが送信するアドバタイズ packets をスキャンします。また、接続可タイプのアドバタイズ packets を受信した場合、接続要求 packets を送信し、接続の確立を開始します。

4.4.5. 接続状態

データ packets のやり取りは、接続状態にある BLE デバイス間において可能になります。2 つの BLE デバイス(デバイス A、デバイス B とする)は、以下のシーケンスで接続状態になります。



デバイス B は、接続可能な状態にあり、アドバタイジング packets をブロードキャスト送信しています。デバイス A は、イニシエーティング状態にあり、アドバタイジング packets のスキャンをしています。デバイス A は、アドバタイジング packets を受信すると、接続要求 packets を送信します。デバイス B は、接続要求 packets を受け入れ、接続応答 packets を送信します。この時点で、接続が作成されたとみなされます。

接続が作成されると、デバイス間でデータ packets のやり取りができるようになります。データ packets を受信した段階で、接続が確立されたとみなされます。接続が確立した後は、接続を開始したデバイスはマスター、接続のためのアドバタイジング packets を送信していたデバイスはスレーブと呼ばれます。マスターデバイスは、複数のスレーブ機器と接続を確立することができます。

接続が確立されると、マスターデバイスからスレーブデバイスに対して、一定間隔でデータ packets を送信し、スレーブデバイスはそれに応答するデータ packets を送信します。これを接続イベントといい、packet 送信間隔を接続間隔といいます。どちらかがデータ packets を送信しない時間が続くと、接続は切断されたとみなされます。接続イベントの間隔は、7.5ms から 4s の範囲で 1.25ms 単位で設定でき、

コネクション確立の際に、デバイス間でネゴシエートされます。

本 BLE スタックでは、コネクションを確立するために、API : `ble_cre_con()`を使用します。コネクションを切断するために、API : `ble_cls_con()`を使用します。

4.5 コネクションパラメータ

主要なコネクションパラメータについて説明します。デバイスのデータスループットと、消費電流は、コネクション間隔とスレーブレイテンシに影響されます。

4.5.1. コネクション間隔

コネクション間隔は、デバイス間のコネクションイベント(データパケットの転送)間隔です。この間隔で、マスターデバイスからスレーブデバイスにデータパケットが送信され、スレーブデバイスはマスターデバイスに応答データパケットを返します。コネクション間隔が小さいほど、データ送信の頻度が高くなるため、データスループットが大きくなります。また、その分消費電流が大きくなります。

4.5.2. スレーブレイテンシ

スレーブレイテンシは、スレーブデバイスがコネクションイベントを無視できる回数です。スレーブデバイスは、マスターデバイスに対して送信するデータが無い場合、コネクションイベントを無視することで消費電流を小さくすることができます。スレーブデバイスは、マスターデバイスに対して送信するデータが無い場合でも、(スレーブレイテンシ+1)回目のコネクションイベントに対しては、データパケットを応答する必要があります。スレーブレイテンシが 0 に設定されている場合、スレーブデバイスはすべてのコネクションイベントに対して応答データパケットを返します。

4.5.3. スーパービジョンタイムアウト

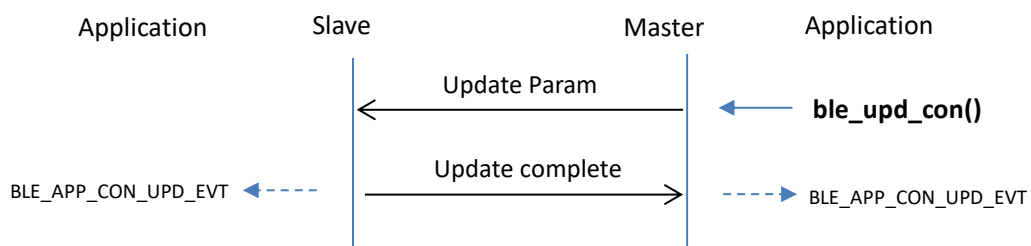
スーパービジョンタイムアウトの時間を超えて、データパケットを受信しなかった場合、コネクションが切断されたとみなされます。例えば、コネクション確立後、片方のデバイスの電源が切れた場合、もう片方のデバイスがデータパケットを受信することができなくなり、スーパービジョンタイムアウトの時間が経過し、コネクションが切断されます。

4.5.4. コネクションパラメータ更新

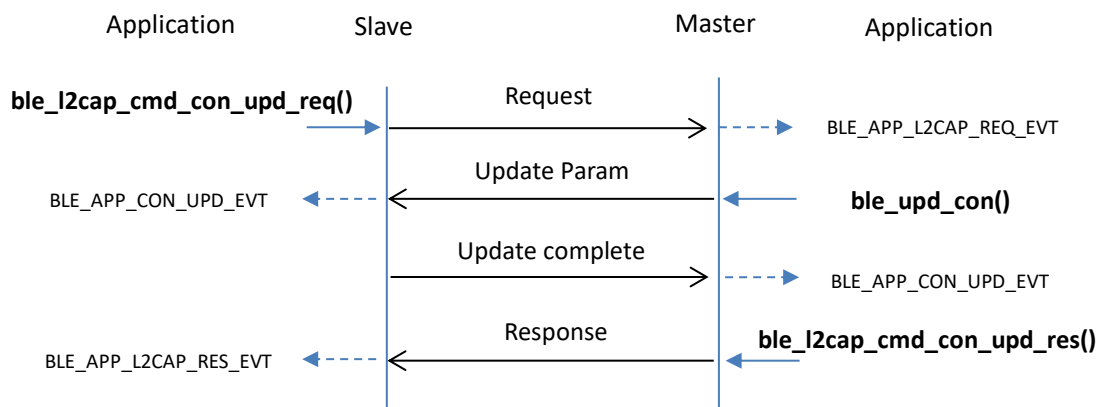
コネクションパラメータは、コネクションを切断せずに変更することができます。この手順をコネクションパラメータ更新といいます。

大量のデータパケットをやり取りするときはコネクション間隔を小さくし、データパケット転送の頻度が低いときはコネクション間隔を大きくして、消費電流の値を制御することができます。

コネクションパラメータ更新は、マスターデバイスのみが開始できます。本 BLE スタックでは、コネクションパラメータ更新のために、API : `ble_upd_con()` を実行します。更新が完了すると、`BLE_APP_CON_UPD_EVT` イベントを受信します。



スレーブデバイスは、コネクションパラメータ更新を開始できませんが、マスターデバイスに対してコネクションパラメータ更新を開始するよう要求することができます。本 BLE スタックでは、API : `ble_l2cap_cmd_con_upd_req()` と、`ble_l2cap_cmd_con_upd_res()` を使用します。またこのとき、`BLE_APP_L2CAP_REQ_EVT`、`BLE_APP_L2CAP_RES_EVT` 及び `BLE_APP_L2CAP_REJ_EVT` イベントが使用されます。



マスターデバイスは、`result` パラメータを 0 以外の値に設定して、API : `ble_l2cap_cmd_con_upd_res()` を実行することで、スレーブの要求を拒否することができます。この場合、スレーブデバイスは `BLE_APP_L2CAP_REJ_EVT` を受信します。

4.6 ホワイトリスト

BLE デバイスが、他デバイスのアドレスリスト(ホワイトリスト)を保持することができます。BLE デバイスの状態に応じて、ホワイトリストは様々な使い方があります。例えば、ホワイトリストに存在するデバイスのパケットのみを受け入れるように設定することができます。

本 BLE スタックでは、アドレスの追加、アドレスの削除、ホワイトリストのリセットのために、それぞれ API : `ble_add_whitelist()`, `ble_rmv_whitelist()`, `ble_clr_whitelist()` を使用します。

4.7 セキュリティ

BLE では、セキュリティ機能として、プライバシー機能とデータ暗号化機能が使用できます。

4.7.1. プライバシー

プライバシー機能により、デバイスアドレスを使用した BLE デバイスの追跡が困難になります。プライバシーモードでは、アドバタイジング、スキャンング及びコネクション確立時に、パブリックアドレスやスタティックアドレスではなく、プライベートアドレスを使用します。プライベートアドレスは定期的に変更されるため、デバイスの追跡が困難になります。

Resolvable プライベートアドレスを使用する場合、プライベートアドレスの解決には IRK (Identity Resolving Key) が必要なため、ボンディングされたデバイス同士のみが相互に通信することができます。

4.7.2. 暗号化

デバイス間には、暗号化されたデータのやり取りを行うことができます。暗号化には AES-CCM 暗号が使用されます。暗号化に使用される鍵は、LTK (Long Term Key) と呼ばれます。LTK は、ペアリング処理において生成されます。

4.7.3. ペアリングとボンディング

IRK 値、LTK 値及びその他のセキュリティ情報は、ペアリング処理において、デバイス間で交換が行われます。あるデバイスが、相手デバイスのセキュリティ情報を持っている場合、そのデバイスは相手デバイスとボンディングしているといえます。

BLE のペアリング方式には、LE セキュアコネクションと LE レガシーペアリングがあります。本 BLE スタックは、より強固なセキュリティを提供する LE セキュアコネクションのみをサポートしています。

ペアリング処理では、IO Capability、MITM、Bonding_Flags、OOB data flag などのパラメータを使用します。IO Capability は、デバイスの入力(キーボードやボタン)や出力(ディスプレイ)機能に関する情報です。MITM (Machine-in-the-middle)は、MITM 保護(Numeric メソッドもしくは Passkey Entry メソッドを使用した認証)を必要とするかどうかを示します。Bonding_Flags は、ペアリング処理要求を開始したデバイスが、ボンディングを必要としているかどうかを示します。OOB (Out of Band、帯域外) data flag は、デバイスが相手デバイスの OOB 認証データを持っているかどうか、すなわち帯域外のメカニズムを使用した認証が利用可能かどうかを示します。

上記の 4 つのパラメータに応じて、Just Works、Numeric、Passkey Entry、OOB のいずれかのペアリングメソッドが使用されます。両デバイスの IO Capability に応じて、Just Works、Numeric、Passkey Entry のいずれかが使用されます。また、IO Capability の組み合わせによらず、両デバイスで MITM がセットされていないときは Just Works が使用され、片方あるいは両方のデバイスで OOB data flag がセットされているときは、OOB が使用されます。

各パラメータや、ペアリングメソッドの詳細な情報等につきましては、Bluetooth Core Specification Version 4, Vol 3, Part H, 2.3 Paring Method を参照してください。

5 BLE アプリケーションプロファイル

BLE 製品間の互換性確保のため、Bluetooth 規格ではアプリケーションデータの構造や、アプリケーションデータにアクセスするための手順を規定しています。これらの仕様はプロファイルと呼ばれています。Bluetooth Core Specification では、GAP と GATT と呼ばれるプロファイルが規定されています。

5.1 GATT プロファイル

GATT プロファイルは、アプリケーションデータ構造の規定です。アプリケーションデータへアクセスする方法は、ATT プロトコルで規定されています。アプリケーションデータを持つデバイスを GATT サーバ、GATT サーバにデータにアクセスするデバイスを GATT クライアントと呼びます。

5.1.1. アトリビュート

GATT を構成する、以下の構造のデータをアトリビュートといいます。アトリビュートはハンドル、タイプ、値、パーミッションの 4 つで構成されます。

アトリビュート ハンドル	アトリビュート タイプ	アトリビュート値	アトリビュート パーミッション
各アトリビュートの識別に使用する 2byte のユニークな値	アトリビュート値のフォーマットを示す。16 or 32 or 128bit の UUID として指定される。	アトリビュートが持つ値	アトリビュート値へのアクセス許可 (Read 、 Write 、 Authentication など)を示す。

5.1.2. アトリビュート UUID

Bluetooth 規格では、アトリビュートタイプを識別するためのユニークな番号：UUID(Universal Unique Identifier)が定義されています。UUID を知ることで、アプリケーションは、アトリビュート値のフォーマットを知ることができます。例えば、UUID:0x2A00 は UTF-8 フォーマットの「デバイス名」であることを示します。

割り当てられた UUID は、Assigned Numbers で確認できます。

また、ベンダー固有のアトリビュートには 128bit UUID が使用されます。

5.1.3. アトリビュート階層構造

GATT サーバは、構造化された複数のアトリビュートを管理し、プロファイルを構成します。プロファイルは、サービス、インクルード、キャラクタリスティック、キャラクタリスティック値、キャラクタリスティックディスクリプタと呼ばれるアトリビュートが使用されます。

GATT サーバは、1 つ以上のプロファイルを実装する必要があります。各プロファイルは、1 つ以上のサービスを持つ必要があります。各サービスは、1 つ以上のキャラクタリスティックを持つ必要があります。各キャラクタリスティックは、1 つのキャラクタリスティック値を持つ必要があります。また、キャラクタリスティックはキャラクタリスティックディスクリプタを持つ場合があります。

サービス	関連するキャラクタリスティックのリスト
キャラクタリスティック	キャラクタリスティック値の UUID
キャラクタリスティック値	アプリケーションデータとそのアクセス許可
キャラクタリスティックディスクリプタ	キャラクタリスティック値の追加情報

5.1.4. サービス

サービスは、GATT サーバが持つ機能と、その機能を実現するためのデータ及び動作の集合です。サービスは「サービス定義」によって定義されます。サービス定義は、必ず 1 つの「サービス宣言」を持ちます。サービス宣言は、サービス定義の最初のアトリビュートである必要があります。サービス定義の範囲は、「サービス宣言」から「次のサービス宣言」の前までです。

サービス宣言のアトリビュート構成を以下に示します。

ハンドル	タイプ(UUID)	値	パーミッション
0xNNNN	0x2800 or 0x2801 0x2800:プライマリサービス 0x2801:セカンダリサービス	サービス UUID 値	Read only, No Authentication, No Authorization

5.1.5. インクルード宣言

サービス定義に「インクルード宣言」を追加することで、他のサービスを参照することができます。

インクルード宣言のアトリビュート構成を以下に示します。

ハンドル	タイプ(UUID)	値			パーミッション
0xNNNN	0x2802 インクルード宣言であることを示す。	(1)	(2)	(3)	Read only, No Authentication, No Authorization

- (1) インクルードサービスアトリビュートハンドル。参照したいサービスの、サービス宣言ハンドルです。
- (2) エンドグループハンドル。参照したいサービス定義の、最後のアトリビュートのハンドルです。
- (3) サービス UUID。

5.1.6. キャラクタリスティック

キャラクタリスティックは、サービス定義が持つ、アプリケーションデータ及びアプリケーションデータに関するメタデータの集合です。キャラクタリスティックは、「キャラクタリスティック定義」によって定義されます。キャラクタリスティック定義は、少なくとも「キャラクタリスティック宣言」と「キャラクタリスティック値宣言」という2つのアトリビュートを持つ必要があります。また、キャラクタリスティック値の後に続けて、「キャラクタリスティックディスクリプタ宣言」を持つことができます。

キャラクタリスティック宣言は、アプリケーションデータであるキャラクタリスティック値のメタデータを提供します。キャラクタリスティック値宣言のアトリビュート構成を示します。

ハンドル	タイプ(UUID)	値			パーミッション
0xNNNN	0x2803 キャラクタリスティック宣言であることを示す	(1)	(2)	(3)	Read only, No Authentication, No Authorization

- (1) キャラクタリスティックプロパティ。GATT クライアントはキャラクタリスティックプロパティを read して、どのような操作を行うことができるかを判断します。プロパティ値の詳細については、Bluetooth Core Specification Vol 3, Part G を参照してください。
- (2) キャラクタリスティック値アトリビュートハンドル。このキャラクタリスティック定義が持つ、「キャラクタリスティック値宣言」アトリビュートのハンドルです。
- (3) キャラクタリスティック UUID。Bluetooth SIG によって定められた 16bit UUID、またはユーザーによって独自に定められた 128bit UUID。

キャラクタリスティック値は、GATTサーバクライアント間でやりとりされるアプリケーションデータを保持します。キャラクタリスティック値のアトリビュート構成を示します。

ハンドル	タイプ(UUID)	値	パーミッション
0xNNNN ※1	0xMMMM ※2	アプリケーションデータ	上位のプロファイル依存。 もしくは実装依存。

※1：キャラクタリスティック宣言の値(2)と同値です。

※2：キャラクタリスティック宣言の値(3)と同値です。

キャラクタリスティックディスクリプタ宣言は、キャラクタリスティック値に関するメタデータの拡張情報です。これはオプションであり、キャラクタリスティック定義は、「キャラクタリスティックディスクリプタ宣言」を持たない場合もあります。各キャラクタリスティックディスクリプタの仕様については、Bluetooth Core Specification Vol 3, Part G を参照してください。

5.1.7. アトリビュート検出

アプリケーションデータへアクセスするということは、キャラクタースティック値アトリビュートへアクセスすることと指します。任意のアトリビュートにアクセスするためには、アトリビュートのハンドル値を把握する必要があります。クライアントは、サーバに対して以下の手順でアトリビュートの情報を問い合わせ、ハンドル値を取得することで、アプリケーションデータへのアクセスが可能になります。

(1) サービス検出

すべてのサービスを照会します。

(2) キャラクタースティック検出

サービスが持つ、すべてのキャラクタースティックを照会します。

(3) キャラクタースティックディスクリプタ検出

キャラクタースティックが持つ、すべてのキャラクタースティックディスクリプタを照会します。

(4) アクセス

任意のキャラクタースティック値にアクセスします。

本 BLE スタックでは、以下の API を使用して、アトリビュート検出やデータへのアクセスを実行します。

`ble_disc_all_pri_svc()`、`ble_disc_inc_svc()`、`ble_disc_all_char()`、`ble_disc_all_char_desc()`、`ble_read_char()`、`ble_write_char()`、など

5.1.8. ATT MTU

前述の通り、アトリビュートへのアクセスには ATT プロトコルを使用します。ATT プロトコルにおいて、デフォルトの MTU (Maximum Transmission Unit) は 23 バイトです。サーバとクライアントは、同じ MTU で通信する必要があります。MTU の値は、MTU 交換手順を実行して増やすことができます。

本 BLE スタックでは、MTU 交換手順を実行するために、API : `ble_exchange_mtu()` を使用します。

5.1.9. ロングアトリビュート値

アトリビュート値のサイズが大きく、複数の ATT パケットでアクセスする必要がある場合があります。これをロングアトリビュート値と呼びます。なお、アトリビュート値のサイズ上限は 512 バイトです。

本 BLE スタックでは、ロングアトリビュート値へのアクセスのために、API : `ble_read_long_char()`、`ble_write_long_char()` を使用します。

5.1.10. Notification と Indication

通常、GATT サーバデータへのアクセスは、クライアントからサーバに対して実行されます。GATT では、以下の 2 つの方法によって、サーバ主体で、サーバからクライアントへのデータを送信することができます。

(1) Notification

サーバからクライアントへデータを送信します。クライアントからの確認応答は不要です。

(2) Indication

サーバからクライアントへデータを送信します。クライアントからの確認応答が必要です。

5.2 GAP プロファイル

GAP は、デバイスの各状態をどのように使用すべきかを規定します。また、デバイスに対するセキュリティレベルの要求を規定します。

5.2.1. デバイスロール

BLE デバイスは、GAP で規定されている以下のロール(役割)のいずれかになります

(1) ブロードキャスター

ブロードキャスターは、アドバタイズパケットのブロードキャスト送信を行います。受信は行わず、また、他のデバイスとのコネクションも行いません。

(2) オブザーバー

オブザーバーは、アドバタイズパケットの受信のみを行います。送信は行わず、また他のデバイスとのコネクションも行いません

(3) セントラル

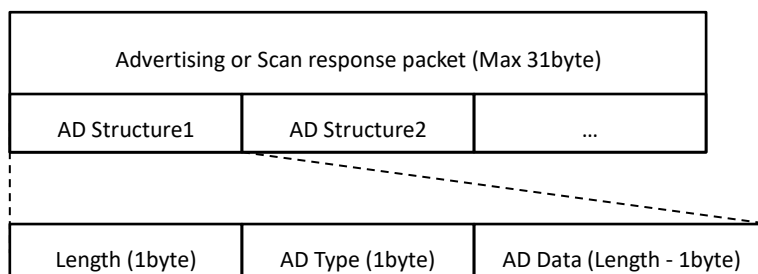
セントラルは、他のデバイスが送信するアドバタイズパケットをスキャンします。受信したアドバタイズパケットに含まれるデバイスアドレスを指定して、コネクション要求を行います。またコネクションが確立したデバイスと、相互に通信を行います。

(4) ペリフェラル

ペリフェラルは、他のデバイスにアドバタイズパケットを送信します。セントラルデバイスからのコネクション要求を受信し、それに応答することでコネクションを確立します。コネクションが確立したデバイスと、相互に通信を行います。

5.2.2. AD タイプ

GAP は、アドバタイジングパケットとスキャンレスポンスパケットのペイロードフォーマット(AD Structure)を規定します。アプリケーションはこのフォーマットを使用する必要があります。なお、ペイロードの最大サイズは 31 バイトです。



ペイロードには、複数の AD Structure を含めることができます。AD Structure は、AD Structure のサイズを示す Length、データの種別を示す AD Type 及び AD Data で構成されます。AD Type の詳細については、<https://www.bluetooth.com/specifications/assigned-numbers/>の、Generic Access Profile を参照してください。AD Data の詳細については、<https://www.bluetooth.com/specifications/bluetooth-core-specification/>の、Core Specification Supplement を参照してください。

本 BLE スタックでは、API : ble_cfg_adv()を実行すると、AD type に AD フラグがセットされます。また、AD Data には、ble_cfg_adv()で設定したパラメータに応じたビットが設定されます。従ってアプリケーションは、API : ble_set_adv_dat() 及び ble_set_scan_rsp_dat()でデータを設定する際、明示的に AD フラグを設定する必要はありません。

5.2.3. 検出モードと手順

GAP において、検出は、セントラルデバイスが、近くのペリフェラルデバイスの情報を得るために使用するプロセスです。GAP では、セントラルデバイスがペリフェラルデバイスを検出することを容易にするために、いくつかのモードと手順を定義しています。また、各モードで使用されるアドバタイジングパラメータ、AD フラグ、継続時間が定義されています。

ペリフェラルデバイスは、コネクション確立前に、以下の表に示すいずれかのモードになります。アプリケーションは、どのモードを使用するかを選択することができます。GAP では、例えば、初回のコネクション確立時には **General Discoverable** モードを使用し、その後は **Non-Discoverable** モードを使用することができます。

本 BLE スタックでは、検出モードを設定するために、API : `ble_cfg_adv0` 実行時に、`adv_disc_mode` パラメータを使用します。

GAP 検出モード	説明
General Discoverable	<ul style="list-style-type: none"> ・コネクション可能なアドバタイジングパケットを送信します。 ・AD フラグには、“LE General Discoverable Mode bit”をセットされます。
Limited Discoverable	<ul style="list-style-type: none"> ・コネクション可能なアドバタイジングパケットを送信します。 ・AD フラグには、“LE Limited Discoverable Mode”ビットがセットされます。 ・180 秒(<code>lim_adv_timeout</code>)を超えてアドバタイジングパケットを送信してはいけません
Non-Discoverable	<ul style="list-style-type: none"> ・コネクション不可能なアドバタイジングパケットを送信します。 ・AD フラグには、“LE General Discoverable Mode”と“LE Limited Discoverable Mode”はセットされません。

セントラルデバイスは、以下の GAP 検出手順のいずれかを実行して、ペリフェラルデバイスを検出することができます。GAP では、各手順で使用するスキャンパラメータが定義されています。

本 BLE スタックでは、検出手順を設定するために、API : `ble_ena_scan()` の `scan_mode` パラメータを使用します。

GAP 検出手順	説明
General	<ul style="list-style-type: none">• General Discoverable モードもしくは Limited Discoverable モードのペリフェラルデバイスを検出します。• 検出時間は、最低でも 10.24 秒は必要です。
Limited	<ul style="list-style-type: none">• Limited Discoverable モードのペリフェラルデバイスを検出します。• 検出時間は、最低でも 10.24 秒は必要です。

5.2.4. コネクションモードと手順

検出と同様、GAP ではコネクションモードとコネクション確立手順が定義されています。ペリフェラルデバイスは、以下のコネクションモードのうち 1 つを使用します。

本 BLE スタックでは、コネクションモードを設定するために、API : `ble_cfg_adv()` の `adv_conn_mode` パラメータを使用します。

GAP コネクションモード	説明
Non-Connectable	・コネクション不可能なモードです。
Directed Connectable	・コネクション要求手順を実行する既知のデバイスからのコネクション要求を受け入れるモードです。
Un-directed Connectable	・コネクション要求手順を実行するデバイスからのコネクション要求を受け入れるモードです。

セントラルデバイスは、以下のコネクション確立手順のうち 1 つを使用します。

本 BLE スタックでは、コネクション確立手順を設定するための専用の API はありません。スキャンやコネクションの API を使用して、コネクション手順を実行します。

GAP コネクション手順	説明
Directed Connection Establishment	・ホワイтлиストを無視して、指定した特定の 1 台のデバイスとのコネクションを確立することができます。
Auto Connection Establishment	・ホワイтлиストに登録されているアドレスと一致するアドレスを持つデバイスと、自動的にコネクションを確立します。
Selective Connection Establishment	・ホワイтлиストに登録されているアドレスを指定して、アドバタイズパケットをフィルタリングします。検出したデバイスとコネクションを確立するかどうか、アプリケーションで選択します。
General Connection Establishment	・すべてのデバイスのアドバタイズパケットを受け入れます。検出したデバイスとコネクションを確立するかどうか、アプリケーションで選択します。

6 機能概要

本 BLE スタックの機能について説明します。

6.1 イベントハンドラ

本 BLE スタックは、コールバック関数を使用して、様々なイベントをアプリケーションに通知します。このコールバック関数/イベントハンドラは、API : `ble_ini()` で登録します。利用可能なイベントタイプとパラメータの詳細については、後述の「7.1 1 コールバックイベント」を参照してください。

※イベントハンドラは、BLE スタックの内部タスクコンテキストから呼び出されるため、アプリケーションでイベントハンドラをブロックしないでください。

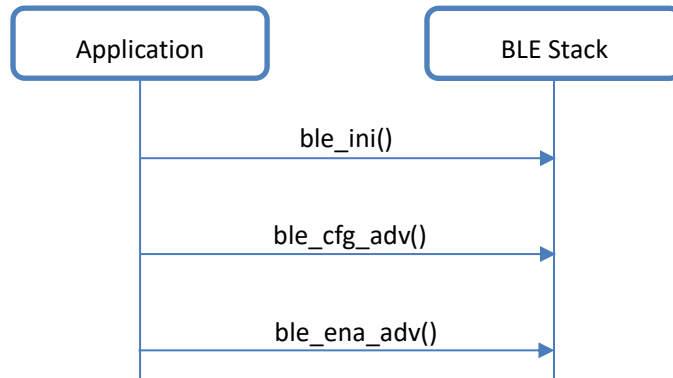
6.2 アイデンティティアドレス

デバイスは、アドバタイジング、スキャン及びコネクション確立の操作において、パブリックアドレスもしくはスタティックアドレスのどちらかを使用します。アドバタイザー、スキャナー及びイニシエータを構成する際は、それぞれパラメータ : `own_addr_type` を設定する必要があります。

デバイスに設定されたパブリックアドレス及びスタティックアドレスは、API : `ble_get_dev_addr()` で読み取ることができます。また、スタティックアドレスは、API : `ble_set_dev_addr()` で設定することができます。

6.3 ブロードキャスターロール

ブロードキャスターロールで動作させるためには、以下の順番で API を実行する必要があります。



(1) `ble_ini()`

BLE スタック初期化。

(2) `ble_cfg_adv()`

アドバタイジングパラメータを設定します。パラメータ：`adv_conn_mode` には、`BLE_CONN_MODE_NON` もしくは `BLE_CONN_MODE_NON_SCAN` を設定してください。パラメータ：`adv_disc_mode` には、`BLE_DISC_MODE_NON` を設定してください。

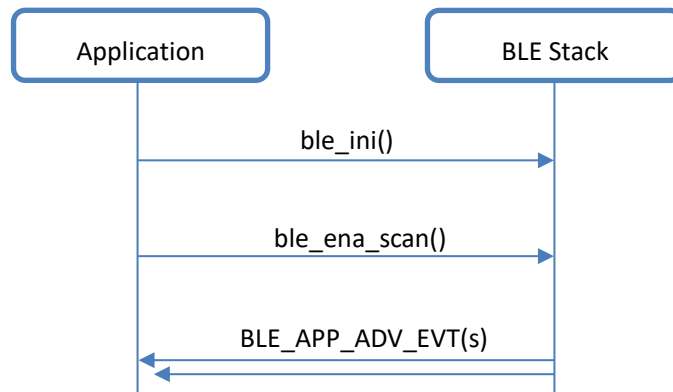
(3) `ble_ena_adv()`

アドバタイジングパケットの送信を開始します。

また必要に応じて、`ble_set_adv_dat()`や`ble_set_adv_scan_rsp_dat()`を実行して、アドバタイジングデータを設定することができます。

6.4 オブザーバーロール

オブザーバーロールで動作させるためには、以下の手順で API を実行する必要があります。



(1) ble_ini()

BLE スタック初期化

(2) ble_ena_scan()

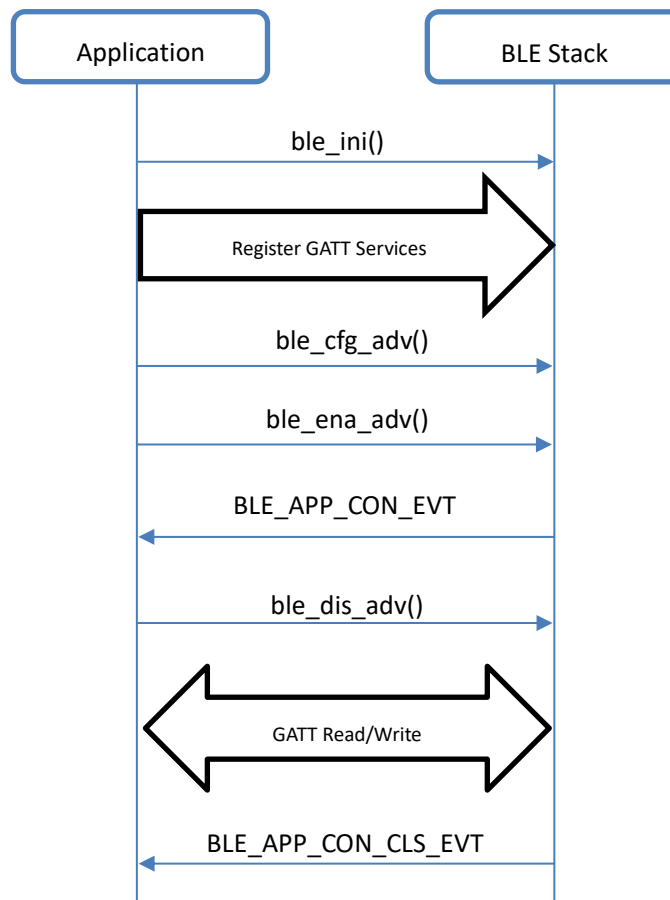
スキャンを開始し、アドバタイジングイベントの通知を待機します。パラメータ : scan_mode には、BLE_SCAN_MODE_OBS を設定してください。パラメータ : scan_type には、BLE_SCAN_TYPE_ACTIVE もしくは BLE_SCAN_TYPE_PASSIVE を設定してください。

(3) BLE_APP_ADV_EVT(s)

デバイスが受信したアドバタイジングレポートは、BLE_APP_ADV_EVT イベントでアプリケーションに通知されます。

6.5 ペリフェラルロール

ペリフェラルロールで動作させるためには、以下の手順で API を実行する必要があります。



- (1) `ble_ini()`
BLE スタック初期化
- (2) `Register GATT Services`
GATT サービスの登録。後述の 6.6.1 GATT サービスの登録を参照してください。
- (3) `ble_cfg_adv()`
アドバタイジングパラメータを設定します。パラメータ：`adv_conn_mode` には、`BLE_CONN_MODE_UND` を設定してください。パラメータ：`adv_disc_mode` には、`BLE_DISC_MODE_GEN` を設定してください。
- (4) `ble_ena_adv()`
アドバタイジングパケットの送信を開始し、コネクションの確立を待機します。
- (5) `BLE_APP_CON_EVT`
コネクション確立イベントの通知。
- (6) `ble_dis_adv()`
アドバタイジングパケットの送信を停止します。

(7) GATT read / write

セントラルデバイスからの read / write 要求処理。後述の 6.6.2 アトリビュートの read / を参照してください。

(8) BLE_APP_CON_CLS_EVT

コネクションが切断された場合、アプリケーションにイベントが通知されます。

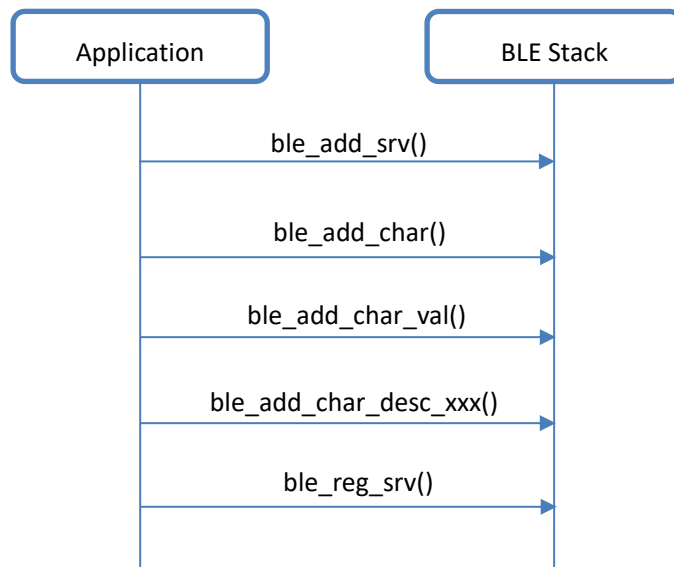
また必要に応じて、ble_set_adv_dat()や ble_set_adv_scan_rsp_dat()を実行して、アドバタイジングデータを設定することができます。

6.6 GATT サーバ

GAP サービス(UUID = 0x1800)と GATT サービス(UUID = 0x1801)は、BLE スタックに含まれています。アプリケーションは、GATT サービス API を使用して他のサービスを追加することができます。

6.6.1. GATT サービスの登録

GATT サービスを登録するためには、以下の手順を実行してください。



- (1) `ble_add_srv()`
サービスアトリビュートを追加します。
- (2) `ble_add_char()`
キャラクタリスティックアトリビュートを追加します。
- (3) `ble_add_char_val()`
キャラクタリスティック値アトリビュートを追加します。
- (4) `ble_add_char_desc_xxx()`
必要に応じて、キャラクタリスティックディスクリプタアトリビュートを追加します。
- (5) `ble_reg_srv()`
サービスを登録します。

(1)を実行した後、(2)から(4)を繰り返すことで、サービスアトリビュートに複数のキャラクタリスティックアトリビュートを追加することができます。サービスアトリビュートは、任意の数のキャラクタリスティックアトリビュートを持つことができ、キャラクタリスティックアトリビュートは、1つのキャラクタリスティック値アトリビュートと、複数のキャラクタリスティックディスクリプタアトリビュートを持つことができることに注意してください。また、追加できるアトリビュートの数は、`CFG_BLE_MAX_GATT_HND` に設定し

た値になります。詳細は、”BLE スタックチュートリアルガイド”を参照してください。

アトリビュートの追加は、`ble_ini()`の実行後かつコネクションが確立される前に実行する必要があります。また、一度追加したアトリビュートを変更することはできません。

6.6.2. アトリビュートの read / write

BLE スタックは、アトリビュートのコールバック関数を介して、受信した read/write 要求をアプリケーションに通知します。アトリビュートのコールバック関数は、アトリビュートの登録時に、アプリケーションによって登録されます。コールバック関数の引数：T_BLE_GATT_HND_PARAM は、受信した要求の詳細をアプリケーションに提供します。

• read 処理

read 要求に対しては、以下のような処理を実行します。

```
UH attr_callback(T_BLE_GATT_HND_PARAM *param)
{
    ..
    /* 1. Check for Read request type */
    if (param->type == BLE_CHAR_CBK_READ) {
        /* 2. Application data size should not be more than param->dat_len*/
        if (app_data_len > param->dat_len) {
            app_data_len = param->dat_len;
        }
        /* 3. Copy application data to 'p_dat' location */
        ble_memcpy(param->p_dat, app_data, app_data_len);
    }
    ..
    return app_data_len; /* 4. Return the copied data size*/
}
```

ロングアトリビュート(ATT の MTU を超えるサイズ)の読み込み要求に対しては、アプリケーションは以下のような処理を実行します。

```
UH attr_callback(T_BLE_GATT_HND_PARAM *param)
{
    ..
    /* 1. Check for Read long request type */
    if (param->type == BLE_CHAR_CBK_READ_LONG) {
        /* 2. Application data size should not be more than param->dat_len*/
        if (app_data_len > param->dat_len) {
            app_data_len = param->dat_len;
        }
        /* 3. Copy application data of request offset to 'p_dat' location */
        ble_memcpy(param->p_dat, app_data[param->offset], app_data_len);
    }
    ..
    return app_data_len; /* 4. Return the copied data size*/
}
```

- write 操作

write 要求に対しては、以下のような処理を実行します。

```
UH attr_callback(T_BLE_GATT_HND_PARAM *param)
{
    ..
    /* 1. Check for Write request type */
    if (param->type == BLE_CHAR_CBK_WRITE) { /* or BLE_CHAR_CBK_WRITE_CMD */
        /* 2. Copy the received data to application buffer */
        ble_memcpy(app_data, param->p_dat, param->dat_len);
    }
    ..
    return 0; /*3. Return value is not used */
}
```

ロングアトリビュートへの write に対しては、Queued write コマンドが使用されます。まず、BLE_CAHR_CBK_WRITE コマンドで受信したデータは、一時的にバッファにコピーします。BLE_CHAR_CBK_WRITE_EXE コマンドを受信したら、一時バッファからアプリケーションバッファにデータをコピーします。

```
UH attr_callback(T_BLE_GATT_HND_PARAM *param)
{
    ..
    /* 1. Check for Write Queue request type */
    if (param->type == BLE_CHAR_CBK_WRITE_PRE) {
        /* 2. Copy the received data to a temporary buffer */
        ble_memcpy(buffer[param->offset], param->p_dat, param->dat_len);
        total_received_len += param->dat_len;

    }
    /* 3. Check for Write execute request type */
    else if (param->type == BLE_CHAR_CBK_WRITE_EXE) {
        /* 4. Check whether the temporary data should be accepted or not */
        if (param->flag == 0x01) { /* 1- Accept, 0 - Reject */
            /* 5. Copy all received data to application buffer */
            ble_memcpy(app_data, buffer, total_received_len);
        }
    }

    ..
    return 0; /* 6. Return value is not used */
}
```

・エラー応答

リクエストパラメータを許容できないケースについては、エラー応答を返す必要があります。利用可能なエラーコードについては、後述の 7.1 2.2 を参照してください。

```
UH att_callback(T_BLE_GATT_HND_PARAM *param)
{
    ..
    /* 1. Set the required ATT Error code */
    param->error = ATT_ERR_INVALID_OFFSET;

    ..
    /* 2. return value should be 0 */
    return 0;
}
```

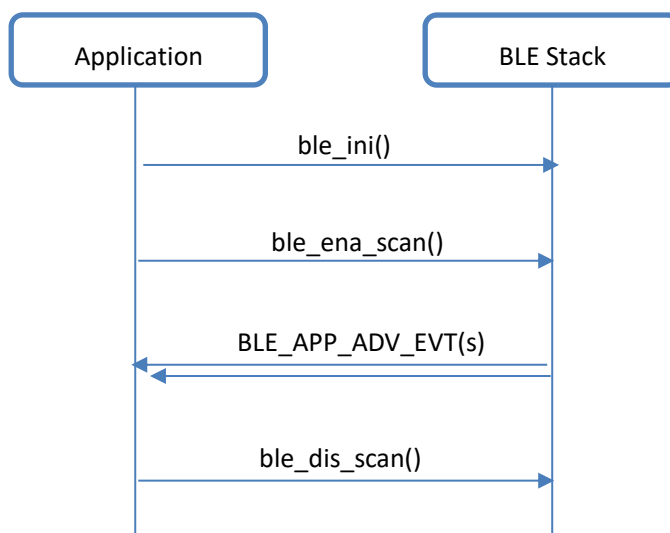
・セキュリティ

read / write 操作において、セキュリティが要求される場合、コネクションが暗号化されているかどうかを確認するために、API : ble_get_enc_state() を使用します。もし暗号化がされていない場合、エラーコード : BLE_ATT_ERR_INSUFFICIENT_ENCRYPTION が返されます。

6.7 セントラルロール

セントラルロールとして動作させるためには、以下の手順で、ペリフェラルデバイスを検出し、コネクションを確立する API を実行する必要があります。

6.7.1. ペリフェラルデバイスの検出



(1) `ble_ini()`

BLE スタック初期化。

(2) `ble_ena_scan()`

スキャンを開始し、アドバタイジングイベントの通知を待機します。パラメータ：
`scan_mode` には、`BLE_SCAN_MODE_LIM` もしくは `BLE_SCAN_MODE_GEN` を設定してください。
 パラメータ：`scan_type` には、`BLE_SCAN_TYPE_ACTIVE` を設定してください。

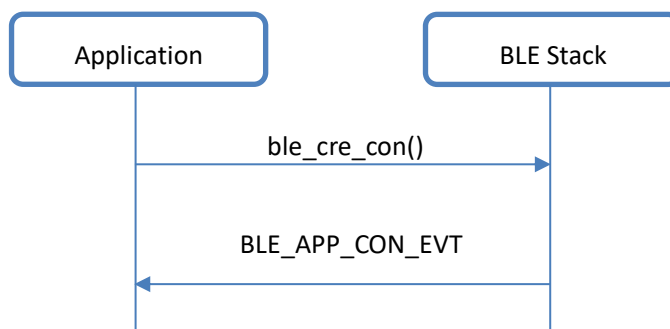
(3) `BLE_APP_ADV_EVT(s)`

デバイスが受信したアドバタイジングレポートは、`BLE_APP_ADV_EVT` イベントでアプリケーションに通知されます。

(4) `ble_dis_scan()`

スキャンを停止します。

6.7.2. コネクション



(1) `ble_cre_con()`

アドレス情報を既に持っている、もしくは上記の検出手順の実行によってアドレス情報入手したペリフェラルデバイスに対して、「コネクション確立」要求を開始します。

(2) `BLE_APP_CON_EVT`

コネクション確立要求に対する結果をアプリケーションに通知します。

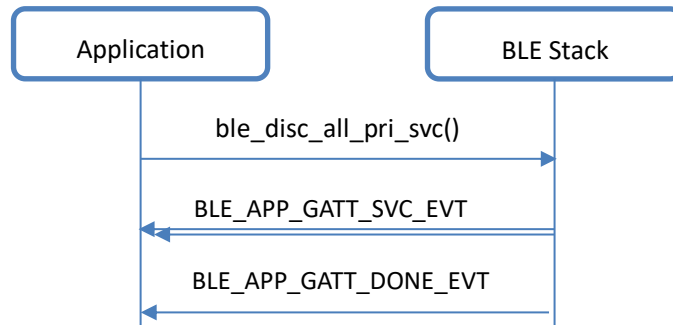
コネクションが確立されると、アプリケーションはアトリビュートの読み込みや書き込みなどの GATT クライアントの手続きを行うことができます。後述の 6.8 GATT クライアントを参照してください。API : `ble_cls_con0`を使用することで、任意のタイミングで確立されたコネクションを切断することができます。

本 BLE スタックは、複数のコネクション確立をサポートしていますが、一度に開始できるのは 1 つのコネクションのみです。サポートされる最大コネクション数は、使用する BLE コントローラに依存、メモリプールやコネクションバッファなど使用するリソースの制限などに依存します。また、`CFG_BLE_MAX_CON`(※詳細についてはチュートリアルガイドを参照)を超える数のコネクションを確立することはできません。

6.8 GATT クライアント

ペリフェラルデバイスのアトリビュートにアクセスするためには、そのアトリビュートのハンドル値を知る必要があります。アプリケーションはアトリビュート値を取得するために、以下の手順を実行する必要があります。

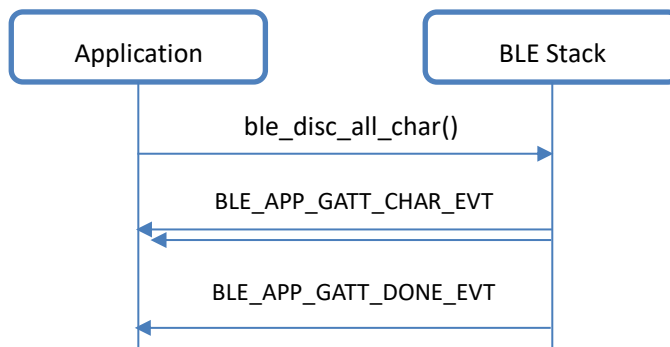
6.8.1. サービス検出



- (1) `ble_disc_all_pri_svc()`
全サービスの検出を開始します。
- (2) `BLE_APP_GATT_SVC_EVT`
検出したサービスアトリビュートの開始ハンドル値、終了ハンドル値及びサービスの UUID を通知します。
- (3) `BLE_APP_GATT_DONE_EVT`
すべてのサービスの検出が完了したことを通知します。なお、サービスの検出に対してペリフェラルがエラー応答を返した場合、`BLE_APP_GATT_ERR_EVT` が通知されます。

なお、他のサービス検出 API として、`ble_disc_inc_svc()` と `ble_disc_all_pri_svc_by_uuid()` があります。`ble_disc_inc_svc()` は、インクルードサービスの検出に使用します。`ble_disc_all_pri_svc_by_uuid()` は、特定の UUID を指定したサービス検出に使用します。

6.8.2. キャラクタリスティック検出

(1) `ble_disc_all_char()`

指定するサービスに含まれる、すべてのキャラクタリスティックの検出を開始します。本 API のパラメータ: `top` 及び `end` には、指定サービスの開始ハンドル値と終了ハンドル値を設定してください。

(2) `BLE_APP_GATT_CHAR_EVT`

検出したキャラクタリスティックアトリビュートのハンドル値、プロパティ、キャラクタリスティック値ハンドル値及びキャラクタリスティック値 **UUID** を通知します。

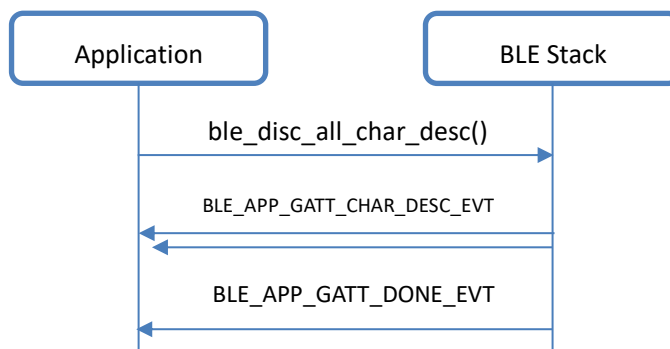
(3) `BLE_APP_GATT_DONE_EVT`

すべてのキャラクタリスティック検出が完了したことを通知します。なお、キャラクタリスティック検出に対して、ペリフェラルがエラー応答を返した場合、`BLE_APP_GATT_ERR_EVT` で通知されます。

本手順では、1つのサービスが持つキャラクタリスティックを検出できます。複数のサービスに対しては、`ble_disc_all_char()`のパラメータを適宜変更して、本手順を繰り返すことで、それぞれのキャラクタリスティックを検出できます。

他のキャラクタリスティック検出 API として、`ble_disc_all_char_by_uuid()`があります。`ble_disc_all_char_by_uuid()`は、特定の **UUID** の指定したキャラクタリスティック検出に使用します。

6.8.3. キャラクタリスティックディスクリプタ検出

(1) `ble_disc_all_char_desc()`

キャラクタリスティックディスクリプタの検出を開始します。本 API のパラメータ：
`top` 及び `end` には、指定キャラクタリスティックの `top` と `end` の値をセットしてください。

(2) `BLE_APP_GATT_CHAR_DESC_EVT`

検出したキャラクタリスティックディスクリプタの値を提供します。

(3) `BLE_APP_GATT_DONE_EVT`

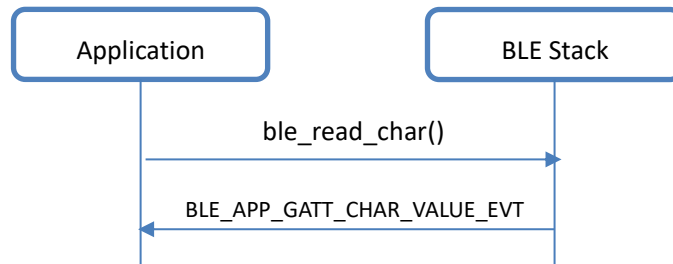
すべてのキャラクタリスティックディスクリプタ検出が完了したことを通知します。なお、キャラクタリスティックディスクリプタ検出に対して、ペリフェラルがエラー応答を返した場合、`BLE_APP_GATT_ERR_EVT` で通知されます。

本手順では、1つのキャラクタリスティックが持つキャラクタリスティックディスクリプタを検出できます。複数のキャラクタリスティックに対しては、`ble_disc_all_char_desc()`のパラメータを適宜変更して、本手順を繰り返すことで、それぞれのキャラクタリスティックディスクリプタを検出できます。

6.8.4. アトリビュートへの read / write

上記の手順によって、キャラクタリスティック値のハンドル値の取得が完了したら、ハンドル値を指定して read / write 手順が実行できます。

・読み込み



(1) ble_read_char()

キャラクタリスティック値の read 要求を開始します。

(2) BLE_APP_GATT_CHAR_VALUE_EVT

read した値を提供します。

他の read API として、ble_disc_all_char_by_uuid()、ble_read_long_char()があります。ble_disc_all_char_by_uuid()は、特定の UUID を指定したキャラクタリスティック値を read する際に使用します。ble_read_long_char()は、ATT MTU を超えるサイズの値を read する際に使用します。

・書き込み



(1) ble_write_char()

キャラクタリスティック値の write 要求を開始します。

(2) BLE_APP_GATT_DONE_EVT

値の write が完了したことを通知します。

他の write API として、ble_write_long_char()と ble_write_cmd_char()があります。ble_write_long_char()は、ATT MTU を超えるサイズの値を write する際に使用します。ble_write_cmd_char()は write が完了してもイベントが発生しません。

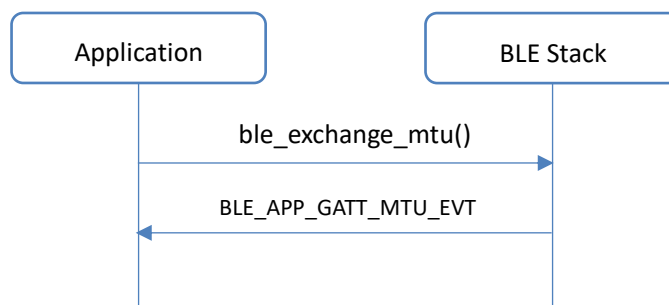
6.8.5. MTU 交換

規格により、GATT サーバと GATT クライアントは、ATT MTU の値として 23byte 以上をサポートする必要があります。両デバイスが、23byte より大きい値をサポートしている場合、ネゴシエーションにより、通信で使用する MTU のサイズを増やすことができます。

まず、クライアントからサーバへ、クライアントがサポートしている MTU 最大値を送信します。サーバがサポートしている MTU 最大値が、クライアントから受信した MTU 最大値より大きい場合、クライアントから受信した値をそのまま返します。サーバがサポートしている MTU 最大値が、クライアントから受信した MTU より小さい場合、サーバがサポートしている MTU 最大値を返します。

MTU 交換手順は、コネクション確立後のデータの送受信より前に行う必要があります。また、この手順は、コネクションの確立につき一度だけ行い、複数回繰り返してはいけません。

MTU 交換は、以下の手順に従って実行してください。



(1) ble_exchange_mtu()

MTU 交換手順を開始します。このとき、パラメータ : mtu に、使用したい MTU 値を設定してください。

(2) BLE_APP_GATT_MTU_EVT

サーバが有効な MTU 値を認識したことを通知します。有効な MTU 値は、本イベントのパラメータ : evt.mtu.val で確認することができます。

6.9 プライバシー

本 BLE スタックは、Resolvable プライベートアドレス(RPA)と Non-Resolvable プライベートアドレス(Non-RPA)をサポートしています。プライベートアドレスは、アドバタイジング、スキャンング及びイニシエーティングの際、アイデンティティアドレスとして使用することができます。

6.9.1. Resolvable プライベートアドレス (RPA)

RPA に関する基礎的な事項と、関連する API について説明します。

・ IRK (Identity Resolving Key)

RPA を生成するためには、IRK 値が必要になります。アプリケーションは、任意の IRK 値を使用するか、API : `ble_gen_irk()`を使用して生成した IRK 値を使用することができます。

ピアデバイスの RPA を解決するためには、そのデバイスの IRK 値が必要です。ピアデバイスの IRK 値は、ペアリングプロセスの際に取得することができます。

・ Resolving リスト

Resolving リストは、ローカルデバイスとピアデバイスの IRK 値で構成されるリストです。このリストは、BLE コントローラが RPA を生成するもしくは RPA を解決するために使用されます。アプリケーションは、API : `ble_cfg_resolve()`を使用して Resolving リストを設定することができます。

・ RPA 生成

アドバタイジングの際 RPA を使用する場合は、以下の手順を実行します。

- (1) API : `ble_cfg_resolve` を使用して、ローカル IRK 値を Resolving リストにセットします。このとき、ローカル IRK の値は、パラメータ : `local_irk` にセットしてください。その他のパラメータには 0 をセットしてください。
- (2) デバイスが RPA を使用する場合は、アイデンティティアドレスとしてパブリックアドレスもしくはスタティックアドレスを使用する必要があります。API : `ble_adv_cfg()`のパラメータ : `own_addr_type` に、以下のいずれかをセットして実行してください。
 - ・ BLE_OWN_ADDR_TYPE_PUBLIC_IRK – アイデンティティアドレスとしてパブリックアドレスを使用します。
 - ・ BLE_OWN_ADDR_TYPE_RANDOM_IRK – アイデンティティアドレスとしてスタティックアドレスを使用します。

- (3) API : `ble_ena_adv()`を使用してアドバタイジングが有効になると、デバイスはアドバタイジングパケットの送信を開始します。このときデバイスアドレスの値は、BLE コントローラによって自動的に生成されます。またアドレスの値は、`CFG_BLE_RPA_TMO_SEC` 秒毎に自動的に更新されます。
- (4) RPA の使用を停止するためには、API : `ble_cfg_resolve()`を、パラメータ : `list_cnt` に 0 をセットして実行してください。Resolving リストがクリアされ、RPA の生成が停止されます。

・ RPA 解決

ピアデバイスから RPA を受信した際、アプリケーションは以下の手順のいずれかを実行して、RPA を解決することができます。

A) Resolving リストを使用する。

- (1) API : `ble_cfg_resolve()`を使用して、Resolving リストにピアデバイスのアイデンティティアドレス、ピアデバイスの IRK 値及びローカルデバイスの IRK 値をセットします。
- (2) BLE コントローラは、ピアデバイスから RPA を受信するたびに、Resolving リストの情報を使用して、アドレスを自動的に解決します。

この手順では、ホワイトリストを使用することができます。BLE コントローラは RPA を解決した後、ホワイトリストをチェックしてフィルタリングすることができます。

Resolving リストのサイズは、BLE コントローラ毎に異なる可能性があります。また、BLE コントローラによっては、Resolving リストをサポートしていない場合があることに注意してください。

B) API : `ble_chk_addr_resolvable()`を使用する。

- (1) 受信したアドバタイズパケットのアドレスタイプが解決可能か、つまり「アドレスタイプが `BLE_PEER_ADDR_TYPE_RANDOM`」かつ「アドレスの MSB が RPA フォーマット」であることを確認します。
- (2) API : `ble_chk_addr_resolvable()`を、パラメータ : `p_peer_addr` には受信したアドレス値、`p_peer_irk` にはピアデバイスの IRK 値をセットして実行してください。API の戻り値が `E_OK` であれば、IRK 一致(RPA 解決)です。
- (3) IRK が一致するまで、手順(2)を繰り返し実行することができます。

この手順では、アドレスの解決はホストにて実行されるため、上記の手順は受信したアドバタイジングレポートに対してのみ実行することができます。スキャン要求や、コネクション開始要求に対しては実行できません。また、ホワイトリストを使用することもできません。

6.9.2. Non-Resolvable プライベートアドレス (Non-RPA)

Non-RPA は、アドバタイジングやスキャンングで使用できますが、Non-Connectable モードにおいてのみ使用できます。

アドバタイジングの際 Non-RPA を使用する場合は、以下の手順を実行してください。

- (1) Non-RPA を使用するためには、RPA が使用されていないことを確認してください。RPA が使用されている場合は、API : `ble_cfg_resolve0` を、パラメータ : `list_cnt` に 0 をセットして実行し、RPA を停止してください。
- (2) API : `ble_adv_cfg0` を、パラメータ : `own_addr_type` に `BLE_OWN_ADDR_TYPE_RANDOM_IRK` をセットして実行してください。
- (3) API : `ble_ena_adv0` を実行してアドバタイジングが有効になると、デバイスは、Non-RAP でアドバタイジングパケットの送信を開始します。アドレスの値は、BLE スタックによって自動生成されます。また、CFG_BLE_RPA_RMO_SEC 秒毎に自動で更新されます。

また、スキャンングの際も Non-RPA を使用することができます。API : `ble_ena_scan0` を、パラメータ : `own_addr_type` に `BLE_OWN_ADDR_TYPE_RANDOM_IRK` をセットして実行してください。

Bluetooth の仕様では、Non-RPA の使用における特定のアドレスタイプは規定されていません。本 BLE スタックでは、Resolving リストがセットされておらずかつ API : `ble_cfg_adv0` もしくは `ble_ena_scan` 実行時に、パラメータ : `own_addr_type` に `BLE_OWN_ADDR_TYPE_RANDOM_IRK` がセットされたときのみ、Non-RPA を使用します。

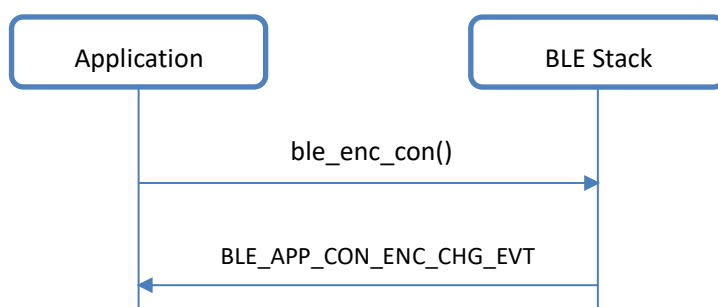
スキャンングで Non-RPA のアドバタイジングレポートを受信した場合、アドレスタイプには `BLE_OWN_ADDR_TYPE_RANDOM` がセットされます。

6.10 暗号化

セントラルデバイスは、ペリフェラルデバイスに対して暗号化の要求を開始することができます。ペリフェラルデバイスは、それに応答することができます。接続の暗号化には、LTK (Long Term Key) 値が必要になります。通常、LTK 値は、ペアリング手順において計算されます。

6.10.1. セントラル

ペリフェラルデバイスとの接続確立後、以下の手順で接続の暗号化を実行します。



(1) ble_enc_con()

セントラルデバイスの BLE コントローラに対して、指定した LTK 値で、接続暗号化の開始を要求します。セントラルデバイスの BLE コントローラは、ペリフェラルデバイスの BLE コントローラに対し、LTK の交換を要求します。

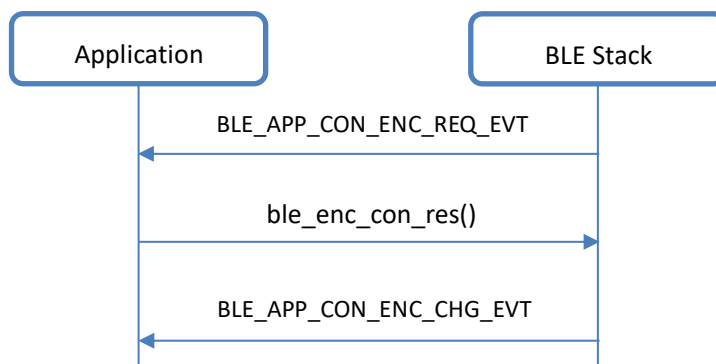
(2) BLE_APP_CON_ENC_CHG_EVT

接続の暗号化処理が完了したことを通知します。暗号化が成功したかどうかは、BLE_APP_CON_ENC_CHG_EVT のパラメータ : enc_enabled を参照する必要があります。

本手順を開始した時点で既に接続が暗号化されている場合、BLE_APP_CON_ENC_CHG_EVT の代わりに BLE_APP_CON_ENC_KEY_REF_EVT が発生し、暗号化状態が更新されたことが通知されます。

6.1 0.2. ペリフェラル

セントラルデバイスとの接続確立後、以下の手順で接続の暗号化を実行します。



(1) **BLE_APP_CON_EVT_ENC_REQ_EVT**

ペリフェラルデバイスの BLE コントローラから、接続の暗号化に使用する LTK 値の要求を受信したことを通知します。

(2) **ble_enc_con_res()**

ペリフェラルデバイスの BLE コントローラに対して、接続の暗号化に使用する LTK 値を送信します。ペリフェラルデバイスの BLE コントローラは、セントラルデバイスの BLE コントローラからの LTK 交換要求に対して、ble_enc_con_res()のパラメータで指定された LTK 値で応答します。

(3) **BLE_APP_CON_ENC_CHG_EVT**

接続の暗号化が完了したことを通知します。

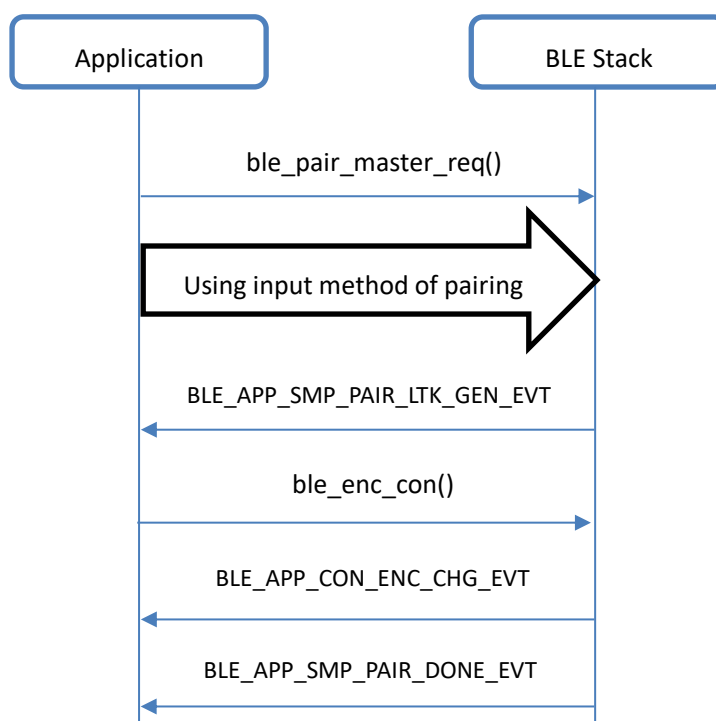
6.1 1 ペアリング

ペアリング手順は、ピアデバイスの IRK とアイデンティティアドレスの受信、LTK 値の計算、LTK の交換、接続の暗号化を実行することで完了します。セントラルデバイスはペアリング要求手順を開始することができ、ペリフェラルデバイスはペアリング要求に応答します。

本 BLE スタックは、LE セキュアペアリングのみをサポートしているため、相手デバイスも同様のペアリングをサポートしている必要があります。

6.1 1.1. セントラル

セントラルデバイスは、ペリフェラルデバイスとの接続が確立した後に、下記のペアリング手順を実行します。



(1) `ble_pair_master_req()`

ペアリング要求を開始します。

(2) Using input method of pairing

`ble_pair_master_req()` で設定したパラメータに応じて、以下に示すイベント(ペアリング時の入力方法イベント)のいずれかが発生します。アプリケーションは、発生したイベントに対する応答を実行する必要があります。

イベント	アプリケーションが実行すべき手順
BLE_APP_SMP_PAIR_KEY_PRESS_EVT	API:ble_pair_user_input()を実行します。 このとき、パラメータ: pk_param->key_press に、適切な値をセットしてください。セットすべき値については、7.10.4を参照してください。
BLE_APP_SMP_PAIR_NUM_DISP_EVT	API:ble_pair_user_input()を実行します。 このとき、パラメータ:pk_param->yes_no には、1 もしくは 0 をセットし、入力された数値に対する承認もしくは拒否を行います。ピアデバイスに表示されている数値と一致している場合 1 をセットし、承認します。
BLE_APP_SMP_PAIR_PK_INPUT_EVT	API:ble_pair_user_input()を実行します。 このとき、パラメータ: pk_param->pass_key に、ピアデバイスのパスキー値をセットします。
BLE_APP_SMP_PAIR_PK_DISPLAY_EVT	API:ble_pair_user_input()を実行します。 このとき、パラメータ pk_param->pass_key に、ローカルデバイスのパスキー値をセットします。また、デバイスにパスキー値を表示します。
BLE_APP_SMP_PAIR_OOB_INPUT_EVT	API:ble_pair_user_input()を実行します。 このとき、パラメータ: pk_param->oob に、リモートデバイスの OOB データをセットします。

(3) BLE_APP_SMP_PAIR_LTK_GEN_EVT

LTK が生成されたことを通知します。

(4) ble_enc_con()

コネクション暗号化の要求を開始します。

(5) BLE_APP_CON_ENC_CHG_EVT

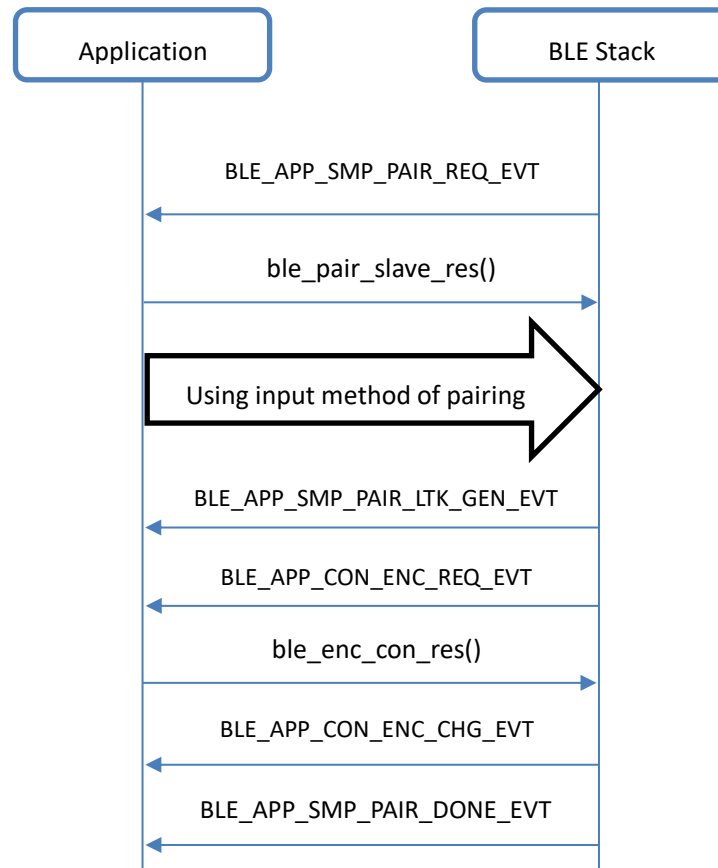
コネクションの暗号化が完了したことを通知します。

(6) BLE_APP_SMP_PAIR_DONE_EVT

ペアリングが完了したことを通知します。

6.1 1.2. ペリフェラル

ペリフェラルデバイスは、セントラルデバイスとの接続が確立した後に、下記のペアリング手順を実行します。



(1) `BLE_APP_SMP_PAIR_REQ_EVT`

セントラルデバイスからペアリングの要求があったことを通知します。

(2) `ble_pair_slave_res()`

ペアリング要求に対する応答を送信します。ペアリング要求を拒否したい場合は、本APIのパラメータ：`status`に理由を設定して実行することで、要求を拒否することができます。

(3) Using input method of pairing

`ble_pair_master_req()`で設定したパラメータに応じたイベントが発生します。アプリケーションは、イベントに対する応答を実行する必要があります。各イベントについては、前項の表を参照してください。

(4) `BLE_APP_SMP_PAIR_LTK_GEN_EVT`

LTK が生成されたことを通知します。

(5) BLE_APP_CON_ENC_REQ_EVT

コネクションの暗号化に使用する LTK 値の要求を受信したことを通知します。

(6) ble_enc_con_res()

ペリフェラルデバイスの BLE コントローラに対して、コネクションの暗号化に使用する LTK 値を送信します。

(7) BLE_APP_CON_ENC_CHG_EVT

コネクションの暗号化が完了したことを通知します。

(8) BLE_APP_SMP_PAIR_DONE_EVT

ペアリングが完了したことを通知します。

6.1 1.3. ペリフェラルデバイスからのセキュリティ要求

ペリフェラルデバイスは、API : ble_pair_slave_req()を使用して、セントラルデバイスに対して、「暗号化やペアリングの開始を要求すること」を要求することができます。ble_pair_slave_req()の実行後、BLE_APP_CON_ENC_REQ_EVTを受信した場合は暗号化処理、BLE_APP_SMP_PAIR_REQ_EVTを受信した場合はペアリング処理を開始することができます。

6.1 1.4. ボンディング

本 BLE スタックには、LTK、IRK 及びアイデンティティアドレスといったペアリングに必要な情報を保存する機能はありません。アプリケーションで実装をお願いします。

API:ble_get_bond_by_con()を使用して、ボンディングリストから、指定されたコネクションのボンディング情報を検索することができますが、この場合もボンディングリストはアプリケーションで用意する必要があります。

7 API

本章で説明する API 関数はタスクコンテキストでの利用を前提としています。従って、割り込み、周期ハンドラなどの非タスクコンテキストから実行しないでください。

表 7-1 API 一覧

種類	関数名	機能概要
デバイス API	ble_ini	BLE スタック初期化
	ble_ext	BLE スタック終了
	ble_get_dev_addr	デバイスアドレス取得
	ble_set_dev_addr	デバイスアドレス設定
	ble_add_whitelist	ホワイトリスト追加
	ble_rmv_whitelist	ホワイトリスト削除
	ble_clr_whitelist	ホワイトリスト全削除
	ble_gen_rnd	乱数生成
アドバタイズ API	ble_cfg_adv	アドバタイズパケット設定
	ble_ena_adv	アドバタイズパケット送信開始
	ble_dis_adv	アドバタイズパケット送信停止
	ble_set_adv_dat	アドバタイズパケットデータ設定
	ble_set_adv_scan_rsp_dat	スキャンレスポンスデータ設定
スキャン API	ble_ena_scan	スキャン開始
	ble_dis_scan	スキャン停止
GAP, GATT プロファイル API	ble_set_gap_srv_device_name	デバイス名キャラクタリスティック 設定
	ble_set_gap_srv_appearance	アピアランスキャラクタリスティック 設定
イニシエータ API	ble_cre_con	コネクション開始
	ble_cls_con	コネクション切断
	ble_upd_con	コネクションパラメータ更新
	ble_enc_con	コネクションの暗号化要求
	ble_enc_con_res	コネクション暗号化に使用する LTK 値の応答
	ble_get_enc_state	コネクション暗号化状態確認
L2CAP API	ble_l2cap_cmd_con_upd_req	コネクションパラメータ更新要求コ マンド送信
	ble_l2cap_cmd_con_upd_res	コネクションパラメータ更新要求に 対する応答コマンド送信

種類	関数名	機能概要
GATT サービス API	ble_add_srv	サービス定義追加プロセス開始
	ble_add_inc_srv	インクルードサービス定義追加
	ble_add_char	キャラクターリスティック定義追加
	ble_add_char_val	キャラクターリスティック値宣言追加
	ble_add_char_desc_ext	拡張プロパティキャラクターリスティックディスクリプタ追加
	ble_add_char_desc_user	キャラクターリスティックユーザーディスクリプタ追加
	ble_add_char_desc_cli	クライアントキャラクターリスティック構成追加
	ble_add_char_desc_ser	サーバキャラクターリスティック構成追加
	ble_add_char_desc_app	キャラクターリスティック表示フォーマットディスクリプタ追加
	ble_add_char_desc_agg	キャラクターリスティックアグリゲートフォーマットディスクリプタ追加
	ble_add_char_desc_cus	キャラクターリスティックカスタムディスクリプタ追加
	ble_reg_srv	サービス定義を登録

種類	関数名	機能概要
GATT プロシージャ API	ble_exchange_mtu	MTU の設定
	ble_disc_all_pri_svc	プライマリサービスを検出
	ble_disc_all_pri_svc_by_uuid	UUID を指定して、特定のプライマリサービスを検出
	ble_disc_inc_svc	インクルードサービスを検出
	ble_disc_all_char	キャラクタースティックを検出
	ble_disc_all_char_by_uuid	UUID を指定して、特定のキャラクタースティックを検出
	ble_disc_all_char_desc	ディスクリプタを検出
	ble_read_char	ハンドルを指定して、キャラクタースティック値を読み取る
	ble_read_long_char	ハンドルを指定して、大きいサイズキャラクタースティック値を読み取る
	ble_read_char_by_uuid	UUID を指定して、キャラクタースティック値を読み取る
	ble_write_char	ハンドルを指定して、キャラクタースティック値を書き込む
	ble_write_cmd_char	ハンドルを指定して、キャラクタースティック値を応答なしで書き込む
	ble_write_long_char	ハンドルを指定して、大きいサイズのキャラクタースティック値を書き込む
	ble_notify	ハンドルを指定して、キャラクタースティックに notify ビットを設定する
	ble_indicate	ハンドルを指定して、キャラクタースティックに indicate ビットを設定する
プライバシー API	ble_gen_irk	IRK 値の生成
	ble_cfg_resolve	Resolving リストの設定
	ble_chk_addr_resolvable	受信 RPA が既知デバイスのものか確認
ペアリング API	ble_pair_master_req	セントラルデバイスからのペアリング開始要求
	ble_pair_slave_res	ペリフェラルデバイスのペアリング応答
	ble_pair_slave_req	ペリフェラルデバイスからのペアリング開始要求
	ble_pair_user_input	ペアリング入力方法イベントに対する応答
	ble_gen_oob_dat	OOB データ生成
	ble_get_bond_by_con	ボンディング情報の検索

7.1 デバイス API

7.1.1. ble_ini

(BLE スタック初期化)

【書式】

```
ER ble_ini(T_BLE_INI_CFG *cfg);
```

【パラメータ】

T_BLE_INI_CFG *	cfg	BLE コンフィグレーション構造体
-----------------	-----	-------------------

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_ILUSE	サービスコール不正使用
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

BLE スタックを初期化します。この API は、他の API を呼び出す前に必ず呼び出す必要があります。

T_BLE_INI_CFG パラメータ

型	フィールド	説明
FP_BLE_APP_EVT_CBK	app_cbk	BLE スタックのアプリケーションコールバック関数です。BLE スタックはこの関数を呼び出して、BLE イベントを通知します。

アプリケーションコールバック関数は、以下で定義されます。

```
typedef void (* FP_BLE_APP_EVT_CBK)(T_BLE_APP_EVENT *);
```

T_BLE_APP_EVENT パラメータ

型	フィールド	説明
ENUM_BLE_APP_EVG	group	イベントグループ値。後述の 7.1 1 を参照してください。
int	type	イベントグループタイプ。後述の 7.1 1 を参照してください。
UB *	param	イベントパラメータへのポインタ。構造はイベントタイプによって異なります。
UH	param_len	イベントパラメータ長。

7.1.2. ble_ext

(BLE スタック終了)

【書式】

ER ble_ext(void);

【パラメータ】なし

【戻り値】

ER

正常終了(E_OK)またはエラーコード

【エラーコード】

E_OBJ

オブジェクト状態エラー

【解説】

BLE スタックを終了します。

7.1.3. ble_get_dev_addr

(デバイスアドレス取得)

【書式】

```
ER ble_get_dev_addr(T_BLE_ADDR *addr, UB *ble_error);
```

【パラメータ】

T_BLE_ADDR *	addr	デバイスアドレス
UB *	ble_error	BLE エラー

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_ILUSE	サービスコール不正使用
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

BLE コントローラのデバイスアドレスを取得します。addr->addr_type に BLE_OWN_ADDR_TYPE_PUBLIC をセットした場合、パブリックデバイスアドレス、BLE_OWN_ADDR_TYPE_RANDOM をセットした場合、ランダムスタティックデバイスアドレスを取得します。

T_BLE_ADDR パラメータ

型	フィールド	説明
UB	addr_type	アドレスタイプ
UB	addr[6]	アドレス値

addr_type パラメータ

パラメータ	説明
BLE_OWN_ADDR_TYPE_PUBLIC	パブリックデバイスアドレス
BLE_OWN_ADDR_TYPE_RANDOM	ランダムスタティックデバイスアドレス

7.1.4. ble_set_dev_addr

(ランダムスタティックデバイスアドレス設定)

【書式】

```
ER ble_set_dev_addr(T_BLE_ADDR *addr, UB *ble_error);
```

【パラメータ】

T_BLE_ADDR *	addr	デバイスアドレス
UB *	ble_error	BLE エラー

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_ILUSE	サービスコール不正使用
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

ランダムスタティックデバイスアドレスをセットします。本 API 実行時は、addr->addr_type に必ず BLE_OWN_ADDR_TYPE_RANDOM をセットする必要があります。また、セットするランダムスタティックデバイスアドレスの値は、Bluetooth core specification の規定に従う必要があります。

7.1.5. ble_add_whitelist

(ホワイトリスト追加)

【書式】

```
ER ble_add_whitelist(T_BLE_ADDR *addr, UB *ble_error);
```

【パラメータ】

T_BLE_ADDR *	addr	デバイスアドレス
UB *	ble_error	BLE エラー

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_ILUSE	サービスコール不正使用
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

ピアデバイスアドレスを、BLE コントローラのホワイトリストに追加します。ホワイトリストのサイズは、使用する BLE コントローラによって異なることに注意してください。

本 API は、アドバタイジング状態、スキャン状態及びコネクションの確立が進行中の場合は使用できません。

7.1.6. ble_rmv_whitelist

(ホワイトリスト削除)

【書式】

ER ble_rmv_whitelist(T_BLE_ADDR *addr, UB *ble_error);

【パラメータ】

T_BLE_ADDR *	addr	デバイスアドレス
UB *	ble_error	BLE エラー

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_ILUSE	サービスコール不正使用
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

ピアデバイスアドレスを、BLE コントローラのホワイトリストから削除します。

本 API は、アドバタイジング状態、スキャン状態及びコネクションの確立が進行中の場合は使用できません。

7.1.7. ble_clr_whitelist

(ホワイトリスト全削除)

【書式】

ER ble_clr_whitelist(UB *ble_error);

【パラメータ】

UB *	ble_error	BLE エラー
------	-----------	---------

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_ILUSE	サービスコール不正使用
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

ホワイトリストのピアデバイスアドレスをすべて削除します。

本 API は、アダプタイジング状態、スキャンング状態及び接続の確立が進行中の場合は使用できません。

7.1.8. ble_gen_rnd

(乱数生成)

【書式】

ER ble_gen_rnd(UB *p_rnd, UB len);

【パラメータ】

UB *	p_rnd	乱数格納先
UB	len	乱数長

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_ILUSE	サービスコール不正使用
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

指定した長さの乱数を生成します。BLE コントローラに対して HCI_LE_Rand コマンドを要求することで乱数を生成します。

7.2 アドバタイズ API

7.2.1. ble_cfg_adv

(アドバタイズパラメータ設定)

【書式】

```
ER ble_cfg_adv(UB adv_disc_mode, UB adv_conn_mode,
               T_BLE_ADV_CFG *cfg, UB *ble_error);
```

【パラメータ】

UB	adv_disc_mode	デイスカバリーモード設定
UB	adv_conn_mode	コネクションモード設定
T_BLE_ADV_CFG *	cfg	アドバタイズ設定
UB *	ble_error	BLE エラー

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_ILUSE	サービスコール不正使用
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

アドバタイズのパラメータを設定します。ble_ena_adv()を実行して、アドバタイズパケットの送信を行う前に、本 API を必ず実行してください。

adv_disc_mode パラメータ

パラメータ	説明
BLE_DISC_MODE_NON	他のデバイスから検出できないモード
BLE_DISC_MODE_LIM	限られた期間のみ検出できるモード AD フラグに”LE Limited Discoverable Mode”がセットされます
BLE_DISC_MODE_GEN	常に検出できるモード AD フラグに”LE General Discoverable Mode”がセットされます

adv_conn_mode パラメータ

パラメータ	説明
BLE_CONN_MODE_NON	他のデバイスがコネクションできないモード
BLE_CONN_MODE_NON_SCAN	他のデバイスがスキャンできない無向アドバタイズモード(※1)
BLE_CONN_MODE_UND	他のデバイスがスキャン可能かつスキャン可能な無向アドバタイズモード
BLE_CONN_MODE_LOW_DIR	他のデバイスがコネクション可能な、低デューティサイクルの有向アドバタイズモード(※2)
BLE_CONN_MODE_HIGH_DIR	他のデバイスがコネクション可能な、高デューティサイクルの有向アドバタイズモード

※1：無向アドバタイズモードは、不特定多数のデバイスからのコネクションを受け入れます。

※2：有向アドバタイズモードは、特定のデバイスからのコネクションのみを受け入れます。

T_BLE_ADV_CFG パラメータ

型	フィールド	説明
UH	adv_int_min	アドバタイズパケット送信の最小間隔。adv_int_max より大きい値を設定することはできません。0を設定した場合、BLE スタックは自動的に適切な値に構成します。設定できる値の範囲は、BLE_ADV_INT_MIN ～ BLE_ADV_INT_MAX です。デフォルト値は BLE_ADV_INT_DEFAULT です。
UB	adv_int_max	アドバタイズパケット送信の最大間隔。adv_int_min より小さい値を設定することはできません。0を設定した場合、BLE スタックは自動的に適切な値に構成します。設定できる値の範囲は、BLE_ADV_INT_MIN ～ BLE_ADV_INT_MAX です。デフォルト値は BLE_ADV_INT_DEFAULT です。
UB	own_addr_type	ローカルデバイスのアドレスタイプ。 アドバタイズパケットで使用するアドレスタイプを指定します。
UB	peer_addr_type	ピアデバイスのアドレスタイプ。 有効アドバタイズモードを使用する際、指定する必要があります。
UB	peer_addr[6]	ピアデバイスのアドレス。 有効アドバタイズモードを使用する際、指定する必要があります。
UB	adv_ch_map	アドバタイズパケットを送信するチャンネル。
UB	adv_filter	アドバタイズのフィルター。

adv_int_min、adv_int_max パラメータ

アドバタイズパケットの送信間隔を指定するパラメータです。設定した値に、0.625ms をかけた値が使用されます。実際の送信間隔は、adv_int_min、adv_int_max の値と、他のパラメータに基づき、BLE コントローラによって自動的に設定されます。

BLE スタックは、デフォルト値として以下の値を提供しています。

パラメータ	説明
BLE_ADV_INT_DEFAULT	0x0800 (× 0.625ms = 1.28 s)
BLE_ADV_INT_MIN	0x0020 (× 0.625ms = 20 ms)
BLE_ADV_INT_MAX	0x4000 (× 0.625ms = 10.24 s)
BLE_ADV_FAST_INT1_MIN	0x0030 (× 0.625ms = 30 ms)
BLE_ADV_FAST_INT1_MAX	0x0060 (× 0.625ms = 60 ms)
BLE_ADV_FAST_INT2_MIN	0x00A0 (× 0.625ms = 100 ms)
BLE_ADV_FAST_INT2_MAX	0x00F0 (× 0.625ms = 150 ms)

own_addr_type、peer_addr_type パラメータ

パラメータ	説明
BLE_PEER_ADDR_TYPE_PUBLIC	パブリックデバイスアドレス
BLE_PEER_ADDR_TYPE_RANDOM	ランダムデバイスアドレス
BLE_OWN_ADDR_TYPE_PUBLIC_IRK	アイデンティティアドレスにパブリックアドレスを使用した Resolvable プライベートアドレス
BLE_OWN_ADDR_TYPE_RANDOM_IRK	アイデンティティアドレスにスタティックアドレスを使用した Resolvable プライベートアドレス。 もしくは Non-Resolvable プライベートアドレス

※RPA を使用するには、ble_cfg_resolve()を使用して resolving リストを設定する必要があります。

adv_ch_map パラメータ

パラメータ	説明
BLE_ADV_CH_MAP_37	ch37 を使用します
BLE_ADV_CH_MAP_38	ch38 を使用します
BLE_ADV_CH_MAP_39	ch39 を使用します
BLE_ADV_CH_MAP_ALL	ch37, 38, 39 すべてを使用します。

adv_filter パラメータ

パラメータ	説明
BLE_ADV_FILTER_ALL	すべてのデバイスからのスキャン要求及びコネクション要求を処理します
BLE_ADV_FILTER_WHITE_SCAN	すべてのデバイスからのスキャン要求を処理します。また、ホワイトリストのデバイスからのコネクション要求のみを処理します
BLE_ADV_FILTER_WHITE_CON	すべてのデバイスからのコネクション要求を処理します。また、ホワイトリストのデバイスからのスキャン要求のみを処理します
BLE_ADV_FILTER_WHITE	ホワイトリストのデバイスからのスキャン要求及びコネクション要求を処理します

自動的に設定されるパラメータ

adv_disc_mode と adv_conn_mode にセットした値に応じて、一部のパラメータは自動的に設定されます。

adv_disc_mode	自動設定値
BLE_DISC_MODE_NON	・アドバタイズデータの AD フラグには何もセットされません。
BLE_DISC_MODE_LIM	・adv_filter には BLE_ADV_FILTER_ALL がセットされます。 ・AD フラグに、LE Limited Discoverable Mode と BR/EDR Not Supported がセットされます。 ・規定により、アドバタイジングは 180 秒経過でタイムアウトします。
BLE_DISC_MODE_GEN	・adv_filter には BLE_ADV_FILTER_ALL がセットされます。 ・AD フラグに、LE General Discoverable Mode と BR/EDR Not Supported がセットされます。

adv_conn_mode	自動設定値
BLE_CONN_MODE_NON BLE_CONN_MODE_NON_SCAN	・adv_int_min に BLE_ADV_FAST_INT2_MIN、adv_int_max に BLE_ADV_FAST_INT2_MAX がセットされます。
BLE_CONN_MODE_UND	・adv_int_min に BLE_ADV_FAST_INT1_MIN、adv_int_max に BLE_ADV_FAST_INT1_MAX がセットされます。
BLE_CONN_MODE_LOW_DIR BLE_CONN_MODE_HIGH_DIR	・アドバタイズパケットの間隔は BLE コントローラによって設定されるため、設定値は無視されます。

7.2.2. ble_ena_adv

(アドバタイズパケット送信開始)

【書式】

ER ble_ena_adv(UW adv_period_in_sec, UB *ble_error);

【パラメータ】

UW	adv_period_in_sec	アドバタイズパケット送信タイムアウト値
UB *	ble_error	BLE エラー

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_ILUSE	サービスコール不正使用
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

アドバタイズパケットの送信を開始します。

adv_period_in_sec に設定した時間がすると、アドバタイズパケットの送信を停止します。adv_period_in_sec に 0 を設定した場合は、アドバタイズパケットを送信し続けます。タイムアウトは、BLE_APP_DEV_TER_EVT によって通知されます。また、コネクションイベントが発生した場合、adv_period_in_sec に設定した値に関わらず、アドバタイズパケットの送信は停止します。

7.2.3. ble_dis_adv

(アドバタイズパケット送信終了)

【書式】

ER ble_dis_adv(UB *ble_error);

【パラメータ】

UB *	ble_error	BLE エラー
------	-----------	---------

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_ILUSE	サービスコール不正使用
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

アドバタイズパケットの送信を停止します。

7.2.4. ble_set_adv_dat

(アドバタイズデータ設定)

【書式】

 ER ble_set_adv_dat(UB *dat, UB len, UB *ble_error);

【パラメータ】

UB *	dat	アドバタイズデータ
UB	len	アドバタイズデータサイズ
UB *	ble_error	BLE エラー

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_ILUSE	サービスコール不正使用
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

AD フォーマットで指定された形式で、アドバタイズパケットデータを設定します。

API : ble_cfg_adv によって、adv_disc_mode パラメータが BLE_DISC_MODE_LIM または BLE_DISC_MODE_GEN に設定されている場合、BLE スタックによって AD フラグは自動的に設定されます。そのため、本 API には AD フラグを渡さないでください。

7.2.5. ble_set_adv_scan_rsp_dat

(スキャン応答データ設定)

【書式】

 ER ble_set_adv_scan_rsp_dat(UB *dat, UB len, UB *ble_error);

【パラメータ】

UB *	dat	スキャン応答データ
UB	len	スキャン応答データサイズ
UB *	ble_error	BLE エラー

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_ILUSE	サービスコール不正使用
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

AD フォーマットで指定された形式で、スキャン応答データを設定します。

7.3 スキャン API

7.3.1. ble_ena_scan

(スキャン開始)

【書式】

```
ER ble_ena_scan(T_BLE_SCAN_CFG *cfg, UB *ble_error);
```

【パラメータ】

T_BLE_SCAN_CFG *	cfg	スキャンコンフィギュレーション構造体
UB *	ble_error	BLE エラー

【戻り値】

ER 正常終了(E_OK)またはエラーコード

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_ILUSE	サービスコール不正使用
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

アドバタイズパケットのスキャンに関わるパラメータを設定し、アドバタイズパケットのスキャンを開始します。BLE_APP_ADV_EVT イベントによって、デバイスを検出したことが通知されます。

T_BLE_SCAN_CFG パラメータ

型	フィールド	説明
UB	scan_mode	スキャンモード設定。
UB	scan_type	スキャンタイプ設定。
UH	scan_int	スキャン間隔。
UB	scan_window	スキャンウィンドウ。scan_int より小さい値を設定しないでください。
UB	own_addr_type	スキャンリクエストパケットに入れるアドレスのタイプ。
UB	scan_filter	スキャンフィルター。
UB	duplicate_filter	0 に設定した場合、同じデバイスアドレスから複数回アドバタイズパケットを受信します。 1 に設定した場合、同じデバイスアドレスからアドバタイズパケットを 1 回のみ受信します。

scan_mode パラメータ

パラメータ	説明
BLE_SCAN_MODE_OBS	コネクションレスのアドバタイズパケットを受信します。
BLE_SCAN_MODE_LIM	LE Limited Discoverable Mode のデバイスからアドバタイズパケットを受信します。
BLE_SCAN_MODE_GEN	LE Limited Discoverable Mode もしくは General Limited Discoverable Mode のデバイスからアドバタイズパケットを受信します。

scan_type パラメータ

パラメータ	説明
BLE_SCAN_TYPE_PASSIVE	デバイスからアドバタイズパケットを受信します。
BLE_SCAN_TYPE_ACTIVE	アドバタイズパケットを受信したのち、スキャンリクエストパケットを送信します。

scan_int パラメータ

スキャン間隔を設定するパラメータです。設定した値×0.625ms の間隔でアドバタイズパケットをスキャンします。

BLE スタックは、デフォルト値として以下の値を提供しています。

パラメータ	説明
BLE_SCAN_INT_DEFAULT	0x0010 (× 0.625ms = 10ms)
BLE_SCAN_INT_MIN	0x0004 (× 0.625ms = 2.5ms)
BLE_SCAN_FAST_INT_MIN	0x30 (× 0.625ms = 30ms)
BLE_SCAN_SLOW_INT1	0x66C (× 0.625ms = 1027.5ms)
BLE_SCAN_SLOW_INT2	0xCD9 (× 0.625ms = 2055.625ms)
BLE_SCAN_INT_MAX	0x4000 (× 0.625ms = 10.24ms)
BLE_SCAN_FAST_INT_MAX	0x60 (× 0.625ms = 37.5ms)

scan_window パラメータ

スキャン時間を設定するパラメータです。設定した値×0.625ms の時間、アドバタイズパケットをスキャンします。

BLE スタックは、デフォルト値として以下の値を提供しています。

パラメータ	説明
BLE_SCAN_WINDOW_DEFAULT	0x0010 (× 0.625ms = 10ms)
BLE_SCAN_WINDOW_MIN	0x0004 (× 0.625ms = 2.5ms)
BLE_SCAN_WINDOW_MAX	0x4000 (× 0.625ms = 10.24ms)
BLE_SCAN_SLOW_WINDOW1	0x44E8 (× 0.625ms = 11023.75ms)
BLE_SCAN_SLOW_WINDOW2	35280 (× 0.625ms = 22.5ms)

own_addr_type パラメータ

API : ble_cfg_adv() の own_addr_type パラメータを参照してください。

scan_filter パラメータ

パラメータ	説明
BLE_SCAN_FILTER_ACCEPT_ALL	有向アドバタイズパケットを除くすべてのアドバタイズパケットを受け入れます。
BLE_SCAN_FILTER_ACCEPT_ONLY	ホワイトリストにあるデバイスアドレスのアドバタイズパケットを受け入れます。
BLE_SCAN_FILTER_ACCEPT_2	RPA の無向及び有向アドバタイズパケットを受け入れます。
BLE_SCAN_FILTER_ACCEPT_3	ホワイトリストに登録されている RPA の無向及び有向アドバタイズパケットを受け入れます。

自動的に設定されるパラメータ

scan_mode にセットした値に応じて、一部の設定は自動的に設定されます。

scan_mode	自動設定値
BLE_SCAN_MODE_GEN BLE_SCAN_MODE_LIM	<ul style="list-style-type: none"> • scan_type に、BLE_SCAN_TYPE_ACTIVE がセットされます。 • scan_filter に、BLE_SCAN_FILTER_ACCEPT_ALL がセットされます。

7.3.2. ble_dis_scan

(スキャン終了)

【書式】

ER ble_dis_scan(UB *ble_error);

【パラメータ】

UB *	ble_error	BLE エラー
------	-----------	---------

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_ILUSE	サービスコール不正使用
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

アドバタイズパケットのスキャンを停止します。

7.4 GAP, GATT プロファイル API

7.4.1. ble_set_gap_srv_device_name

(デバイス名設定)

【書式】

```
ER ble_set_gap_srv_device_name(UB *name, UH len, UB permission);
```

【パラメータ】

UB *	name	デバイス名
UH	len	デバイス名サイズ
UB	permission	アクセス権限

以下の値を設定できます。

BLE_CHAR_PERM_RD - read 許可

BLE_CHAR_PERM_WR - write 許可

BLE_CHAR_PERM_WR_NO_RSP - 応答なし write 許可

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
-------	----------

【解説】

GAP のデバイス名キャラクターリスティックを設定します。デバイス名サイズの上限は 248 バイトです。

7.4.2. ble_set_gap_srv_appearance

(アピアランス設定)

【書式】

ER ble_set_gap_srv_appearance (UH appearance, UB permission);

【パラメータ】

UH	appearance	アピアランス 値については、Bluetooth Core Specification を参照してください
UB	permission	アクセス権限 以下の値を設定できます。 BLE_CHAR_PERM_RD - read 許可 BLE_CHAR_PERM_WR - write 許可 BLE_CHAR_PERM_WR_NO_RSP - 応答なし write 許可

【戻り値】

ER 正常終了(E_OK)またはエラーコード

【エラーコード】

E_PAR パラメータエラー

【解説】

GAP のアピアランスキャラクターリスティックを設定します。

7.5 イニシエータ API

7.5.1. ble_cre_con

(コネクション確立)

【書式】

```
ER ble_cre_con(T_BLE_CON_CFG *cfg, T_BLE_ADDR *peer, UB *ble_error);
```

【パラメータ】

T_BLE_CON_CFG *	cfg	コネクションパラメータコンフィ ギュレーション
T_BLE_ADDR *	peer	ピアデバイスアドレス
UB *	ble_error	BLE エラー

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_ILUSE	サービスコール不正使用
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

コネクション可能なアドバタイズパケットを送信しているピアデバイスをスキャンし、
cfg で設定したパラメータに従ってピアデバイスとのコネクションを確立します。

T_BLE_CON_CFG パラメータ

型	フィールド	説明
UH	scan_int	スキャン間隔。 設定可能な値の範囲は、BLE_SCAN_INT_MIN から BLE_SCAN_INT_MAX です。
UH	scan_wnd	スキャンウィンドウ。 設定可能な値の範囲は、BLE_SCAN_WINDOW_MIN から BLE_SCAN_WINDOW_MAX です。また、scan_int より小さい値を設定しないでください。 scan_int と同じ値を設定した場合は、連続スキャンになります。
UB	own_addr_type	スキャンリクエストパケットに入れるアドレスのタイプ。
UH	con_int_min	最小コネクションイベント間隔。 BLE_CON_INT_MIN から BLE_CON_INT_MAX までの値を設定してください。
UH	con_int_max	最大コネクションイベント間隔。 BLE_CON_INT_MIN から BLE_CON_INT_MAX までの値を設定してください。con_int_min より小さい値を設定しないでください。
UH	con_latency	最大コネクションレイテンシ BLE_CON_LATENCY_MIN から BLE_CON_LATENCY_MAX までの値を設定してください。
UH	supervision_timeout	スーパービジョンタイムアウト設定 BLE_SUPERVISION_TIMEOUT_MIN から BLE_SUPERVISION_TIMEOUT_MAX までの値を設定してください。また、 $((1 + \text{con_latency}) * \text{con_int_max} * 2)$ より大きい値を設定してください。
UH	ce_len_min	予想されるコネクションイベントの最小長さ。 BLE_CON_INT_MIN から BLE_CON_INT_MAX までの値を設定してください。
UH	ce_len_max	予想されるコネクションイベントの最大長さ。 BLE_CON_INT_MIN から BLE_CON_INT_MAX までの値を設定してください。 ce_len_min より小さい値を設定しないでください。

con_int_min、con_int_max パラメータ

以下の値の範囲で、コネクションイベント間隔を指定できます。設定した値 $\times 1.25\text{ms}$ の間隔で、コネクションイベントが開始されます。

パラメータ	説明
BLE_CON_INT_MIN	0x0006 ($\times 1.25\text{ms} = 7.5\text{ ms}$)
BLE_CON_INT_MAX	0x0C80 ($\times 1.25\text{ms} = 4\text{s}$)

con_latency パラメータ

以下の値の範囲で、コネクションレイテンシを設定できます。コネクションレイテンシは、スレーブデバイスがマスターデバイスに送信すべきデータが無いとき、コネクションイベントを無視できる回数です。

パラメータ	説明
BLE_CON_LATENCY_MIN	0x0000
BLE_CON_LATENCY_MAX	0x01F3

supervision_timeout パラメータ

以下の値の範囲で、スーパービジョンタイムアウトを設定できます。タイムアウト時間は設定した値×10ms の値になります。タイムアウト時間を経過してもスレーブからのパケットが届かない場合、コネクションは切断されます。

パラメータ	説明
BLE_SUPERVISION_TIMEOUT_MIN	0x000A (× 10ms = 100ms)
BLE_SUPERVISION_TIMEOUT_MAX	0x0C80 (× 10ms = 32s)

ce_len_min、ce_len_max パラメータ

以下の値の範囲で、予想されるコネクションイベントの長さを設定できます。コネクションイベントの長さは、設定した値×0.625ms の値になります。

パラメータ	説明
BLE_CE_LEN_MIN	0x0000 (× 0.625ms = 0ms)
BLE_CE_LEN_MAX	0xFFFF (× 0.625ms = 40.959ms)

7.5.2. ble_cls_con

(コネクション切断)

【書式】

 ER ble_cls_con(UH con_hnd, UB cls_reason, UB *ble_error);

【パラメータ】

UH	con_hnd	コネクションハンドル
UB	cls_reason	切断理由
UB *	ble_error	BLE エラー

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_ILUSE	サービスコール不正使用
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

コネクションハンドルを指定して、確立されたコネクションを切断します。

cls_reason に設定する値は、Bluetooth Core Specification のエラーコードを参照してください。

7.5.3. ble_upd_con

(コネクションパラメータ更新)

【書式】

ER ble_upd_con(UH con_hnd, T_BLE_CON_UPD *upd, UB *ble_error);

【パラメータ】

UH	con_hnd	コネクションハンドル
T_BLE_CON_UPD	upd	コネクションパラメータ構造体
UB *	ble_error	BLE エラー

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_NOSPT	未サポート機能
E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_ILUSE	サービスコール不正使用
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

コネクションハンドルを指定して、コネクションパラメータを更新します。本 API は、マスターデバイスとして動作しているときのみ使用可能です。

T_BLE_CON_UPD パラメータ

型	フィールド	説明
UH	con_int_min	最低コネクションイベント間隔 BLE_CON_INT_MIN から BLE_CON_INT_MAX までの 値を設定してください。
UH	con_int_max	最大コネクションイベント間隔 BLE_CON_INT_MIN から BLE_CON_INT_MAX までの 値を設定してください。con_int_min より小さい値を設定 しないでください。
UH	con_latency	最大コネクションレイテンシ BLE_CON_LATENCY_MIN から BLE_CON_LATENCY_MAX までの値を設定してくださ い。
UH	supervision_timeout	スーパービジョンタイムアウト設定 BLE_SUPERVISION_TIMEOUT_MIN から BLE_SUPERVISION_TIMEOUT_MAX までの値を設定 してください。また、 $((1 + \text{con_latency}) * \text{con_int_max} * 2)$ より大きい値を設定してください
UH	ce_len_min	コネクションイベントの最小長。 BLE_CON_INT_MIN から BLE_CON_INT_MAX までの 値を設定してください。
UH	ce_len_max	コネクションイベントの最大長。 BLE_CON_INT_MIN から BLE_CON_INT_MAX までの 値を設定してください。 ce_len_min より小さい値を設定しないでください。

7.5.4. ble_enc_con

(コネクション暗号化)

【書式】

ER ble_enc_con(UH con_hnd, T_BLE_CON_ENC *enc_info, UB *ble_error);

【パラメータ】

UH	con_hnd	コネクションハンドル
T_BLE_CON_ENC *	enc_info	暗号化パラメータ構造体
UB *	ble_error	BLE エラー

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_NOEXS	オブジェクト未生成
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

BLE コントローラに対して、指定した LTK 値でコネクション暗号化の開始を要求します。BLE コントローラは、ペリフェラルデバイスの BLE コントローラに対し、LTK の交換を要求します。暗号化処理の完了は、BLE_APP_CON_ENC_CHG_EVT イベントによって通知されます。

コネクションが既に暗号化されている状態で本 API を実行した場合、BLE_APP_CON_ENC_KEY_REF_EVT イベントが発生し、暗号化状態が新しい LTK 値で更新されたことを示します。

T_BLE_CON_ENC パラメータ

型	フィールド	説明
UB	ltk[16]	LTK 値。
UB	rnd[8]	本パラメータは使用しません。
UB	ediv[2]	本パラメータは使用しません。

7.5.5. ble_enc_con_res

(コネクション暗号化応答)

【書式】

```
ER ble_enc_con_res(UH con_hnd, T_BLE_CON_ENC *enc_info, BOOL accept,
                  UB *ble_error);
```

【パラメータ】

UH	con_hnd	コネクションハンドル
T_BLE_CON_ENC *	enc_info	暗号化パラメータ構造体
BOOL	accept	TRUE - リクエスト許可 FALSE - リクエスト拒否
UB *	ble_error	BLE エラー

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_NOEXS	オブジェクト未生成
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

本 API は、BLE_APP_CON_ENC_REQ_EVT イベントに対する応答として実行します。BLE_APP_CON_EVT_ENC_REQ_EVT は、BLE コントローラから、コネクションの暗号化に使用する LTK 値の交換要求があったことを通知します。本 API を実行することで、LTK 交換要求に対する応答を返します。

7.5.6. ble_get_enc_state

(コネクション暗号化確認)

【書式】

ER ble_get_enc_state(UH con_hnd, UB * p_enc_state)

【パラメータ】

UH	con_hnd	コネクションハンドル
UB *	p_enc_state	暗号化状態

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_NOEXS	オブジェクト未生成
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

コネクションが暗号化されているかどうかを確認します。本 API 実行後、p_enc_state の値が 1 であれば、コネクションが暗号化されています。0 の場合、コネクションは暗号化されていません。

7.6 L2CAP API

7.6.1 ble_l2cap_cmd_con_upd_req

(コネクションパラメータ更新要求コマンド)

【書式】

```
ER ble_l2cap_cmd_con_upd_req(UH con_hnd, T_BLE_L2CAP_CON_UPD_REQ
*param);
```

【パラメータ】

UH	con_hnd	コネクションハンドル
T_BLE_L2CAP_CON_UPD_REQ *	param	コネクションパラメータ構造体

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除

【解説】

コネクションハンドルを指定して、コネクションパラメータ更新要求コマンドを送信します。本 API を使用することで、ペリフェラルデバイスがセントラルデバイスに対して、コネクションパラメータの更新を要求することができます。本 API は、ペリフェラルデバイスのみ実行可能です。

T_BLE_L2CAP_CON_UPD_REQ パラメータ

型	フィールド	説明
UH	int_min	最小コネクションイベント間隔 BLE_CON_INT_MIN から BLE_CON_INT_MAX までの値を設定してください。
UH	int_max	最大コネクションイベント間隔 BLE_CON_INT_MIN から BLE_CON_INT_MAX までの値を設定してください。また、int_min より小さい値を設定しないでください。
UH	slave_latency	スレーブレイテンシ BLE_CON_LATENCY_MIN から BLE_CON_LATENCY_MAX までの値を設定してください。またこの値は、 $(\text{timeout_multiplier} * 10 / (\text{int_max} * 2) - 1)$ より小さい値を設定してください。
UH	timeout_multiplier	10 から 3200 までの値を設定してください。この値はスーパービジョンタイムアウト時間の計算に使用されます。 スーパービジョンタイムアウト時間 = $\text{timeout_multiplier} * 10[\text{ms}]$

7.6.2. ble_l2cap_cmd_con_upd_res

(コネクションパラメータ更新要求への応答コマンド)

【書式】

ER ble_l2cap_cmd_con_upd_res(UH con_hnd, UH result);

【パラメータ】

UH	con_hnd	コネクションハンドル
UH	result	応答結果

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除

【解説】

ペリフェラルデバイスから受信した、コネクションパラメータ更新要求に対する応答コマンドを返します。本 API は、セントラルデバイスのみ実行可能です。

result には、コネクションパラメータを受け入れる場合 0x0000、拒否する場合 0x0001 を渡します。

7.7 GATT サービス API

7.7.1. ble_add_srv

(サービス追加)

【書式】

```
ER ble_add_srv(T_BLE_SRV *srv);
```

【パラメータ】

T_BLE_SRV *	srv	サービスコンフィギュレーション構造体
-------------	-----	--------------------

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
-------	----------

【解説】

GATT サービス定義追加のプロセスを開始します。本 API を呼び出すことで、他の GATT サービス API を実行することができます。

API : ble_reg_srv を実行することで、サービス定義追加プロセスを完了します。

T_BLE_SRV パラメータ

型	フィールド	説明
T_BLE_UUID *	p_type	プライマリサービスまたはセカンダリサービス
T_BLE_UUID *	p_value	UUID の値

T_BLE_UUID パラメータ

型	フィールド	説明
マクロ	UUID_16BIT	16bit Bluetooth SIG サービス UUID
マクロ	UUID_32BIT	32bit Bluetooth SIG サービス UUID
マクロ	UUID_128BIT	128bit カスタムサービス UUID
UB	len	UUID 長
UH	uuid_16bit	16bit UUID 値
UW	uuid_32bit	32bit UUID 値
UB	uuid_128bit [16]	128bit UUID 値
UB	raw [16]	16Byte の生値

7.7.2. ble_add_inc_srv

(インクルードサービス定義追加)

【書式】

```
ER ble_add_inc_srv(T_BLE_SRV *srv, T_BLE_INC_SRV *inc_srv);
```

【パラメータ】

T_BLE_SRV *	srv	サービスコンフィギュレーション構造体
T_BLE_INC_SRV *	inc_srv	インクルードサービスコンフィギュレーション構造体

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
-------	----------

【解説】

指定したサービスに、インクルードサービス定義を追加します。1つのサービス定義に含めるインクルードサービス定義は1つまでです。インクルードサービス定義を追加することで、他のサービスを参照することができます。

T_BLE_INC_SRV パラメータ

型	フィールド	説明
UH	top_group_hnd	プライマリサービスの開始ハンドル
UH	end_group_hnd	プライマリサービスの終了ハンドル
T_BLE_UUID *	p_srv_uuid	サービスの UUID

7.7.3. ble_add_char

(キャラクタースティック定義追加)

【書式】

```
ER ble_add_char(T_BLE_SRV *srv, T_BLE_CHAR *chr);
```

【パラメータ】

T_BLE_SRV *	srv	サービスコンフィギュレーション構造体
T_BLE_CHAR *	chr	キャラクタースティックコンフィギュレーション構造体

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
-------	----------

【解説】

指定したサービスに、キャラクタースティック定義を追加します。キャラクタースティック定義には、続けて 1 つのキャラクタースティック値宣言と、複数のキャラクタースティックディスクリプタ宣言を追加することができます。

T_BLE_CHAR パラメータ

型	フィールド	説明
T_BLE_UUID *	p_char_uuid	キャラクタースティック UUID
UB	property	キャラクタースティックプロパティ

property パラメータ

以下の値を指定して、キャラクターリスティックに対して可能な操作及び手順を設定します。

パラメータ	説明
BLE_CHAR_PERM_BD	ブロードキャスト このキャラクターリスティックの値を、アドバタイズパケットに入れることができます。
BLE_CHAR_PERM_RD	read GATT クライアントは、このキャラクターリスティックの値を読み出すことができます。
BLE_CHAR_PERM_WR_NO_RSP	write without response GATT クライアントは、このキャラクターリスティックに値を書き込むことができます。書き込みに対する応答はありません。
BLE_CHAR_PERM_WR	write GATT クライアントは、このキャラクターリスティックに値を書き込むことができます。
BLE_CHAR_PERM_NTY	notify GATT サーバから、能動的にキャラクターリスティック値の通知を行うことができます。
BLE_CHAR_PERM_IND	indicate GATT サーバから、能動的にキャラクターリスティック値の通知を行い、更に GATT クライアントからの応答を要求します。
BLE_CHAR_PERM_EXT_PPT	extended properties キャラクターリスティックに、拡張プロパティが存在することを示します。 本パラメータを設定する場合、API : <code>ble_add_char_desc_ext</code> を事項して、キャラクターリスティック定義に拡張プロパティディスクリプタを追加する必要があります。

7.7.4. ble_add_char_val

(キャラクターリスティック値宣言追加)

【書式】

```
ER ble_add_char_val(T_BLE_CHAR *chr, T_BLE_CHAR_VALUE *val);
```

【パラメータ】

T_BLE_CHAR *	chr	キャラクターリスティックコンフィギュレーション構造体
T_BLE_CHAR_VAL *	val	キャラクターリスティック値コンフィギュレーション構造体

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
-------	----------

【解説】

指定したキャラクターリスティック定義に、キャラクターリスティック値宣言を追加します。

T_BLE_CHAR_VALUE パラメータ

型	フィールド	説明
T_BLE_UUID *	p_char_uuid	キャラクターリスティック UUID
FP_BLE_GATT_HND	cbk	キャラクターリスティック値にアクセスするためのコールバック関数。 この関数は、ピアデバイスからの read / write 要求があるたびに、BLE スタックによって呼び出されます。

キャラクターリスティック値コールバック関数は、以下で定義されます。

```
typedef UH (*FP_BLE_GATT_HND)(T_BLE_GATT_HND_PARAM *);
```

FP_BLE_GATT_HND_PARAM パラメータ

型	フィールド	説明
UH	con_hnd	コネクションハンドル値
UH	hnd	アトリビュートハンドル値
UH	type	<p>BLE_CHAR_CBK_READ - read 要求であることを示します。</p> <p>BLE_CHAR_CBK_WRITE - write 要求であることを示します。</p> <p>BLE_CHAR_CBK_WRITE_CMD - 応答なし write 要求であることを示します。</p> <p>BLE_CHAR_CBK_READ_LONG - ロングアトリビュート値 read 要求であることを示します。</p> <p>BLE_CHAR_CBK_WRITE_PRE - ロングアトリビュート値 write 要求の際に使用されます。write データのキューイングであることを示します。</p> <p>BLE_CHAR_CBK_WRITE_EXE - ロングアトリビュート値 write 要求の際に使用されます。write データキューイングの終了を示します。</p>
UB *	p_dat	<p>アプリケーションデータバッファへのポインタ。</p> <p>read 要求の場合、アプリケーションは p_dat にデータをセットして応答する必要があります。</p> <p>write 要求の場合、アプリケーションは p_dat にセットされたデータをキャラクタリスティック値に書き込む必要があります。</p>
UH	dat_len	<p>アプリケーションデータサイズ。</p> <p>read 要求の場合、応答するデータのサイズをセットする必要があります。</p> <p>write 要求の場合、受信したデータのサイズがセットされています。</p>
UH	offset	<p>アプリケーションデータのオフセット値。</p> <p>ロングアトリビュート値の read / write 要求の場合に使用します。</p>
UH	Flag	<p>BLE_CHAR_CBK_WRITE_EXE の場合に使用するフラグ。</p> <p>BLE_CHAR_CBK_WRITE_PRE イベントで受信したデータを受け入れるか、またはキャンセルするかを決定します。</p> <p>0x00 - すべての書き込みをキャンセルする。</p> <p>0x01 - ペンディングデータの書き込みを許可する。</p>
UB	error	<p>受信した要求が無効な場合、0 以外の値をセットしてください。値については、7.1 2.3 BLE ATT エラーコードを参照してください。</p>

7.7.5. ble_add_char_desc_ext

(キャラクタースティック拡張プロパティディスクリプタ追加)

【書式】

```
ER ble_add_char_desc_ext(T_BLE_CHAR *chr,
                        T_BLE_CHAR_EXT_DESC *desc);
```

【パラメータ】

T_BLE_CHAR *	chr	キャラクタースティックコンフィギュレーション構造体
T_BLE_CHAR_EXT_DESC *	desc	拡張キャラクタースティックディスクリプタコンフィギュレーション構造体

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
-------	----------

【解説】

指定したキャラクタースティック定義に、キャラクタースティック拡張プロパティディスクリプタを追加します。キャラクタースティック拡張プロパティディスクリプタは、キャラクタースティック値に対する追加のアクセス許可を指定するために使用されます。拡張プロパティディスクリプタを追加するためには、キャラクタースティック定義に、BLE_CHAR_PERM_EXT_PPT が設定されている必要があります。

T_BLE_CHAR_EXT_DESC パラメータ

型	フィールド	説明
UH	property	以下の値を指定できます。 GATT_ATTR_REL_WRITE – 信頼できる書き込み※を許可します。 GATT_ATTR_WRITE – キャラクタースティックユーザーディスクリプタへの書き込みを許可します

※2 段階の書き込み手順を実行して、書き込みを行います。

まずクライアントからサーバに **prepare write** リクエスト/レスポンスでデータを送信します。次に **execute write** リクエスト/レスポンスで、書き込みを行います。**execute write** リクエスト/レスポンスごとに、サーバはデータの正当性を確認し、エラーがあった場合はクライアントに通知します。これにより、通常の書き込み手順と比較して、データの信頼性が高い書き込みが実行できます。

7.7.6. ble_add_char_desc_user

(キャラクターリスティックユーザーディスクリプタ追加)

【書式】

```
ER ble_add_char_desc_user(T_BLE_CHAR *chr,
                          T_BLE_CHAR_USER_DESC *desc);
```

【パラメータ】

T_BLE_CHAR *	chr	ble_add_char で定義したキャラクターリスティックへのポインタ
T_BLE_CHAR_USR_DESC *	desc	キャラクターリスティックユーザーディスクリプタ構造体

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
-------	----------

【解説】

指定したキャラクターリスティック定義に、キャラクターリスティックユーザーディスクリプタを追加します。キャラクターリスティックユーザーディスクリプタは、キャラクターリスティック値の説明を、UTF-8 文字列で記述するために使用されます。

T_BLE_CHAR_USR_DESC パラメータ

型	フィールド	説明
UH	permission	以下の値を指定できます。 GATT_CHAR_PERM_RD – read を許可します。 GATT_CHAR_PERM_WR – write を許可します。 この値を設定する場合は、拡張プロパティディスクリプタで、GATT_ATTR_WRITE を設定する必要があることに注意してください。
FP_BLE_GATT_HND	cbk	キャラクターリスティック値にアクセスするためのコールバック関数。 この関数は、ピアデバイスからの read/write 要求があるたびに、BLE スタックによって呼び出されます。

7.7.7. ble_add_char_desc_cli

(クライアントキャラクタリスティック構成ディスクリプタ追加)

【書式】

```
ER ble_add_char_desc_cli(T_BLE_CHAR *chr,
                        T_BLE_CHAR_CLI_DESC *desc);
```

【パラメータ】

T_BLE_CHAR *	chr	ble_add_char で定義したキャラクタリスティックへのポインタ
T_BLE_CHAR_CLI_DESC *	desc	クライアントキャラクタリスティック構成ディスクリプタ構造体

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
-------	----------

【解説】

指定したキャラクタリスティック定義に、クライアントキャラクタリスティック構成ディスクリプタを追加します。クライアントキャラクタリスティック構成ディスクリプタは、クライアントに対する、特有の構成を定義するためのディスクリプタです。このディスクリプタは、キャラクタリスティック定義で、BLE_CHAR_PERM_IND または BLE_CHAR_PERM_NTY が設定されている場合にのみ追加します。

T_BLE_CHAR_CLI_DESC パラメータ

型	フィールド	説明
UB	permission	デフォルトで read 許可がセットされます。 write を許可する場合は、BLE_CHAR_PERM_WR を設定してください。
UH	property	以下の値を指定できます。 GATT_ATTR_NFY – キャラクタリスティック値の notify を許可します。 GATT_ATTR_IND – キャラクタリスティック値の indicate を許可します。
UH *	p_con_hnd_list	コネクションハンドル配列へのポインタ

7.7.8. ble_add_char_desc_ser

(サーバキャラクタリスティック構成ディスクリプタ追加)

【書式】

```
ER ble_add_char_desc_ser(T_BLE_CHAR *chr,
                        T_BLE_CHAR_SER_DESC *desc);
```

【パラメータ】

T_BLE_CHAR *	chr	ble_add_char で定義したキャラクタリスティックへのポインタ
T_BLE_CHAR_SER_DESC *	desc	サーバキャラクタリスティック構成ディスクリプタ

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
-------	----------

【解説】

指定したキャラクタリスティック定義に、サーバキャラクタリスティック構成ディスクリプタを追加します。サーバキャラクタリスティックディスクリプタは、サーバ特有の構成を定義するために使用します。このディスクリプタは、キャラクタリスティック定義で、BLE_CHAR_PERM_BD が設定されている場合にのみ追加します。

T_BLE_CHAR_SER_DESC パラメータ

型	フィールド	説明
UB	permission	デフォルトで read 許可がセットされます。 write を許可する場合は、BLE_CHAR_PERM_WR を設定してください。
UH	property	以下の値を指定できます。 GATT_ATTR_BROADCAST – ブロードキャストモードにおいて、アドバタイズパケットのデータに、キャラクタリスティック値が使用されます。

7.7.9. ble_add_char_desc_app

(キャラクターリスティック表示フォーマットディスクリプタ追加)

【書式】

```
ER ble_add_char_desc_app(T_BLE_CHAR *chr,
                        T_BLE_CHAR_APP_DESC *desc);
```

【パラメータ】

T_BLE_CHAR *	chr	ble_add_char で定義したキャラクターリスティックへのポインタ
T_BLE_CHAR_APP_DESC *	desc	キャラクターリスティック表示フォーマットディスクリプタ

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
-------	----------

【解説】

指定したキャラクターリスティック定義に、キャラクターリスティック表示フォーマットディスクリプタを追加します。キャラクターリスティック表示フォーマットディスクリプタは、キャラクターリスティック値のフォーマット方法を定義するために使用されます。

1つのキャラクターリスティック定義に対して、このディスクリプタを2つ以上追加するとき、同一のキャラクターリスティック定義に、キャラクターリスティックアグリゲートフォーマットディスクリプタを追加する必要があります。

T_BLE_CHAR_APP_DESC

型	フィールド	説明
UB	format	<p>キャラクタリスティック値のフォーマット。 下記リンクの Format Types を参照してください。 https://www.bluetooth.com/specifications/assigned-numbers/</p>
UB	exponent	<p>指数値。 Exponent Value が Yes のフォーマットに対して設定可能です。 キャラクタリスティック値 $\times 10^{\text{exponent}}$ として扱います。 例: キャラクタリスティック値が 23 で、exponent が 2 の場合、2300 として取り扱います。</p>
UH	unit	<p>下記リンクの 16-bit UUIDs で定義された UUID。 https://www.bluetooth.com/specifications/assigned-numbers/</p>
UB	name_space	<p>名前空間。 下記リンクの GATT Namespace Descriptors を参照してください。 https://www.bluetooth.com/specifications/assigned-numbers/</p>
UH	description	<p>説明。 下記リンクの GATT Namespace Descriptors 及び各プロファイルの仕様書を参照してください。 https://www.bluetooth.com/specifications/assigned-numbers/</p>

7.7.10. ble_add_char_desc_agg

(キャラクターリスティックアグリゲートフォーマットディスクリプタ追加)

【書式】

```
ER ble_add_char_desc_agg(T_BLE_CHAR *chr,
                        T_BLE_CHAR_AGG_DESC *desc);
```

【パラメータ】

T_BLE_CHAR *	chr	ble_add_char で定義したキャラクターリスティックへのポインタ
T_BLE_CHAR_AGG_DESC *	desc	キャラクターリスティックアグリゲートフォーマットディスクリプタ

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
-------	----------

【解説】

指定したキャラクターリスティック定義に、キャラクターリスティックアグリゲートフォーマットディスクリプタを追加します。

1 つのキャラクターリスティック定義に、2 つ以上のキャラクターリスティック表示フォーマットディスクリプタが存在する場合、このディスクリプタが必要です。このディスクリプタのアトリビュート値は、各キャラクターリスティック表示フォーマットディスクリプタのハンドルリストになります。

T_BLE_CHAR_AGG_DESC パラメータ

型	フィールド	説明
UH *	p_hnd_list	キャラクターリスティック表示フォーマットディスクリプタのハンドルリスト。
UH	hnd_cnt	ハンドルの数。

7.7.1 1. ble_add_char_desc_cus

(キャラクターリスティックカスタムディスクリプタ追加)

【書式】

```
ER ble_add_char_desc_cus(T_BLE_CHAR *chr,
                        T_BLE_CHAR_CUS_DESC *desc);
```

【パラメータ】

T_BLE_CHAR *	chr	ble_add_char で定義したキャラクターリスティックへのポインタ
T_BLE_CHAR_CUS_DESC *	desc	キャラクターリスティックカスタムディスクリプタ

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_NOMEM	メモリ不足

【解説】

指定したキャラクターリスティック定義に、キャラクターリスティックカスタムディスクリプタを追加します。このディスクリプタはオプションです。

T_BLE_CHAR_CUS_DESC パラメータ

型	フィールド	説明
T_BLE_UUID *	p_type	ディスクリプタ UUID
UB	permission	BLE_CHAR_PERM_RD - read 許可 BLE_CHAR_PERM_WR - write 許可
FP_BLE_GATT_HND	cbk	キャラクターリスティック値にアクセスするためのコールバック関数。 この関数は、ピアデバイスからの read/write 要求があるたびに、BLE スタックによって呼び出されます。

7.7.1 2. ble_reg_srv

(サービス定義登録)

【書式】

ER ble_reg_srv(T_BLE_SRV*srv);

【パラメータ】

T_BLE_SRV *	srv	ble_add_srv で定義したサービスへのポインタ
-------------	-----	-----------------------------

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
-------	----------

【解説】

サービス定義を登録し、サービス定義追加プロセスを完了します。登録が成功すると、ピアデバイスがサービスを利用できるようになります。

7.8 GATT プロシージャ API

7.8.1. ble_exchange_mtu

(MTU 交換)

【書式】

ER ble_exchange_mtu(UH con_hnd, UH mtu);

【パラメータ】

UH	con_hnd	コネクションハンドル
UH	mtu	MTU サイズ

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー

【解説】

ATT プロトコルにおいて、MTU のデフォルト値は 23 バイトですが、サーバとクライアントのどちらか、もしくはどちらもデフォルト値より大きい値をサポートしている場合があります。このとき、クライアントはサーバに対して、MTU 交換手順を開始することができます。MTU 交換手順によって、クライアントの MTU に、クライアントとサーバがサポートしている MTU の最大値が設定されます。

本 API の実行によって、mtu に設定した値で MTU 交換手順が開始されます。mtu に設定可能な値の範囲は、23~CFG_BLE_MAX_ATT_MTU_SZ です。本 API に対して生成されるイベントは、BLE_APP_GATT_MTU_EVENT、BLE_APP_GATT_ERR_EVT です。

本 API は、クライアントデバイスのみ実行可能です。

7.8.2. ble_disc_all_pri_svc

(プライマリサービス検出)

【書式】

ER ble_disc_all_pri_svc(UH con_hnd);

【パラメータ】

UH	con_hnd	コネクションハンドル
----	---------	------------

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー

【解説】

サーバから、すべてのプライマリサービスを検出します。この API に対して生成されるイベントは、BLE_SVC_EVENT、BLE_PROC_DONE_EVENT または BLE_ERROR_EVENT です。

7.8.3. ble_disc_all_pri_svc_by_uuid

(特定プライマリサービス検出)

【書式】

```
ER ble_disc_all_pri_svc_by_uuid(UH con_hnd, T_BLE_UUID *uuid);
```

【パラメータ】

UH	con_hnd	コネクションハンドル
T_BLE_UUID *	uuid	プライマリサービスの UUID

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー

【解説】

UUID を指定して、サーバから特定のプライマリサービスを検出します。この API に対して生成されるイベントは、BLE_SVC_EVENT、BLE_PROC_DONE_EVENT または BLE_ERROR_EVENT です。

7.8.4. ble_disc_inc_svc

(インクルードサービス検出)

【書式】

ER ble_disc_inc_svc(UH con_hnd);

【パラメータ】

UH	con_hnd	コネクションハンドル
----	---------	------------

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー

【解説】

サーバから、インクルードサービスを検出します。インクルードサービスは、セカンダリサービスである必要があります。この API に対して生成されるイベントは、BLE_INC_SVC_EVENT、BLE_PROC_DONE_EVENT または BLE_ERROR_EVENT です。

7.8.5. ble_disc_all_char

(キャラクターリスティック検出)

【書式】

ER ble_disc_all_char(UH con_hnd, UH top, UH end);

【パラメータ】

UH	con_hnd	コネクションハンドル
UH	top	サービスの開始ハンドル
UH	end	サービスの終了ハンドル

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー

【解説】

サービスの開始ハンドル及び終了ハンドルを指定して、サーバから、キャラクターリスティックを検出します。この API に対して生成されるイベントは、BLE_CHAR_EVENT、BLE_PROC_DONE_EVENT または BLE_ERROR_EVENT です。

7.8.6. ble_disc_all_char_by_uuid

(特定キャラクタースティック検出)

【書式】

```
ER ble_disc_all_char_by_uuid(UH con_hnd, UH top, UH end,
                             T_BLE_UUID *uuid);
```

【パラメータ】

UH	con_hnd	コネクションハンドル
UH	top	サービスの開始ハンドル
UH	end	サービスの終了ハンドル
T_BLE_UUID	uuid	キャラクタースティック UUID

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー

【解説】

UUID を指定して、サーバ上の特定のキャラクタースティックを検出します。この API に対して生成されるイベントは、BLE_CHAR_EVENT、BLE_PROC_DONE_EVENT または BLE_ERROR_EVENT です。

7.8.7. ble_disc_all_char_desc

(ディスクリプタ検出)

【書式】

ER ble_disc_all_char_desc(UH con_hnd, UH top, UH end);

【パラメータ】

UH	con_hnd	コネクションハンドル
UH	top	キャラクタリスティックの開始ハンドル
UH	end	キャラクタリスティックの終了ハンドル

【戻り値】

ER 正常終了(E_OK)またはエラーコード

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー

【解説】

キャラクタリスティックの開始ハンドル及び終了ハンドルを指定して、サーバ上のディスクリプタを検出します。この関数に対して生成されるイベントは、BLE_CHAR_DESC_EVENT、BLE_PROC_DONE_EVENT または BLE_ERROR_EVENT です。

7.8.8. ble_read_char

(キャラクタリスティック値読み取り)

【書式】

ER ble_read_char(UH con_hnd, UH attr_hnd);

【パラメータ】

UH	con_hnd	コネクションハンドル
UH	attr_hnd	キャラクタリスティックハンドル

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー

【解説】

ハンドルを指定して、キャラクタリスティック値を読み取ります。この API に対して生成されるイベントは、BLE_CHAR_VALUE_EVENT または BLE_ERROR_EVENT です。

7.8.9. ble_read_long_char

(キャラクターリスティック値読み取り(大きいサイズ))

【書式】

ER ble_read_long_char(UH con_hnd, UH attr_hnd, UH offset);

【パラメータ】

UH	con_hnd	コネクションハンドル
UH	attr_hnd	キャラクターリスティックハンドル
UH	offset	オフセット

【戻り値】

ER 正常終了(E_OK)またはエラーコード

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー

【解説】

ハンドルを指定して、大きいサイズのキャラクターリスティック値を読み取ります。
 ble_read_char で読み取れる最大サイズは、(MTU - 1)バイトであるため、これを超えるサイズの値を読み取る際は、こちらの API を実行してください。この API に対して生成されるイベントは、BLE_CHAR_LONG_VALUE_EVENT または BLE_ERROR_EVENT です。

7.8.1 0. ble_read_char_by_uuid

(キャラクターリスティック値読み取り(UUID 指定))

【書式】

```
ER ble_read_char_by_uuid(UH con_hnd, UH top, UH end,
                        T_BLE_UUID *uuid);
```

【パラメータ】

UH	con_hnd	コネクションハンドル
UH	top	キャラクターリスティックの開始ハンドル
UH	end	キャラクターリスティックの終了ハンドル
T_BLE_UUID	uuid	キャラクターリスティック UUID

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー

【解説】

UUID を指定して、サーバ上の読み取り許可として設定されているキャラクターリスティック値を読み取ります。この API に対して生成されるイベントは、BLE_CHAR_VALUE_EVENT または BLE_ERROR_EVENT です。

7.8.1 1. ble_write_char

(キャラクターリスティック値書き込み)

【書式】

ER ble_write_char(UH con_hnd, UH char_hnd, UB*dat, UH len);

【パラメータ】

UH	con_hnd	コネクションハンドル
UH	char_hnd	キャラクターリスティック値ハンドル
UB *	dat	書き込むデータ
UH	len	データ長

【戻り値】

ER 正常終了(E_OK)またはエラーコード

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー

【解説】

ハンドルを指定して、サーバ上の書き込み権限が許可されたキャラクターリスティック値に、データを書き込みます。この API に対して生成されるイベントは、BLE_PROC_DONE_EVENT または BLE_ERROR_EVENT です。

7.8.1 2. ble_write_cmd_char

(キャラクターリスティック値書き込み(応答なし))

【書式】

```
ER ble_write_cmd_char(UH con_hnd, UH char_hnd, UB*dat,
                     UH len, BOOL sign_wr);
```

【パラメータ】

UH	con_hnd	コネクションハンドル
UH	char_hnd	キャラクターリスティック値ハンドル
UB *	dat	書き込むデータ
UH	len	データ長
BOOL	sign_wr	TRUE または FALSE(※)

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー

※sign_wr は拡張用パラメータです。本関数の実行時には FALSE を指定してください。

【解説】

ハンドルを指定して、サーバ上の書き込み権限が許可されたキャラクターリスティック値に、応答なしでデータを書き込みます。

7.8.1 3. ble_write_long_char

(キャラクタリスティック値書き込み(大きいサイズ))

【書式】

```
ER ble_write_long_char(UH con_hnd, UH char_hnd, UB*dat, UH len, UH
offset);
```

【パラメータ】

UH	con_hnd	コネクションハンドル
UH	char_hnd	キャラクタリスティック値ハンドル
UB *	dat	書き込むデータ
UH	len	データ長
UH	offset	オフセット

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー

【解説】

ハンドルを指定して、サーバ上の、書き込み権限が許可されたキャラクタリスティック値に、大きいサイズのデータを書き込みます。ble_write_char で書き込める最大サイズは、MTU - 3 バイトであるため、これを超えるサイズのデータを書き込むときは、この API を実行してください。この API に対して生成されるイベントは、BLE_PROC_DONE_EVENT または BLE_ERROR_EVENT です。

7.8.1 4. ble_notify

(キャラクタースティック値 notify)

【書式】

ER ble_notify(UH con_hnd, UH char_hnd);

【パラメータ】

UH	con_hnd	コネクションハンドル
UH	char_hnd	キャラクタースティック値ハンドル

【戻り値】

ER 正常終了(E_OK)またはエラーコード

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_BOVR	バッファオーバーフロー

【解説】

サーバ上の指定したキャラクタースティックプロパティで、notify ビットがセットされている場合、クライアントからの要求なしで、サーバからクライアントにキャラクタースティック値を送信します。

7.8.1 5. ble_indicate

(キャラクターリスティック値 indicate)

【書式】

ER ble_indicate(UH con_hnd, UH char_hnd);

【パラメータ】

UH	con_hnd	コネクションハンドル
UH	char_hnd	キャラクターリスティック値ハンドル

【戻り値】

ER 正常終了(E_OK)またはエラーコード

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_BOVR	バッファオーバーフロー

【解説】

サーバ上の指定したキャラクターリスティックで、**indicate** ビットがセットされている場合、クライアントからの要求なしで、サーバからクライアントにキャラクターリスティック値を送信します。この API に対して生成されるイベントは、BLE_PROC_DONE_EVENT または BLE_ERROR_EVENT です。

7.9 プライバシーAPI

7.9.1. ble_gen_irk

(IRK 生成)

【書式】

```
ER ble_gen_irk(UB *p_ir, UB *p_irk);
```

【パラメータ】

UB *	p_ir	16 バイトの IR 値
UB *	p_irk	16 バイトの IRK 値

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_BOVR	バッファオーバーフロー

【解説】

設定された IR 値に対して IRK 値を生成します。本 API は Bluetooth core specification, "Vol.3, Part H, Appendix B.2.3 Generating Keys from IR"に記載されたアルゴリズムを使用して IRK 値を生成します。

7.9.2. ble_cfg_resolve

(Resolving リストの設定)

【書式】

```
ER ble_cfg_resolve(T_BLE_RES *pk_resolve_list, UB list_cnt, UB *ble_error);
```

【パラメータ】

T_BLE_RES *	pk_resolve_list	Resolving リスト
UB	list_cnt	Resolving リストのエントリ数
UB *	ble_error	BLE エラー

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_RLWAI	待ち状態強制解除
E_BOVR	バッファオーバーフロー

【解説】

BLE コントローラの Resolving リストを設定します。BLE コントローラは、Resolving リストの情報を使用して、RPA の生成や、受信パケットのプライベートアドレス解決を行います。Resolving リストに設定できるエントリの最大数は、BLE コントローラに依存することに注意してください。

T_BLE_RES パラメータ

型	フィールド	説明
UB	peer_addr_type	ピアデバイスアドレスタイプ
UB	peer_addr[6]	ピアデバイスアドレス
UB	peer_irk[16]	ピアデバイス IRK
UB	local_irk[16]	ローカルデバイス IRK

7.9.3. ble_chk_addr_resolvable

(受信 RPA が既知デバイスのものか確認)

【書式】

 ER ble_chk_addr_resolvable(UB *p_peer_addr, UB * p_peer_irk);

【パラメータ】

UB *	p_peer_addr	ピアデバイスアドレス
UB *	p_peer_irk	ピアデバイス IRK

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー(RPA 解決不可)
E_RLWAI	待ち状態強制解除
E_BOVR	バッファオーバーフロー

【解説】

受信した RPA が、既知のピアデバイスから受信したものであるかどうかチェックします。p_peer_addr が、p_peer_irk で解決可能な場合、E_OK を返します。解決不可の場合、E_OBJ を返します。

7.10 ペアリング API

7.10.1. ble_pair_master_req

(セントラルデバイスからのペアリング開始要求)

【書式】

```
ER ble_pair_master_req(UH con_hnd, T_BLE_PAIR_PARAM *pk_param);
```

【パラメータ】

UH	con_hnd	コネクションハンドル
T_BLE_PAIR_PARAM *	pk_param	ペアリングパラメータ

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_OBJ	オブジェクト状態エラー

【解説】

セントラルデバイスが、ペアリング要求プロセスを開始します。ペアリングプロセスが完了すると、BLE_APP_SMP_PAIR_DONE_EVT が発生します。このとき、イベントパラメータの T_BLE_BOND が、相手デバイスのボンディング情報を提供します。

ペアリングプロセスには、“LE Secure connection pairing”を使用します。“LE Legacy pairing”はサポートしていないことに注意してください。また、ペリフェラルデバイスは、本 API を実行しないでください。

T_BLE_PAIR_PARAM パラメータ

型	フィールド	説明
UB	io_cap	IO 能力 下記の IO 能力値を参照し、IO 能力に応じた値をセットしてください。
UB	bond	ボンディング情報 0x00 - デバイスはボンディング情報を保持しません。 0x01 - デバイスはボンディング情報を保持します。
UB	mitm	MITM 情報 0x00 - MITM 保護が要求されます。 0x01 - MITM 保護は要求されません。
UB	key_press	キープレス情報 セントラルとペリフェラルの両方でこの値がセットされている場合、ble_pair_user_input()を使用してキープレス通知を送信できます
UB	min_enc_keysz	サポートする暗号鍵の最小サイズ 設定できる値の範囲は 7 から 16 です。 max_enc_keysz より大きい値を設定しないでください。
UB	max_enc_keysz	サポートする暗号鍵の最大サイズ 設定できる値の範囲は 7 から 16 です。
UB	itr_key_dis	ローカルデバイス鍵配布 0x00 - IRK を配布しません。 0x02 - IRK を配布します。p_irk に IRK 値をセットする必要があります。
UB	res_key_dis	リモートデバイス鍵配布 0x00 - リモートデバイスに IRK を配布しないことを要求します。 0x01 - リモートデバイスに IRK を配布することを要求します。
UB *	p_irk	ローカルデバイスの IRK 値 IRK が使用できない場合は、NULL をセットしてください。
UB	oob_flag	OOB フラグ 0x00 - OOB 認証データは利用できません。 0x01 - OOB 認証データが利用できます。
T_BLE_OOB *	pk_oob	ローカルデバイスの OOB データ OOB 認証データが利用できない場合は、NULL をセットしてください

IO 能力値

パラメータ	説明
BLE_PAIR_IO_CAPS_DISPLAY_ONLY	6桁の10進数値を表示または通信する機能を持つことを示します。
BLE_PAIR_IO_CAPS_DISPLAY_YESNO	6桁の10進数値を表示または通信する機能を持つことを示します。 また、「はい」と「いいえ」に対応する2つ以上のボタン、もしくはそれに相当する機能を持つことを示します。
BLE_PAIR_IO_CAPS_KEYBOARD_ONLY	0 から 9 までの数字を入力できる機能を持つことを示します。 また、「はい」と「いいえ」に対応する2つ以上のボタン、もしくはそれに相当する機能を持つことを示します。
BLE_PAIR_IO_CAPS_KEYBOARD_DISPLAY	6桁の10進数値を表示また通信する機能を持つことを示します。 また、0 から 9 までの数字を入力する機能を持つことを示します。また、「はい」と「いいえ」に対応する2つ以上のボタン、もしくはそれに相当する機能を持つことを示します。
BLE_PAIR_IO_CAPS_NONE	数値の表示や入力の機能を持たないことを示します。

T_BLE_OOB パラメータ

型	フィールド	説明
UB	rnd[16]	乱数
UB	cfm[16]	rnd の確認値 Bluetooth Core Specification で規定されているセキュリティ関数 f4 で生成される値です

7.1 0.2. ble_pair_slave_res

(ペリフェラルデバイスのペアリング応答)

【書式】

```
ER ble_pair_slave_res(UH con_hnd, T_BLE_PAIR_PARAM *pk_param,
                      UB status);
```

【パラメータ】

UH	con_hnd	コネクションハンドル
T_BLE_PAIR_PARAM *	pk_param	ペアリングパラメータ
UB	status	ペアリングステータス

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_OBJ	オブジェクト状態エラー

【解説】

セントラルデバイスに、ペアリング応答を送信します。本 API は、BLE_APP_SMP_PAIR_REQ_EVT が発生したときに、実行する必要があります。ペアリングプロセスが完了すると、BLE_APP_SMP_PAIR_DONE_EVT します。このとき、イベントパラメータの T_BLE_BOND が、相手デバイスのボンディング情報を提供します。

7.1 0.3. ble_pair_slave_req

(ペリフェラルデバイスからのペアリング開始要求)

【書式】

ER ble_pair_slave_req(UH con_hnd, T_BLE_PAIR_PARAM *pk_param);

【パラメータ】

UH	con_hnd	コネクションハンドル
T_BLE_PAIR_PARAM *	pk_param	ペアリングパラメータ

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_OBJ	オブジェクト状態エラー

【解説】

ペリフェラルデバイスが、セントラルデバイスに対してペアリングプロセスの開始を要求することができます。本 API 実行時は、pk_param のパラメータのうち、bond、mitm 及び key_press のみ設定してください。

7.1 0.4. ble_pair_user_input

(ペアリング入力方法イベントに対する応答)

【書式】

```
ER ble_pair_user_input(UH con_hnd, UB type,
                      T_BLE_PAIR_USER_INPUT *pk_param);
```

【パラメータ】

UH	con_hnd	コネクションハンドル
UB	type	入力タイプ
T_BLE_PAIR_USER_INPUT *	pk_param	入力タイプパラメータ

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_OBJ	オブジェクト状態エラー

【解説】

本 API は、ペアリングプロセス中に、ユーザーデータを入力するために使用します。BLE_APP_SMP_PAIR_NUM_DISP_EVT、BLE_APP_SMP_PAIR_PK_INPUT_EVT もしくは BLE_APP_SMP_PAIR_OOB_INPUT_EVT イベントが発生したとき、実行する必要があります。以下に従って、発生イベントに対応した値を **type** にセットしてください。

イベント	type にセットする値
BLE_APP_SMP_PAIR_KEY_PRESS_EVT	BLE_PAIR_USER_INPUT_KEY_PRESS
BLE_APP_SMP_PAIR_NUM_DISP_EVT	BLE_PAIR_USER_INPUT_YESNO
BLE_APP_SMP_PAIR_PK_INPUT_EVT	BLE_PAIR_USER_INPUT_PASS_KEY
BLE_APP_SMP_PAIR_PK_DISPLAY_EVT	BLE_PAIR_USER_INPUT_PASS_KEY
BLE_APP_SMP_PAIR_OOB_INPUT_EVT	BLE_PAIR_USE_INPUT_OOB

T_BLE_PAIR_USER_INPUT パラメータ

型	フィールド	説明
UB	yes_no	type に BLE_PAIR_USER_INPUT_YESNO がセットされている場合、有効になります。入力された値を受け入れる場合 1、拒否する場合 0 をセットしてください。
UB	key_press	type に BLE_PAIR_USER_INPUT_KEY_PRESS がセットされている場合、有効になります。key_press パラメータに記載した値から、適切なものを選択してセットしてください。
UW	pass_key	BLE_PAIR_USER_INPUT_PASS_KEY がセットされている場合、有効になります。Passkey の値をセットしてください。
T_BLE_OOB	oob	BLE_PAIR_USE_INPUT_OOB がセットされている場合、有効になります。OOB データをセットしてください。

key_press パラメータ

パラメータ	説明
BLE_PAIR_KEY_P_START	パスキーエントリの開始を通知します
BLE_PAIR_KEY_P_DIGIT_ENTERD	パスキーが 1 桁入力されたことを通知します
BLE_PAIR_KEY_P_DIGIT_ERASED	パスキーが 1 桁削除されたことを通知します
BLE_PAIR_KEY_P_CLEARED	パスキーがクリアされたことを通知します
BLE_PAIR_KEY_P_COMPLETED	パスキーエントリが完了したことを通知します

7.1 0.5. ble_gen_oob_dat

(OOB データ生成)

【書式】

```
ER ble_gen_oob_dat(UB *p_rnd, T_BLE_OOB *pk_oob);
```

【パラメータ】

UB *	p_rnd	16 バイト乱数
T_BLE_OOB *	pk_oob	生成される OOB データ

【戻り値】

ER	正常終了(E_OK)またはエラーコード
----	---------------------

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号
E_NOMEM	メモリ不足
E_OBJ	オブジェクト状態エラー
E_NOEXS	オブジェクト未生成
E_RLWAI	待ち状態強制解除
E_TMOUT	ポーリング失敗またはタイムアウト
EV_BLE_ERR	BLE スタック固有エラー

【解説】

p_rnd にセットされた乱数にもとづいて OOB データを生成します。生成した OOB データは、ペアリングリクエストやレスポンスに使用できます。p_rnd に値がセットされていない場合は、API 内部で乱数を生成した後、OOB データを生成します。

7.1 0.6. ble_get_bond_by_con

(ボンディング情報の検索)

【書式】

```
ER ble_get_bond_by_con(UH *con_hnd, T_BLE_BOND *pk_bond_list, UB cnt,
                      T_BLE_BOND **ppk_bond_match);
```

【パラメータ】

UH *	con_hnd	コネクションハンドル
T_BLE_BOND *	pk_bond_list	ボンディングリスト
UB	cnt	ボンディングリストの エントリ数
T_BLE_BOND **	ppk_bond_match	一致ボンディング情報

【戻り値】

ER 正常終了(E_OK)またはエラーコード

【エラーコード】

E_PAR	パラメータエラー
E_OBJ	オブジェクト状態エラー
E_NOEXS	オブジェクト未生成

【解説】

コネクションしているピアデバイスのアドレス情報を使用して、ボンディングリストからピアデバイスのボンディング情報を検索することができます。一致するボンディング情報があった場合、戻り値 E_OK を返します。また、一致したボンディング情報が、**ppk_bond_match に返されます。

なお、本 BLE スタックは、ボンディングリストを保存する機能をサポートしていないため、ボンディングリストの保存はアプリケーションで実装する必要があります。

T_BLE_BOND パラメータ

型	フィールド	説明
T_BLE_ADDR	own	ローカルデバイスアドレス
T_BLE_ADDR	peer	ピアデバイスアドレス
UB	own_irk[16]	ローカル IRK 値
UB	peer_irk[16]	ピア IRK 値
UB	ltk[16]	ペアリングプロセスで生成された LTK 値
UB	keysz	LTK サイズ。7～16
UB	auth	0x00 - ペアリングが認証されていないことを示す。 0x01 - ペアリングが認証されていることを示す。
UB	bond	0x00 - ボンディング情報が保存されていることを示す。 0x01 - ボンディング情報が保存されていないことを示す。

7.1.1 コールバックイベント

アプリケーションコールバック関数は、ble_ini()で登録されます。コールバック関数は、アプリケーションにイベント通知するため、BLE スタックから呼び出されます。

イベントの一覧を以下に示します。

イベントグループ	説明
BLE_APP_EVG_DEV	BLE コマンド関連のイベントです
BLE_APP_EVG_GATT	GATT プロシージャ関連のイベントです
BLE_APP_EVG_L2CAP	L2CAP プロシージャ関連のイベントです
BLE_APP_EVG_SM	ペアリングに関連するイベントです

BLE_APP_EVG_DEV イベント

イベント名	パラメータ
BLE_APP_ADV_EVT	アドバタイズイベント 本イベントのパラメータは、後述の T_BLE_LE_ADV を参照してください。
BLE_APP_ADV_DIR_EVT	有向アドバタイズイベント 本イベントのパラメータは、後述の T_BLE_LE_DIRECT_ADV を参照してください。
BLE_APP_ADV_TER_EVT	アドバタイズタイムアウトイベント 本イベントのパラメータはありません。
BLE_APP_CON_EVT	コネクション完了イベント 本イベントのパラメータは、後述の T_BLE_LE_CON_COMP を参照してください。
BLE_APP_CON_CLS_EVT	コネクション終了イベント 本イベントのパラメータは、後述の T_BLE_LE_DIS_COMP を参照してください。
BLE_APP_CON_UPD_EVT	コネクション更新イベント 本イベントのパラメータは、後述の T_BLE_LE_CON_COMP を参照してください。
BLE_APP_CON_ENC_REQ_EVT	コネクション暗号化要求イベント 本イベントのパラメータは、後述の T_BLE_CON_ENC_REQ_EVT を参照してください。
BLE_APP_CON_ENC_CHG_EVT	暗号化変更イベント 本イベントのパラメータは、後述の T_BLE_CON_ENC_EVT を参照してください。
BLE_APP_CON_ENC_KEY_REF_EVT	暗号鍵リフレッシュイベント 本イベントのパラメータは、後述の T_BLE_CON_ENC_EVT を参照してください。

T_BLE_LE_ADV パラメータ

型	フィールド	説明
UB	event_type	イベントタイプを示します。 0x00 - コネクション可能、無向アドバタイズ 0x01 - コネクション可能、有向アドバタイズ 0x02 - スキャン可能、コネクション不可アドバタイズ 0x03 - コネクション不可、無向アドバタイズ
UB	addr_type	アドレスタイプを示します。 0x00 - パブリックデバイスアドレス 0x01 - ランダムデバイスアドレス 0x02 - パブリックアイデンティティアドレス 0x03 - ランダムアイデンティティアドレス
UB *	addr	6 バイトのアドレス値
UB	data_length	アドバタイズデータサイズ
UB *	data	アドバタイズデータ
B	rssi	RSSI 値 -127 から+20 の値をとります。

T_BLE_LE_DIRECT_ADV パラメータ

型	フィールド	説明
UB	event_type	イベントタイプを示します。 0x00 - コネクション可能、無向アドバタイズ 0x01 - コネクション可能、有向アドバタイズ 0x02 - スキャン可能、コネクション不可アドバタイズ 0x03 - コネクション不可、無向アドバタイズ
UB	addr_type	アドレスタイプを示します。 0x00 - パブリックデバイスアドレス 0x01 - ランダムデバイスアドレス 0x02 - パブリックアイデンティティアドレス 0x03 - ランダムアイデンティティアドレス
UB *	addr	6 バイトのアドレス値
UB	direct_addr_type	0x01 固定 ランダムデバイスアドレスであることを示します。
UB *	direct_addr	ランダムデバイスアドレス値
B	rssi	RSSI 値 -127 から+20 の値をとります。

T_BLE_LE_CON_COMP パラメータ

型	フィールド	説明
UB	status	コネクション状態 0：コネクション成功、それ以外はエラーを示す。
UH	conn_handle	コネクションハンドル値
UB	role	0x00 - マスターであることを示します。 0x01 - スレーブであることを示します。
UB	peer_addr_type	0x00 - ピアデバイスがパブリックデバイスアドレスを使用していることを示します。 0x01 - ピアデバイスがランダムデバイスアドレスを使用していることを示します。
UB	peer_addr[6]	ピアデバイスのアドレス値
UH	conn_interval	コネクションインターバル値
UH	conn_latency	スレーブレイテンシ値
UH	supervision_timeout	スーパービジョンタイムアウト値
UB	mca	Master clock accuracy マスターデバイスのクロック精度

T_BLE_LE_DIS_COMP パラメータ

型	フィールド	説明
UB	status	コネクション状態 0：切断成功、それ以外はエラーを示す。
UH	conn_handle	コネクションハンドル値
UB	reason	コネクション切断理由 値については、Bluetooth Core Specification Vol 2, Part D, Error Code Description を参照してください。

T_BLE_CON_ENC_REQ_EVT パラメータ

型	フィールド	説明
UH	con_hnd	コネクションハンドル値
UB	status	本パラメータは使用しません
UB *	rnd	本パラメータは使用しません
UB *	ediv	本パラメータは使用しません

T_BLE_CON_ENC_EVT パラメータ

型	フィールド	説明
UH	con_hnd	コネクションハンドル値
UB	status	0：暗号化状態変更。それ以外はエラーを示す。
UB *	enc_enabled	0x00 - 接続が暗号化されていません 0x01 - 接続が暗号化されています ※BLE_APP_CON_ENC_KEY_REF_EVT では使用しません

BLE_APP_EVT_GATT イベント

イベント	T_BLE_CLI_APP_EVT パラメータ
BLE_APP_GATT_MTU_EVT	UH mtu.val - MTU の値
BLE_APP_GATT_SVC_EVT	UH svc.top - サービスの開始ハンドル UH svc.end - サービスの終了ハンドル T_BLE_UUID svc.uuid - サービスの UUID
BLE_APP_GATT_INC_SVC_EVT	UH inc.hnd - インクルードサービスのハンドル UH inc.top - インクルードサービスの開始ハンドル UH inc.end - インクルードサービスの終了ハンドル T_BLE_UUID inc.uuid - インクルードサービスの UUID
BLE_APP_GATT_CHAR_EVT	UH chr.hnd - キャラクタリスティックのハンドル UH chr.property - キャラクタリスティックプロパティ UH chr.val_hnd - キャラクタリスティック値ハンドル T_BLE_UUID chr.uuid - キャラクタリスティック値の UUID
BLE_APP_GATT_CHAR_VALUE_EVT	UH chr_val.char_hnd - キャラクタリスティックのハンドル UH chr_val.len - キャラクタリスティック値のサイズ UH *chr_val.val - キャラクタリスティック値
BLE_APP_GATT_CHAR_DESC_EVT	UH desc.hnd - ディスクリプタのハンドル T_BLE_UUID desc.uuid - ディスクリプタの UUID
BLE_APP_GATT_ERR_EVT	UB err.opc - エラーオペコード UH err.hnd - エラーハンドル UH err.reason - エラー理由
BLE_APP_GATT_CHAR_LONG_VALUE_EVT	UH long_chr_val.char_hnd - キャラクタリスティックのハンドル UH long_chr_val.len - キャラクタリスティック値のサイズ UH long_chr_val.offset - オフセット UH *long_chr_val.value - キャラクタリスティック値
BLE_APP_GATT_DONE_EVT	UH con_hnd - コネクションハンドル
BLE_APP_GATT_TMO_EVT	UH con_hnd - コネクションハンドル
BLE_APP_GATT_NOTIFY_EVT	UH nfy.hnd - キャラクタリスティックのハンドル UB *nfy.value - キャラクタリスティックの値 UH nfy.len - キャラクタリスティック値のサイズ
BLE_APP_GATT_INDICATE_EVT	UH ind.hnd - キャラクタリスティックのハンドル UB *ind.value - キャラクタリスティックの値 UH ind.len - キャラクタリスティック値のサイズ

BLE_APP_EVTG_L2CAP イベント

イベント	T_BLE_L2CAP_APP_EVT パラメータ
BLE_APP_L2CAP_REQ_EVT (L2CAP コネクションパラメータの更新要求パケットを受信したときに通知されます。)	UH con_hnd - コネクションハンドル UH l2cap_req.int_min - 最小コネクションイベント間隔 UH l2cap_req.int_max - 最大コネクションイベント間隔 UH l2cap_req.slave_latency - スレーブレイテンシ UH l2cap_req.timeout_multiplier - スーパービジョンタイムアウトに使う計算値
BLE_APP_L2CAP_RES_EVT (L2CAP コネクションパラメータの更新応答パケットが受信されたときに通知されます。)	UH con_hnd - コネクションハンドル UH l2cap_res.result 0x0000 - コネクションパラメータ許可 0x0001 - コネクションパラメータ拒否
BLE_APP_L2CAP_REJ_EVT (L2CAP コマンドの拒否パケットを受信したときに通知されます。)	UH con_hnd - コネクションハンドル UH l2cap_rej.reason 0x0000 - コマンド不明 0x0001 - MTU 超過 0x0002 - CID 無効 UB *dat - オプションデータ UH dat_len - オプションデータ長
BLE_APP_L2CAP_REQ_TMO (ble_l2cap_cmd_con_upd_req() の呼び出しに対して、30 秒間応答が無かった場合に通知されます。)	UH con_hnd - コネクションハンドル

BLE_APP_EVG_SM イベント

イベント	T_BLE_PAIR_APP_EVT パラメータ
BLE_APP_SMP_PAIR_REQ_EVT (ペアリング要求イベント)	UH con_hnd - コネクションハンドル UB pair_req.io_cap - IO 能力値 UB pair_req.oob_dat_flg - OOB データフラグ値 UB pair_req.bond - ボンディング情報 UB pair_req.mitm - MITM 値 UB pair_req.sc - セキュアコネクション値 1 の場合 LE セキュアコネクションの要求、0 の場合 LE レガシーペアリングの要求。 UB pair_req.keypress - キープレス値 両デバイスで 1 にセットした場合、キープレス通知による Pass Key エントリが実行される。 UB max_encr_keysz - 暗号鍵最大サイズ UB pair_req.itr_key_dis - イニシエータ鍵配布値 UB pair_req.res_key_dis - レスポンダ鍵配布値
BLE_APP_SMP_PAIR_NUM_DIS_P_EVT (数値表示イベント)	UH con_hnd - コネクションハンドル UW pass_key.value - 入力された数値
BLE_APP_SMP_PAIR_PK_INPUT_EVT (パスキー入力イベント)	UH con_hnd - コネクションハンドル
BLE_APP_SMP_PAIR_PK_DISPLAY_EVT (Pass key display event)	UH con_hnd - コネクションハンドル
BLE_APP_SMP_PAIR_OOB_INPUT_EVT (OOB データ入力)	UH con_hnd - コネクションハンドル
BLE_APP_SMP_PAIR_LTK_GEN_EVT (LTK 値生成イベント)	UH con_hnd - コネクションハンドル UB ltk.p_value - 生成された LTK 値 UB ltk.keysz - LTK サイズ。値は 7 から 16
BLE_APP_SMP_PAIR_DONE_EVENT (ペアリング完了イベント)	UH con_hnd - コネクションハンドル ER done.ercd - ペアリングプロセスのエラーステータス。E_OK は、ペアリング成功を示す。それ以外の値は、ペアリングエラーを示す。 UB done.status - ペアリングエラー時、エラーの詳細を提供する。後述の UB done.pk_bond - Bond information. Refer to “T_BLE_BOND parameter” section.
BLE_APP_SMP_PAIR_KEY_PRESSED_EVT (キープレス通知イベント)	UH con_hnd - コネクションハンドル UB type - キープレス通知タイプ
BLE_APP_SMP_PAIR_SEC_REQ_EVT (ペリフェラルからのペアリング要求イベント)	UB bond - ボンディング情報 UB mitm - MITM 値 UB sc - セキュアコネクション値 UB keypress - キープレス値

7.1.2 エラーコード

7.1.2.1. API 戻り値

API の戻り値として返される、BLE スタック固有のエラーコードです。

エラーコード	値	説明
EV_BLE_ERR	-511841	BLE プロトコル特有のエラーです。実際のエラーコードは、引数”ble_error”の値を参照してください。エラーコードの説明は、Bluetooth Core Specification Vol 2, Part D ERROR CODES を参照してください。

7.1.2.2. BLE エラーコード

Bluetooth Core Specification, Vol 2, Part D, Error Codes にもとづいて定義した、本 BLE スタックが使用するエラーコードマクロの一覧です。

マクロ名	値
BLE_ERR_SUCCESS	0x00
BLE_ERR_UNWN_HCI_CMD	0x01
BLE_ERR_UNWN_CONN_IDENT	0x02
BLE_ERR_HW_FAILURE	0x03
BLE_ERR_PAGE_TIMEOUT	0x04
BLE_ERR_AUTHENTICATION_FAILURE	0x05
BLE_ERR_PIN_OR_KEY_MISSING	0x06
BLE_ERR_MEMORY_CAPACITY_EXCEEDED	0x07
BLE_ERR_CONNECTION_TIMEOUT	0x08
BLE_ERR_CONNECTION_LIMIT_EXCEEDED	0x09
BLE_ERR_SYNC_CONN_LIMIT_TO_A_DEVICE_EXCEEDED	0x0A
BLE_ERR_ACL_CONNECTION_ALREADY_EXISTS	0x0B
BLE_ERR_COMMAND_DISALLOWED	0x0C
BLE_ERR_CONNECTION_REJECTED_DUE_TO_LIMITED_RESOURCES	0x0D
BLE_ERR_CONNECTION_REJECTED_DUE_TO_SECURITY_REASONS	0x0E
BLE_ERR_CONNECTION_REJECTED_DUE_TO_UNACCEPTABLE_BD_ADDR	0x0F
BLE_ERR_CONNECTION_ACCEPT_TIMEOUT_EXCEEDED	0x10
BLE_ERR_UNSUPPORTED_FEATURE_OR_PARAMETER_VALUE	0x11
BLE_ERR_INVALID_HCI_COMMAND_PARAMETERS	0x12
BLE_ERR_REMOTE_USER_TERMINATED_CONNECTION	0x13
BLE_ERR_REMOTE_DEVICE_TERMINATED_CONNECTION_DUE_TO_LOW_RESOURCES	0x14
BLE_ERR_REMOTE_DEVICE_TERMINATED_CONNECTION_DUE_TO_POWER_OFF	0x15
BLE_ERR_CONNECTION_TERMINATED_BY_LOCAL_HOST	0x16
BLE_ERR_REPEATED_ATTEMPTS	0x17
BLE_ERR_PAIRING_NOT_ALLOWED	0x18
BLE_ERR_UNKNOWN_LMP_PDU	0x19
BLE_ERR_UNSUPPORTED_REMOTE_FEATURE_UNSUPPORTED_LMP_FEATURE	0x1A
BLE_ERR_SCO_OFFSET_REJECTED	0x1B
BLE_ERR_SCO_INTERVAL_REJECTED	0x1C

マクロ名	値
BLE_ERR_SCO_AIR_MODE_REJECTED	0x1D
BLE_ERR_INVALID_LMP_PARAMETERS_INVALID_LL_PARAMETERS	0x1E
BLE_ERR_UNSPECIFIED_ERROR	0x1F
BLE_ERR_UNSUPPORTED_LMP_PARAMETER_VALUE_UNSUPPORTED_LL_PARAMETER_VALUE	0x20
BLE_ERR_ROLE_CHANGE_NOT_ALLOWED	0x21
BLE_ERR_LMP_RESPONSE_TIMEOUT_LL_RESPONSE_TIMEOUT	0x22
BLE_ERR_LMP_ERROR_TRANSACTION_COLLISION	0x23
BLE_ERR_LMP_PDU_NOT_ALLOWED	0x24
BLE_ERR_ENCRYPTION_MODE_NOT_ACCEPTABLE	0x25
BLE_ERR_LINK_KEY_CANNOT_BE_CHANGED	0x26
BLE_ERR_REQUESTED_QOS_NOT_SUPPORTED	0x27
BLE_ERR_INSTANT_PASSED	0x28
BLE_ERR_PAIRING_WITH_UNIT_KEY_NOT_SUPPORTED	0x29
BLE_ERR_DIFFERENT_TRANSACTION_COLLISION	0x2A
BLE_ERR_QOS_UNACCEPTABLE_PARAMETER	0x2C
BLE_ERR_QOS_REJECTED	0x2D
BLE_ERR_CHANNEL_CLASSIFICATION_NOT_SUPPORTED	0x2E
BLE_ERR_INSUFFICIENT_SECURITY	0x2F
BLE_ERR_PARAMETER_OUT_OF_MANDATORY_RANGE	0x30
BLE_ERR_ROLE_SWITCH_PENDING	0x32
BLE_ERR_RESERVED_SLOT_VIOLATION	0x34
BLE_ERR_ROLE_SWITCH_FAILED	0x35
BLE_ERR_EXTENDED_INQUIRY_RESPONSE_TOO_LARGE	0x36
BLE_ERR_SECURE_SIMPLE_PAIRING_NOT_SUPPORTED_BY_HOST	0x37
BLE_ERR_HOST_BUSY_PAIRING	0x38
BLE_ERR_CONNECTION_REJECTED_DUE_TO_NO_SUITABLE_CHANNEL_FOUND	0x39
BLE_ERR_CONTROLLER_BUSY	0x3A
BLE_ERR_UNACCEPTABLE_CONNECTION_PARAMETERS	0x3B
BLE_ERR_DIRECTED_ADVERTISING_TIMEOUT	0x3C
BLE_ERR_CONNECTION_TERMINATED_DUE_TO_MIC_FAILURE	0x3D
BLE_ERR_CONNECTION_FAILED_TO_BE_ESTABLISHED	0x3E
BLE_ERR_MAC_CONNECTION_FAILED	0x3F
BLE_ERR_COARSE_CLOCK_ADJUSTMENT_REJECTED_BUT_WILL_TRY_TO_ADJUST_USING_CLOCK_DRAGGING	0x40

7.1 2.3. BLE ATT エラーコード

Bluetooth Core Specification 4.2, Vol 3, Part F, Table 3.3 Error codes にもとづいて定義した、本 BLE スタックが使用する ATT エラーコードマクロの一覧です。

マクロ名	値
BLE_ATT_ERR_INVALID_HANDLE	0x01
BLE_ATT_ERR_READ_NOT_PERMITTED	0x02
BLE_ATT_ERR_WRITE_NOT_PERMITTED	0x03
BLE_ATT_ERR_INVALID_PDU	0x04
BLE_ATT_ERR_INSUFFICIENT_AUTHENTICATION	0x05
BLE_ATT_ERR_REQUEST_NOT_SUPPORTED	0x06
BLE_ATT_ERR_INVALID_OFFSET	0x07
BLE_ATT_ERR_INSUFFICIENT_AUTHORIZATION	0x08
BLE_ATT_ERR_PREPARE_QUEUE_FULL	0x09
BLE_ATT_ERR_ATTRIBUTE_NOT_FOUND	0x0A
BLE_ATT_ERR_ATTRIBUTE_NOT_LONG	0x0B
BLE_ATT_ERR_INSUFFICIENT_ENCRYPTION_KEY_SIZE	0x0C
BLE_ATT_ERR_INVALID_ATTRIBUTE_VALUE_LENGTH	0x0D
BLE_ATT_ERR_UNLIKELY_ERROR	0x0E
BLE_ATT_ERR_INSUFFICIENT_ENCRYPTION	0x0F
BLE_ATT_ERR_UNSUPPORTED_GROUP_TYPE	0x10
BLE_ATT_ERR_INSUFFICIENT_RESOURCES	0x11

7.1 2.4. BLE ペアリングエラーコード

Bluetooth Specification 4.2, Vol 3, Part H, Table 3.7 Error codes にもとづいて定義した、本 BLE スタックが使用する BLE ペアリングエラーコードのマクロ一覧です。

マクロ名	値
BLE_PAIR_ERR_PASSKEY_ENTRY_FAILED	0x01
BLE_PAIR_ERR_OOB_NOT_AVAILABLE	0x02
BLE_PAIR_ERR_AUTH_REQ_NOT_MET	0x03
BLE_PAIR_ERR_CONFIRM_VALUE_FAIL	0x04
BLE_PAIR_ERR_PAIRING_NOT_SUPP	0x05
BLE_PAIR_ERR_ENC_KEY_SIZE	0x06
BLE_PAIR_ERR_CMD_UNSUPPORTED	0x07
BLE_PAIR_ERR_UNSPECIFIED	0x08
BLE_PAIR_ERR_REPEATED_ATTEMPTS	0x09
BLE_PAIR_ERR_INVALID_PARAMS	0x0A
BLE_PAIR_ERR_DHKEY_FAILURE	0x0B
BLE_PAIR_ERR_NUM_COMP_FAILURE	0x0C
BLE_PAIR_ERR_BR_EDR_IN_PROG	0x0D
BLE_PAIR_ERR_TRANS_KEY_DISALLOWED	0x0E

7.13 グローバル UUID 変数

本 BLE スタックでは、以下をグローバル変数として定義しており、アプリケーションが GATT アトリビュートを定義する際に使用することができます。

```
T_BLE_UUID ble_primary_srv_uuid = {UUID_16BIT, {0x2800U}};  
T_BLE_UUID ble_secondary_srv_uuid = {UUID_16BIT, {0x2801U}};  
T_BLE_UUID ble_inc_srv_uuid = {UUID_16BIT, {0x2802U}};  
T_BLE_UUID ble_char_uuid = {UUID_16BIT, {0x2803U}};  
T_BLE_UUID ble_char_ext_desc_uuid = {UUID_16BIT, {0x2900U}};  
T_BLE_UUID ble_char_usr_desc_uuid = {UUID_16BIT, {0x2901U}};  
T_BLE_UUID ble_char_cli_desc_uuid = {UUID_16BIT, {0x2902U}};  
T_BLE_UUID ble_char_ser_desc_uuid = {UUID_16BIT, {0x2903U}};  
T_BLE_UUID ble_char_app_desc_uuid = {UUID_16BIT, {0x2904U}};  
T_BLE_UUID ble_char_agg_desc_uuid = {UUID_16BIT, {0x2905U}};
```

8 OS 資源

BLE スタックが使用する OS 資源を説明します。なお、各パラメータはデフォルト値で、アプリケーション実装によっては、変更して使用するケースがあります。詳細については、「9 コンフィグレーション」を参照してください。

8.1 タスク

ID	優先度	スタック サイズ	用途
ID_BLE_SDIO_TSK	4	512 byte	BLE スタック SDIO 制御タスク
ID_BLE_CTL_TSK	4	1024 byte	BLE スタックコントロールタスク
ID_BLE_SM_TSK	4	1024 byte	BLE SM モジュールタスク

8.2 セマフォ

ID	用途
なし	

8.3 イベントフラグ

ID	用途
ID_BLE_SDIO_FLG	SDIO 制御用
ID_BLE_CTL_FLG	BLE スタックイベントフラグ
ID_BLE_SM_FLG	BLE スタック SM モジュールイベントフラグ

8.4 メールボックス

ID	用途
ID_BLE_SDIO_MBX	SDIO(HCI)データ送信

8.5 固定長メモリプール

ID	ブロック数	ブロック サイズ	用途
ID_BLE_CMD_MPF	16	320 byte	HCI コマンドデータ (T_BLE_HCI_CMD_PKT 構造体のデータ格納用)
ID_BLE_EVT_MPF	16	600 byte	BLE パケットデータ (T_BLE_PKT 構造体のデータ格納用)、HCI イベント データ (T_BLE_HCI_EVT 構造体のデータ格納用)
ID_BLE_SDIO_MPF	8	360 byte	SDIO データ送信用 (T_BLE_HCI_PKT 構造体 のデータ格納用)

8.6 周期ハンドラ

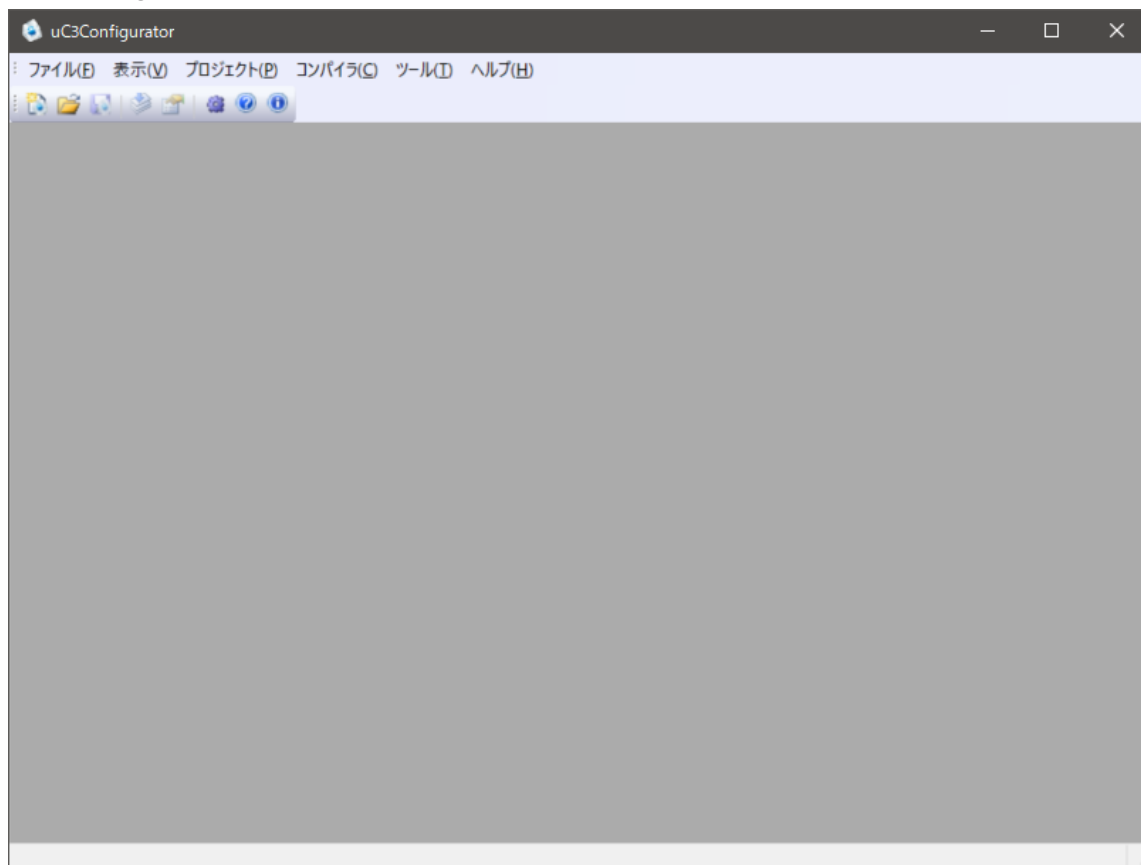
ID	起動周期	用途
ID_BLE_TIM_CYC	100ms	BLE タイマイベント管理

9 コンフィグレーション

BLE スタックを使用する場合、システム設計で決定される各オブジェクトのパラメータとなるコンフィグレーション情報をコンフィグレータに入力し、必要なソースファイルを生成することができます。

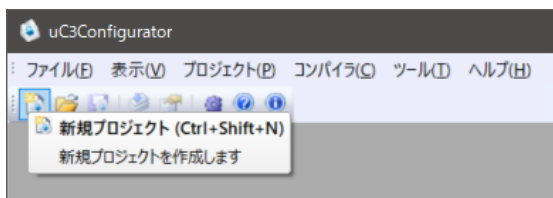
9.1 コンフィグレータの起動

「Configurator.exe」を起動してください。



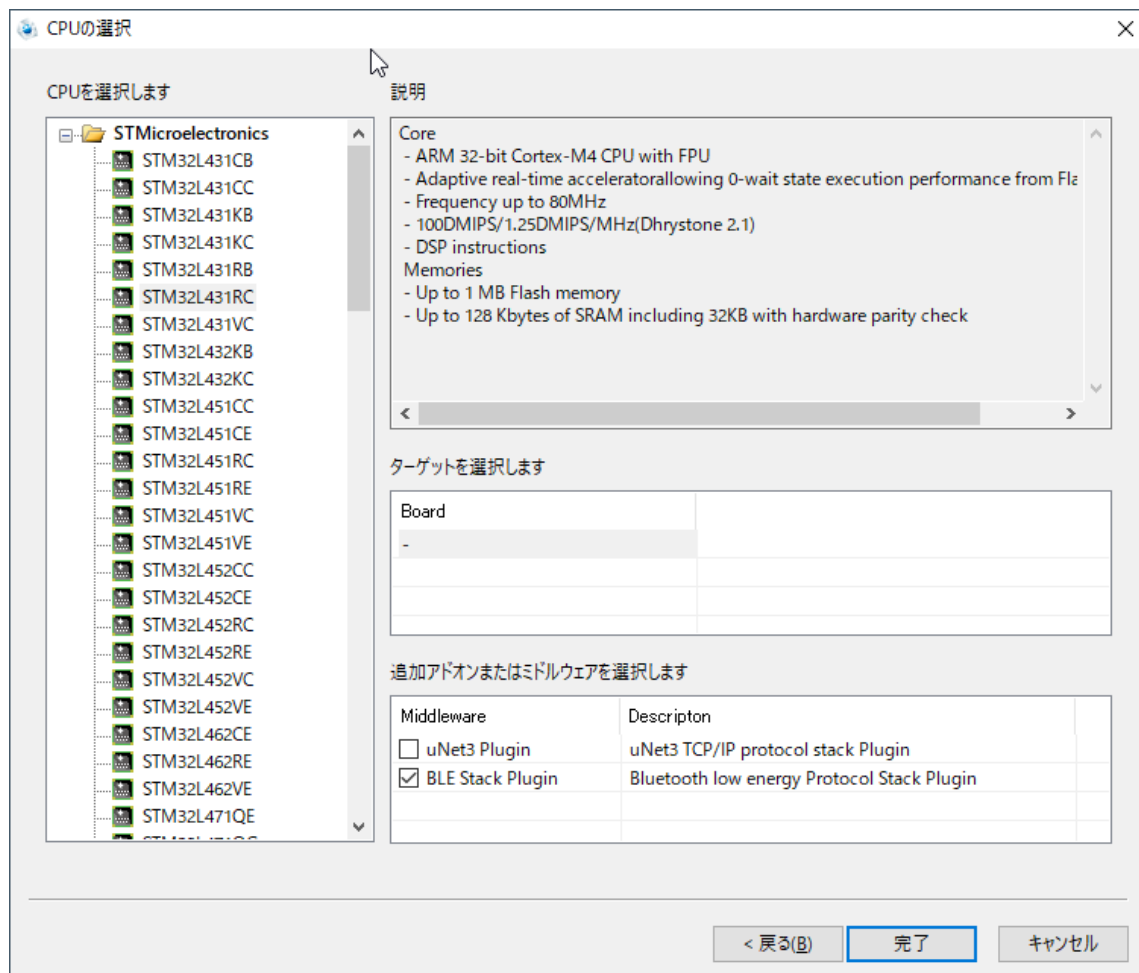
A) 新規でプロジェクトを生成する場合

ツールバーの「新規プロジェクト」をクリックし、CPU の選択へ進みます。



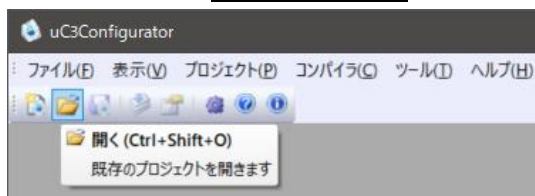
CPU の選択

リストより CPU 型番を選択し、「追加アドオンまたはミドルウェアを選択します」の、「BLE Stack Plugin」にチェックを入れます。「完了」をクリックし、C)メイン画面へ進みます。



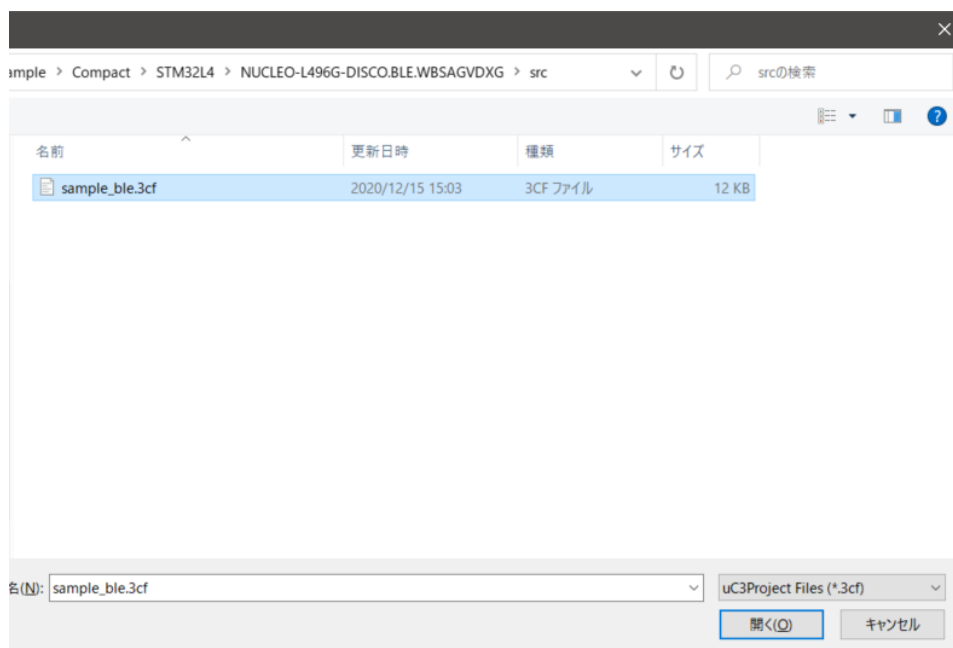
B) 既存のプロジェクトを開く場合

ツールバーの「開く」をクリックし、ファイルを開くへ進みます。



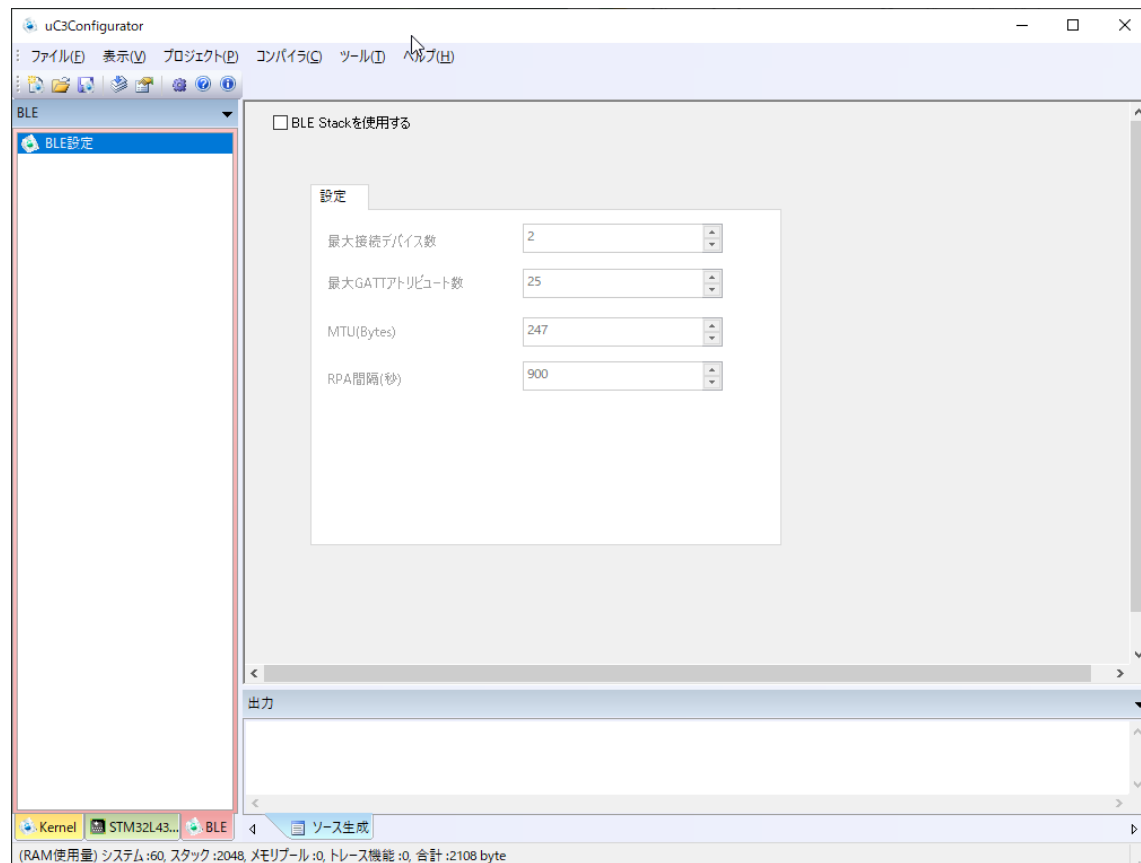
ファイルを開く

プロジェクトファイル(拡張子.3cf)を選択します。「開く」をクリックし、C)メイン画面へ進みます。



C) メイン画面

A)またはB)実行後は、プロジェクトの参照と編集が可能なメイン画面に遷移します。BLEスタックのコンフィグレーションを設定するためには、左側のメニュー画面で「BLE」タブを選択してください。



9.2 BLE スタックの設定

基本設定

① ☒ BLE Stackを使用する

設定

② 最大接続デバイス数 2

③ 最大GATTアトリビュート数 25

④ MTU(Bytes) 247

⑤ RPA間隔(秒) 900

① BLE スタックを使用する

BLE スタックの仕様有無をチェックボックスで指定してください。

チェックを ON にすると、BLE スタックが有効になります。BLE スタックが有効になると、BLE スタックが使用する OS リソースが登録されます。また、②～⑤のパラメータが編集可能になります。

② 最大接続デバイス数

BLE スタックが動作するデバイスと、コネクション可能な最大デバイス数を設定してください。設定可能な値の範囲は、1～25 です。

③ 最大 GATT アトリビュート数

BLE スタックが使用する GATT アトリビュートの数を設定してください。設定可能な値の範囲は、10～65535(※)です。

※デフォルトで 10 個のアトリビュートを使用します。そのため、「10+ アプリケーションで使用するアトリビュート数」を設定してください。

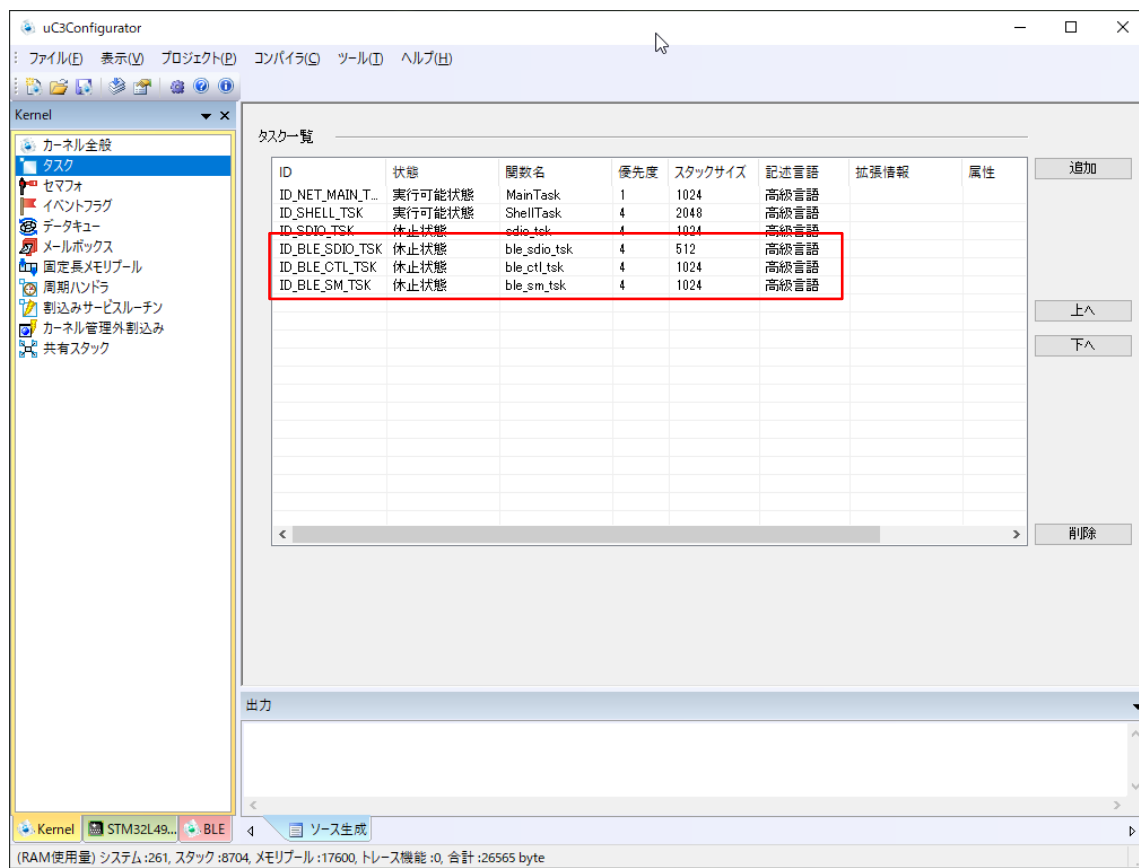
④ MTU

MTU を設定してください。設定可能な値の範囲は、23～247 です。

⑤ RPA 間隔

RPA 間隔を設定してください。設定可能な値の範囲は、1～41400 です。

タスク設定



BLE スタックを有効にすると、タスクに ID_BLE_SDIO_TSK、ID_BLE_CTL_TSK、ID_BLE_SM_TSK が登録されます。各タスクの設定画面から、「優先度の初期値」を変更することができます。上記タスクの優先度を変更する場合、各タスクの優先度を同値に設定してください。

ID_BLE_EVT_MPF 設定

BLE スタックを有効にすると、固定長メモリプールに ID_BLE_SDIO_MPF、ID_BLE_CMD_MPF、ID_BLE_EVT_MPF が追加されます。

上記の固定長メモリプールのうち、ID_BLE_EVT_MPF のみパラメータを変更可能にしています。これは、アプリケーション実装に応じて、メモリブロック数を変更するケースが想定されるためです。このとき、ID_BLE_EVT_MPF のメモリブロック数以外のパラメータを変更したり、ID_BLE_EVT_MPF 自体を削除したりしないよう注意してください。

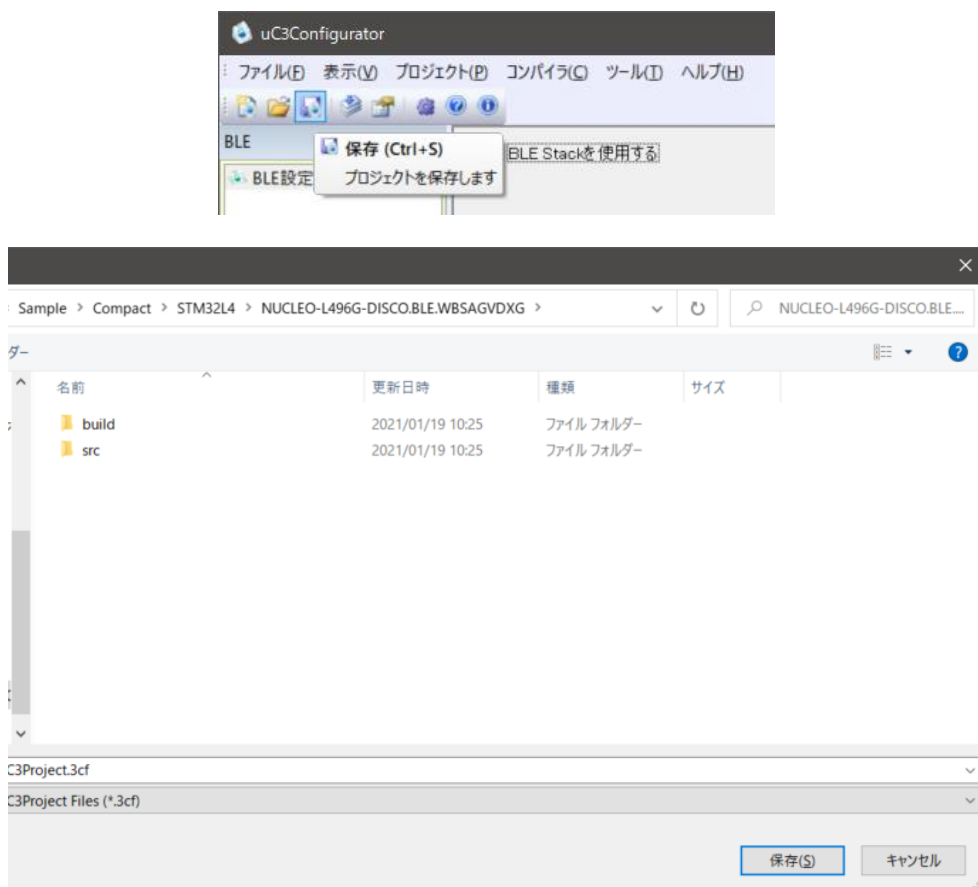
SDIO ドライバコンフィグレーション

本 BLE スタックは、SDIO 経由で BLE コントローラを制御する構成となっております。そのため、SDIO ドライバを使用するためのコンフィグレーションが必要となります。SDIO ドライバのコンフィグレーションについては、「uC3 XXX ドライバ ユーザズガイド デバイス依存部 YYY 編」を参照して設定してください。

※ご使用の製品に合わせて、XXX は BLE コントローラ名、YYY は CPU 名で置き換えてください。

9.3 プロジェクトファイルの保存

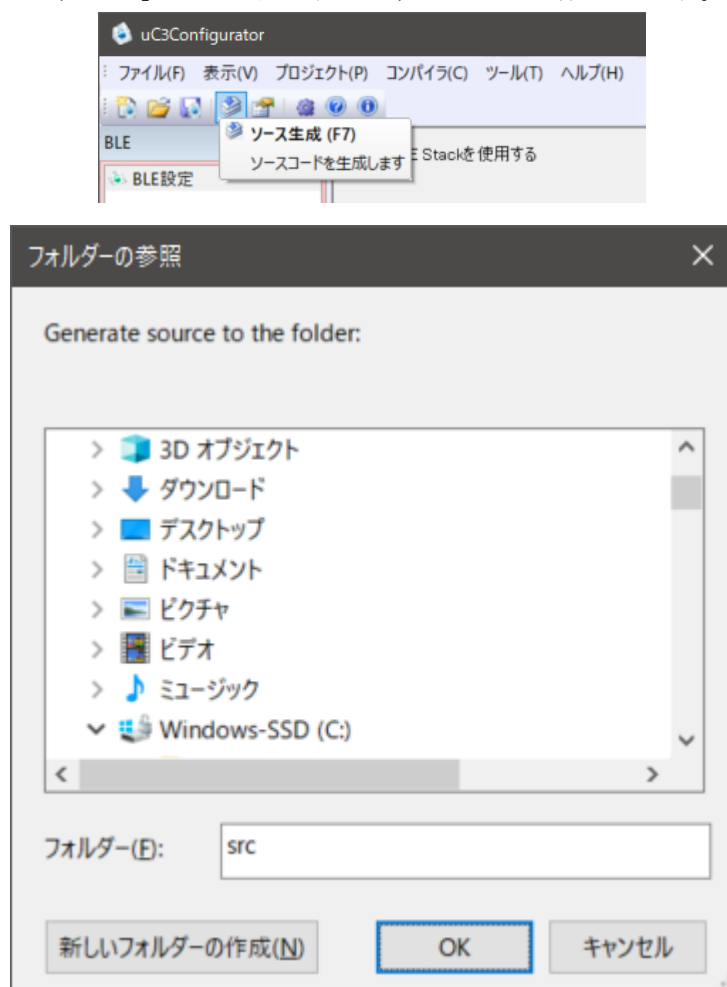
ツールバーの「保存」をクリックすると、「名前を付けて保存」の画面が開きます。プロジェクトファイルの保存先フォルダを指定し、「OK」をクリックします。



保存されるファイルは、プロジェクトファイル(デフォルト：uC3Project.3cf)と、拡張子を.xml に変えたファイルが保存されます。このファイルをブラウザで開くことで、コンフィグレーション情報を確認することができます。

9.4 ソース生成

ツールバーのソース生成をクリックすると、「フォルダーの参照」画面が開きます。任意のフォルダを選択し、「OK」をクリックすると、ソースが生成されます。



生成されるファイル

ファイル	説明
ble_cfg.c ble_cfg.h	BLE スタックのコンフィグレーションコード
ble_strlib.c ble_strlib.h	文字列操作関連のソースファイル
BLE スタックライブラリ	BLE スタックライブラリファイル (.a ファイル及び BLE/inc/配下の.h ファイル群)

μC3 BLE スタックユーザーズガイド

2020 年 9 月	第 1 版
2020 年 10 月	第 2 版
2020 年 11 月	第 3 版
2021 年 1 月	第 4 版
2021 年 1 月	第 5 版
2021 年 10 月	第 6 版

イー・フォース株式会社 <https://www.eforce.co.jp/>

〒103-0014 東京都中央区日本橋富沢町 5-4 ゲンベエビル

TEL 03-5614-6918

FAX 03-5614-6919

お問い合わせ info@eforce.co.jp

Copyright (C) 2020-2021 eForce Co.,Ltd. All Rights Reserved.
