User's Manual

Bluetooth® Low Energy プロトコルスタック

APIリファレンスマニュアル　基本編

ルネサスマイクロコンピュータ

対象デバイス

RL78/G1D

本資料に記載の全ての情報は本資料発行時点のものであり、ルネサス エレクトロニクスは、
予告なしに、本資料に記載した製品または仕様を変更することがあります。
ルネサス エレクトロニクスのホームページなどにより公開される最新情報をご確認ください。

ルネサス エレクトロニクス
www.renesas.com

Rev.1.19　2018.3

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、
   家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、
   金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1.　本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2.　本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.4.0-1　2017.11)

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

このマニュアルの使い方

1. 目的と対象者

このマニュアルは、ルネサス Bluetooth Low Energy マイコン（RL78/G1D）を使用した応用製品の開発に利用するソフトウェア「Bluetooth Low Energy プロトコルスタック」の基本機能の API（Application Program Interface）について説明するものです。本ソフトウェアを用いた応用システムを設計するユーザを対象にしています。このマニュアルを使用するには、マイクロコンピュータ、Bluetooth Low Energy に関する基本的な知識が必要です。

関連資料

関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

資料名	資料番号	
	和文	英文
Bluetooth Low Energy プロトコルスタック		
ユーザーズマニュアル	R01UW0095J	R01UW0095E
API リファレンスマニュアル　基本編	このマニュアル	R01UW0088E
API リファレンスマニュアル　FMP 編	R01UW0089J	R01UW0089E
API リファレンスマニュアル　PXP 編	R01UW0090J	R01UW0090E
API リファレンスマニュアル　HTP 編	R01UW0091J	R01UW0091E
API リファレンスマニュアル　BLP 編	R01UW0092J	R01UW0092E
API リファレンスマニュアル　HOGP 編	R01UW0093J	R01UW0093E
API リファレンスマニュアル　ScPP 編	R01UW0094J	R01UW0094E
API リファレンスマニュアル　HRP 編	R01UW0097J	R01UW0097E
API リファレンスマニュアル　CSCP 編	R01UW0098J	R01UW0098E
API リファレンスマニュアル　CPP 編	R01UW0099J	R01UW0099E
API リファレンスマニュアル　GLP 編	R01UW0103J	R01UW0103E
API リファレンスマニュアル　TIP 編	R01UW0106J	R01UW0106E
API リファレンスマニュアル　RSCP 編	R01UW0107J	R01UW0107E
API リファレンスマニュアル　ANP 編	R01UW0108J	R01UW0108E
API リファレンスマニュアル　PASP 編	R01UW0109J	R01UW0109E
API リファレンスマニュアル　LNP 編	R01UW0113J	R01UW0113E
サンプルプログラムアプリケーションノート	R01AN1375J	R01AN1375E
rBLE コマンド仕様書	R01AN1376J	R01AN1376E

2. 略語および略称の説明

略語／略称	フルスペル	備考
ANP	Alert Notification Profile	
ANS	Alert Notification Service	
API	Application Programming Interface	
ATT	Attribute Protocol	
BAS	Battery Service	
BB	Base Band	
BD_ADDR	Bluetooth Device Address	
BLE	Bluetooth low energy	
BLP	Blood Pressure Profile	
BLS	Blood Pressure Service	
CPP	Cycling Power Profile	
CPS	Cycling Power Service	
CSCP	Cycling Speed and Cadence Profile	
CSCS	Cycling Speed and Cadence Service	
CSRK	Connection Signature Resolving Key	
CTS	Current Time Service	
DIS	Device Information Service	
EDIV	Encrypted Diversifier	
FMP	Find Me Profile	
GAP	Generic Access Profile	
GATT	Generic Attribute Profile	
GLP	Glucose Profile	
GLS	Glucose Service	
HCI	Host Controller Interface	
HID	Human Interface Device	
HIDS	HID Service	
HOGP	HID over GATT Profile	
HRP	Heart Rate Profile	
HRS	Heart Rate Service	
HTP	Health Thermometer Profile	
HTS	Health Thermometer Service	
IAS	Immediate Alert Service	
IRK	Identity Resolving Key	
L2CAP	Logical Link Control and Adaptation Protocol	
LE	Low Energy	
LL	Link Layer	
LLS	Link Loss Service	

LNP	Location and Navigation Profile	
LNS	Location and Navigation Service	
LTK	Long Term Key	
MCU	Micro Controller Unit	
MITM	Man-in-the-middle	
MTU	Maximum Transmission Unit	
OOB	Out of Band	
OS	Operating System	
PASP	Phone Alert Status Profile	
PASS	Phone Alert Status Service	
PXP	Proximity Profile	
RF	Radio Frequency	
RSCP	Running Speed and Cadence Profile	
RSCS	Running Speed and Cadence Service	
RSSI	Received Signal Strength Indication	
ScPP	Scan Parameters Profile	
ScPS	Scan Parameters Service	
SM	Security Manager	
SMP	Security Manager Protocol	
STK	Short Term Key	
TK	Temporary Key	
TPS	Tx Power Service	
UART	Universal Asynchronous Receiver Transmitter	
UUID	Universal Unique Identifier	

略語／略称	フルスペル	備考
APP	Application	
CSI	Clocked Serial Interface	
IIC	Inter-Integrated Circuit	
RSCIP	Renesas Serial Communication Interface Protocol	
VS	Vendor Specific	

Bluetooth は、Bluetooth SIG, Inc., U.S.A.の登録商標です。
すべての商標および登録商標は、それぞれの所有者に帰属します。

目次

1. はじめに

このマニュアルは、ルネサス Bluetooth Low Energy マイコン（RL78/G1D）を使用した Bluetooth 応用製品の開発に利用するソフトウェア「Bluetooth Low Energy プロトコルスタック」(以降、BLE ソフトウェア)の基本機能の API について説明しています。

「BLE ソフトウェア」のソフトウェア構成および機能の詳細につきましては、「Bluetooth Low Energy プロトコルスタック ユーザーズマニュアル」を参照下さい。

2. 概要

2.1 BLE ソフトウェアとその API

「BLE ソフトウェア」とは Bluetooth Low Energy (Bluetooth v4.2)に対応した BLE スタックを含むソフトウェア一式を指します。

図 2-1 に BLE ソフトウェアの構成図を示します。

BLE ソフトウェアは、アプリケーションが RL78/G1D に搭載される構成(以降、Embedded 構成)と、別の MCU に搭載される構成(以降、Modem 構成)で動作し、両構成で、同じアプリケーションを使用することが可能な API を提供します。

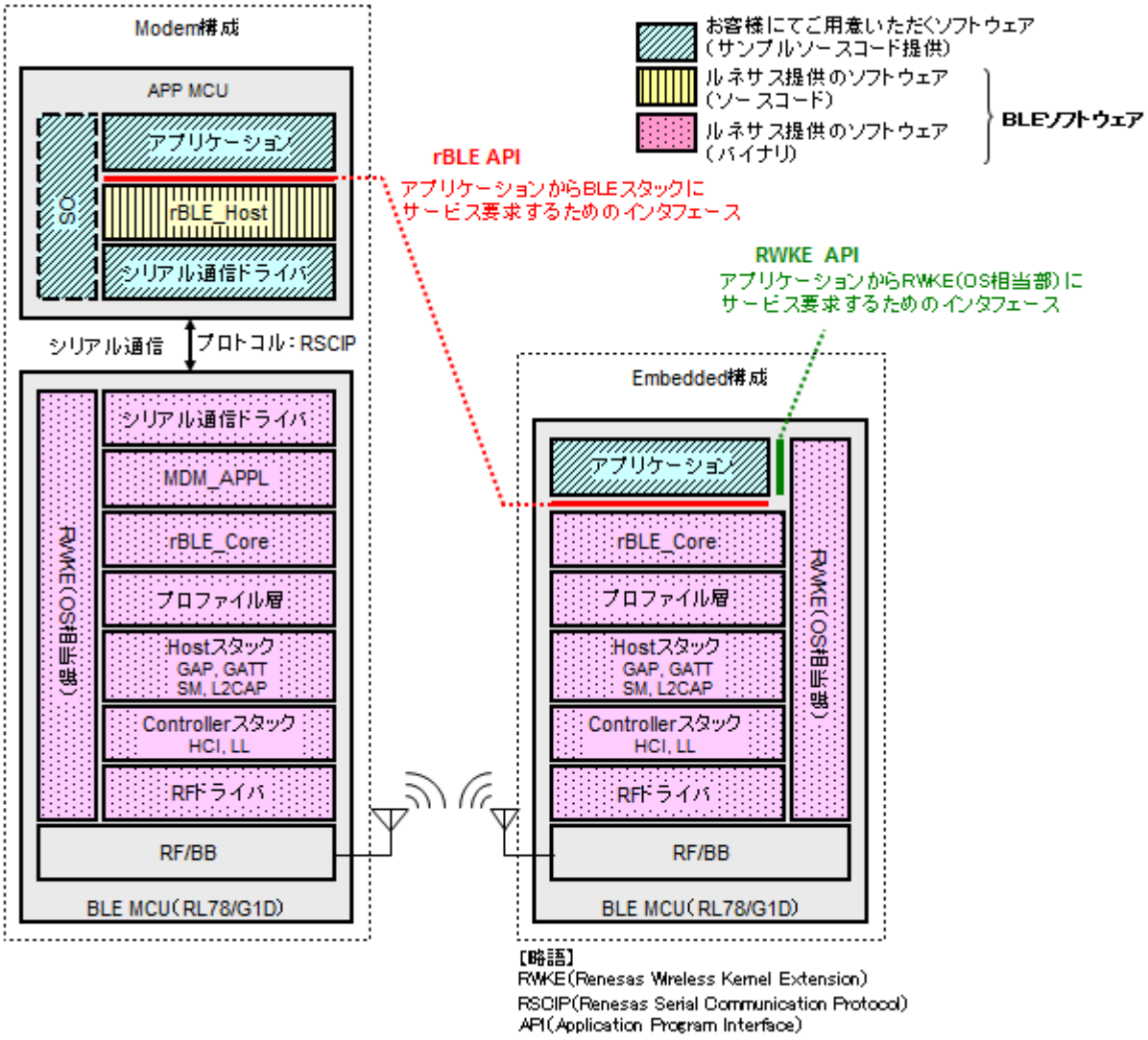


図 2-1 BLE ソフトウェア構成図

Modem 構成の BLE ソフトウェアは、APP MCU と BLE MCU(RL78/G1D)の 2 つのチップで動作し、APP MCU で動作する「rBLE_HOST」部（図の ）と BLE MCU で動作するソフトウェア（図の ）で構成されます。

また、お客さまにてご用意いただくソフトウェア（図の ）は、APP MCU の「アプリケーション」部と「UART ドライバ」部、および「OS」部になります。ただし、「rBLE_Host」部は OS のリソースを使用していないため、APP MCU に OS が搭載されていない場合には、「OS」部のソフトウェアを用意する必要はありません。

APP MCU で動作するアプリケーションは、rBLE_Host を介して BLE MCU と BLE サービスのやり取りが行われます。APP MCU と BLE MCU は物理的に UART で接続され、rBLE_Host の制御により RSCIP (Renesas Serial Communication Interface Protocol)を使用した通信が行われます。

一方、Embedded 構成時の BLE ソフトウェアは、BLE MCU(RL78/G1D)のみの 1 チップで動作します。お客様にてご用意いただくソフトウェアは、「アプリケーション」部のみとなり、BLE MCU 上に実装されます。

本ドキュメント記載の「BLE ソフトウェア」の API は、図 2-1 に示す「rBLE API」と「RWKE API」に相当します。「rBLE API」は、アプリケーションから BLE スタックにサービス要求をするための API であり、「RWKE API」は、BLE ソフトウェアを動作させるために設計された簡易オペレーティングシステムである RWKE（Renesas Wireless Kernel Extention）にサービス要求をするための API です。

2.2 rBLE API

2.2.1 使用言語

rBLE API の使用記述言語は、C 言語になります。

2.2.2 rBLE API の手続き

本節は「rBLE API」の手続きについて説明します。

図 2-2、図 2-3 に示すように、アプリケーションから rBLE_HOST、もしくは rBLE_Core へのコマンド要求については、API 関数コールを使用します。また、rBLE_HOST、もしくは rBLE_Core からアプリケーションへのイベント通知に関しては、イベント通知用のコールバック関数がコールされます。

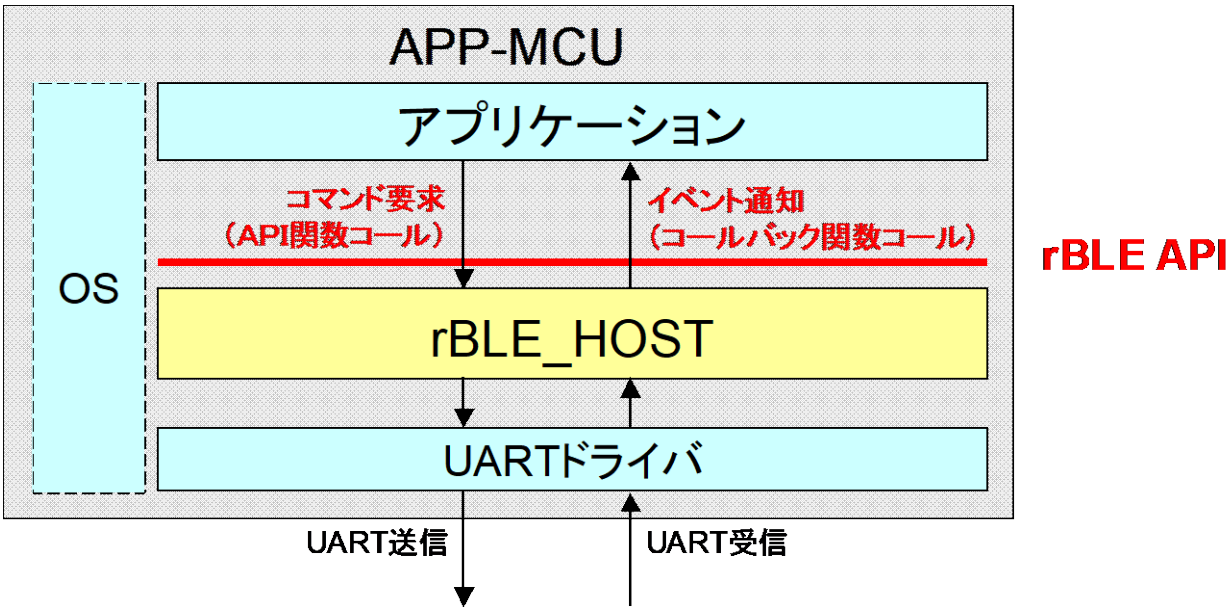


図 2-2 rBLE API の手続き（Modem 構成時）

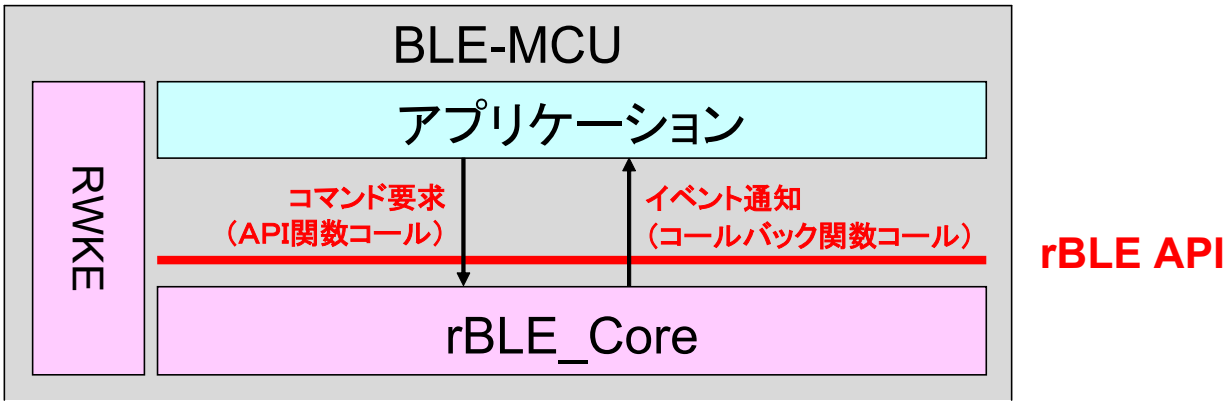


図 2-3 rBLE API の手続き（Embedded 構成時）

アプリケーションから発行されたコマンドの処理結果は、API 関数の戻り値や、API 関数呼び出しとは非同期に発生するイベントとして通知されます。

コマンド要求時にコールする API 関数の戻り値がエラーの場合、そのコマンド要求は処理されません。したがって、コマンド要求に関連するイベント通知も発生しません。基本的な API 関数のエラーケースとしては、パラメータ値異常や処理できない状態が挙げられます。

2.2.3 rBLE API の機能分類

本節は、rBLE API の機能分類について説明します。

「rBLE API」は、基本機能の Initialization、Generic Access Profile(GAP)、Security Manager(SM)、Generic Attribute Profile (GATT)、Vendor Specific(VS)と GATT-based Profile に機能が分類されます。表 2-1 にその機能分類と概要についてまとめます。

表 2-1 API 機能分類

機能	略称	概要
Initialization	INIT	BLE ソフトウェアの初期化処理を行います。
Generic Access Profile	GAP	周辺デバイスの検索や、ピアデバイスとの接続・切断等のリンク管理、セキュリティ要件に応じた各種手続きを行います。
Security Manager	SM	2 デバイス間のセキュリティを確保するためにペアリングを行い通信内容の暗号化またはデータ署名を行います。またそれらに必要となるデバイス同士の情報交換を行います。
Generic Attribute Profile	GATT	クライアントからサーバの公開する特性値のハンドルを取得することが可能です。(使用可能な機能は一部に限定されます)
Vendor Specific	VS	Direct Test Mode やルネサス独自の Direct Test Mode 拡張機能等を提供します。
GATT-based Profile	-	Bluetooth Low Energy プロトコルスタック・ユーザーズマニュアルの 7.機能説明を参照してください。

4 章以降に記載の rBLE API の詳細説明についても、表 2-1 の機能に分類して説明いたします。

また、C 言語記述において各機能の略称を用いている箇所が多数あります。機能分類に対する略称は表中の略称をご参照下さい。

2.2.4 rBLE API 関数のパラメータ扱い

本書に記載の API の引数においてポインタ型の引数が多数ありますが、ポインタで示されるアドレスのメモリ領域については全て入力扱いです。API 関数内がこのメモリ領域を書き換えることは一切ありません。

2.2.5 イベント通知用コールバック関数の登録

本節は、rBLE API のイベント通知用コールバック関数の主に取り扱いについて説明します。

イベント通知用コールバック関数は、お客様でご用意いただく関数となります。これは、イベント通知に対する応答動作のプログラムをイベント通知用コールバック関数内で処理できるようにするためです。

そのため、イベント通知用コールバック関数は、そのコールバック関数を登録する必要があります。また、イベント通知用コールバック関数は、2.2.3 記載の機能毎に登録する仕様となっております。表 2-2 に機能毎にイベント通知用コールバック関数を登録するための関数をまとめます。

表 2-2 各機能のコールバック登録関数一覧

機能	略称	コールバック登録関数	備考
Initialization	INIT	RBLE_Init	
Generic Access Profile	GAP	RBLE_GAP_Reset	GAP と SM は同時タイミングでコールバック関数を登録するため、同一関数を使用。
Security Manager	SM		
Generic Attribute Profile	GATT	RBLE_GATT_Enable	
Vendor Specific	VS	RBLE_VS_Enable	
GATT-based Profile	-	RBLE_xxx_***_Enable	Bluetooth Low Energy プロトコルスタック・API リファレンスマニュアルの各プロファイル編に記載されているロール毎の Enable 関数を参照してください。

※xxx には GATT-based Profile の略称が、***にはロール名が入ります。

GAP と SM は同時タイミングで登録するため、RBLE_GAP_Reset 関数にて同時にコールバック関数を登録します。

また、GATT-based Profile に関しては、仕様で規定されている役割（ロール）毎にコールバック関数の登録が必要です。この理由は、通信するにあたって、互いのデバイスがどちらか一方のロールを担うことになり、基本的には片側のロールしか使用しないため、リソース軽減の目的で分離されています。

図 2-4に、イベント通知用コールバック関数の登録を行っているプログラム例を示します。

```
/* GAP イベント通知用コールバック関数 */
void GAP_CallBack( RBLE_GAP_EVENT *event )
{
     switch( event->type ) {
            case RBLE_GAP_EVENT_RESET_RESULT:
                  /* イベント処理入力 */
                  break;
            default:
                  break;
     }
}

/* SM イベント通知用コールバック関数 */
void SM_CallBack( RBLE_SM_EVENT *event )
{
     switch( event->type ) {
            case RBLE_SM_EVENT_SET_CNF:
                  /* イベント処理入力 */
                  break;
            default:
                  break;
     }
}

/* GAP リセット処理 */
void GAP_Reset_Function( void )
{
     RBLE_GAP_Reset( &GAP_CallBack, &SM_CallBack );
}
```

図 2-4 イベント通知用コールバック関数の登録プログラム例

2.2.6 イベント通知用コールバック関数の基本動作

本節は、イベント通知用コールバック関数の基本動作について説明します。

イベント通知用コールバック関数は、rBLE_HOST（Embedded 構成の場合は rBLE_Core）から発生したイベント毎にイベントタイプを指定して、そのイベントタイプ毎に定義されたデータフォーマットでデータをアプリケーションに渡します。図 2-5 に、FMP の Target ロールで使用されるイベントタイプのデータ構造体を示します。

```
/* FMP の Target ロールのイベントタイプデータ構造体 */
typedef struct RBLE_FMPT_EVENT_t
{
     RBLE_FMP_EVENT_TYPE            type;              /* イベントタイプ */
     uint8_t                        reserved;
     union Event_Fmt_Parameter_u {
          /* RBLE_EVT_FMP_Target_Enable_Comp */
          struct RBLE_FMP_Target_Enable_t{
               RBLE_STATUS          status;
               uint8_t              reserved;
               uint16_t             conhdl;
          }target_enable;

          /* RBLE_EVT_FMP_Target_Disable_Comp */
          struct RBLE_FMP_Target_Disable_t{
               RBLE_STATUS          status;
               uint8_t              reserved;
               uint16_t             conhdl;
          }target_disable;

          /* RBLE_EVT_FMP_Target_Alert_Ind */
          struct RBLE_FMP_Target_Alert_Ind_t{
               uint16_t             conhdl;
               uint8_t              alert_lvl;
               uint8_t              reserved;
          }target_alert_ind;

          /* RBLE_EVT_FMP_CMD_DISALLOWED_IND */
          struct RBLE_FMP_Target_Command_Disallowed_Ind_t{
               RBLE_STATUS          status;
               uint8_t              reserved;
               uint16_t             opcode;
          }cmd_disallowed_ind;
     }param;
}RBLE_FMPT_EVENT;
```

図 2-5 FMP Target ロールのイベントタイプデータ構造体

図 2-5 の構造体は、イベントタイプ通知用の type メンバとイベントタイプ毎のデータフォーマットが共用体で 4 つ分（target_enable、target_disable、target_alert_ind、cmd_err_ind）定義されています。

これに対して、イベントを通知されたコールバック関数側の処理を図 2-6 に示します。

```
void FMPT_CallBack( RBLE_FMPT_EVENT *event )
{
     switch( event->type){
        case RBLE_FMP_EVENT_TARGET_ENABLE_COMP:
            /* イベント処理記入 */
            break;
        case RBLE_FMP_EVENT_TARGET_DISABLE_COMP:
            /* イベント処理記入 */
            break;
        case RBLE_FMP_EVENT_TARGET_ALERT_IND:
            /* イベント処理記入 */
            break;
        case RBLE_FMP_EVENT_TARGET_COMMAND_ERROR_IND:
            /* イベント処理記入 */
            break;
        default:
            break;
    }
}
```

図 2-6 FMP Target ロールのイベント通知用コールバック関数例

図 2-6 のコールバック関数では、FMP Target ロールで起きる 4 つのイベントに対して応答処理が行えるようにプログラムされています。まず、event->type にてイベントタイプを判別し、switch 文で処理を分岐しています。そして、各イベントの応答処理を組み込むことで、アプリケーションを実装してください。

2.3 BLE ソフトウェアの状態遷移

本節は、BLE ソフトウェアの状態定義とその状態遷移について説明します。

まず、BLE ソフトウェアには 3 つの状態があり、rBLE_Core の状態を「rBLE コアモード」、rBLE_HOST の状態を「rBLE モード」と呼びます。これらのモードは、モード変化通知用コールバック RBLE_INIT_CB で通知されます。

2.3.1 rBLE_Core の状態遷移

表 2-3 に rBLE コアモードの状態定義を示します。

表 2-3 rBLE コアモードの定義

rBLE コアモード	説明
RBLE_MODE_RESET	rBLE_Core が初期化前の状態であることを意味します。 RBLE_Init()関数コールにより、RBLE_MODE_INITIALIZE に移行します。
RBLE_MODE_INITIALIZE	rBLE_Core が初期化中を意味します。 初期化が完了すると、RBLE_MODE_ACTIVE に移行します。
RBLE_MODE_ACTIVE	rBLE_Core がアクティブ状態（動作可能）であることを意味します。 一度、この状態に遷移すると、初期化を行わない限り、他の状態に遷移することはありません。

次に、rBLEコアモードの状態遷移図を以下に示します。

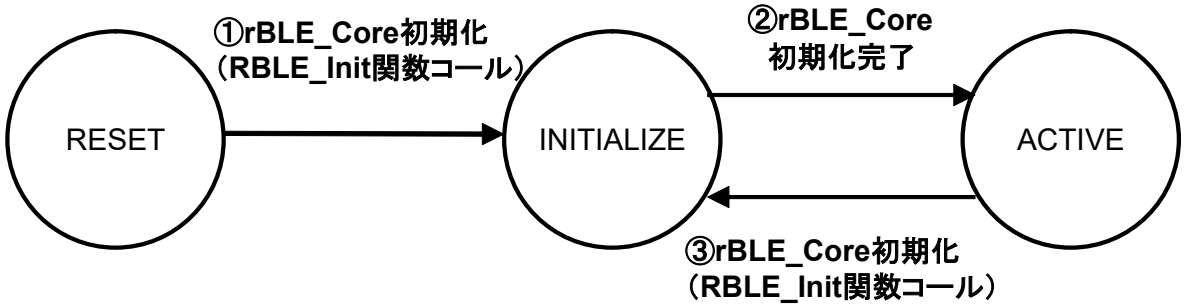


図 2-7 rBLE_Core の状態遷移図

状態遷移①は、rBLE_Core の初期化タイミングです。アプリケーション（Modem 構成の場合は MDM APPL）が rBLE_Core の初期化を行う際に RBLE_Init 関数をコールするタイミングになります。

状態遷移②は、rBLE_Core の初期化完了タイミングです。アプリケーションから RBLE_Init がコールされた後、rBLE_Core が初期化を完了したタイミングになります。

状態遷移③は、rBLE_Core のリセットタイミングです。アクティブ状態からアプリケーションが rBLE_Core をリセットする際に RBLE_Init 関数をコールするタイミングになります。

2.3.2 rBLE_HOST の状態遷移

表 2-4 に rBLE モードの状態定義を示します。

表 2-4 rBLE モードの定義

rBLE モード	説明
RBLE_MODE_INITIALIZE	rBLE_HOST が初期化中を意味します。 RBLE_Init()関数コールにより RBLE_MODE_INITIALIZE に移行します。 初期化が完了すると、RBLE_MODE_ACTIVE に移行します。
RBLE_MODE_ACTIVE	rBLE_HOST がアクティブ状態（動作可能）であることを意味します。 RSCIP 接続がリセットされると、RBLE_MODE_RESET に移行します。
RBLE_MODE_RESET	RSCIP 接続がリセットされたことにより、リセット処理中であることを意味します。 リセットが完了すると、RBLE_MODE_ACTIVE に移行します。

次に、rBLEモードの状態遷移図を以下に示します。

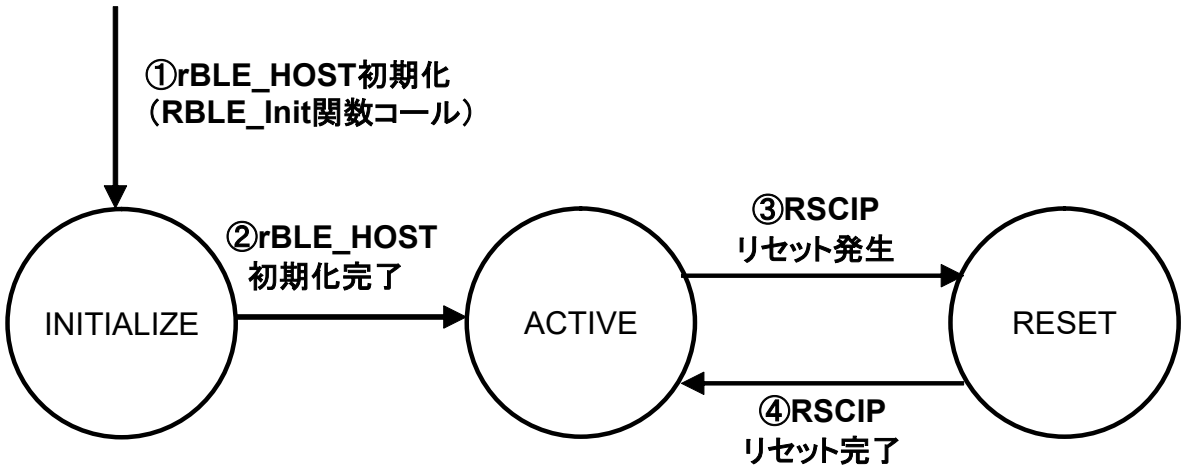


図 2-8 rBLE_HOST の状態遷移図

状態遷移①は、rBLE_HOST の初期化タイミングです。アプリケーションが rBLE_HOST の初期化を行う際に RBLE_Init 関数をコールするタイミングになります。

状態遷移②は、rBLE_HOST の初期化完了タイミングです。アプリケーションから RBLE_Init がコールされた後、rBLE_HOST が初期化を完了したタイミングになります。

状態遷移③は、RSCIP の接続がリセットになるタイミングです。APP MCU および BLE MCU 側のどちらかで異常状態検知によるリセット発生や通信異常による通信リセットにより、本状態遷移が発生します。

状態遷移④は、RSCIP の接続がリセット完了になるタイミングです。本状態遷移は、RSCIP の接続がリセットされたことに起因して、そのリセット処理が完了し再度使用可能な状態に移行します。

2.4 BLE ソフトウェアの初期化手順

本節は、BLE ソフトウェアの初期化手順について説明します。
下図に、初期化手順の例として、FMP Target ロール応用の初期化手順をシーケンスチャートで示します。

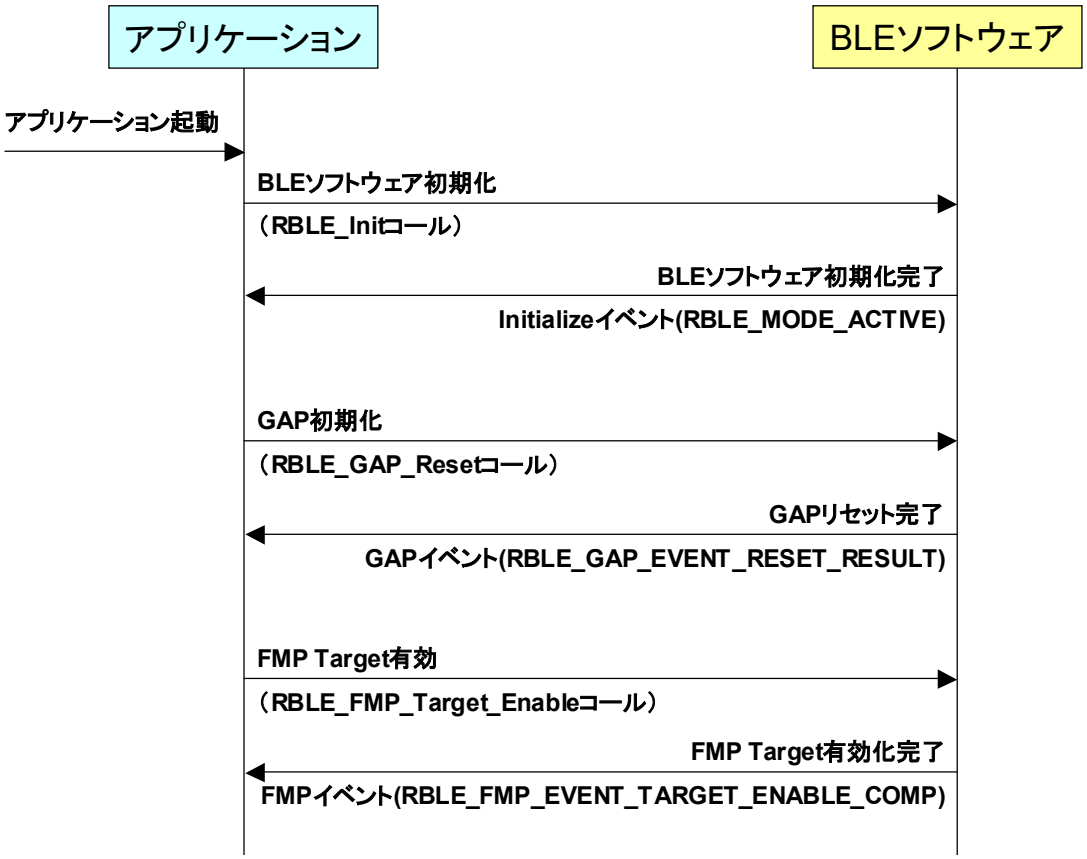


図 2-9 BLE ソフトウェアの初期化手順例

BLE ソフトウェアは、Initialize 機能に属す RBLE_Init 関数のコールにより初期化が行われます。初期化の完了は、rBLE モード変化通知用コールバックで通知され、その状態は RBLE_MODE_ACTIVE が通知されます。
次に、GAP と SM 機能を有効にするため、RBLE_GAP_Reset 関数をコールします。これに対し、GAP リセット完了を通知するイベント RBLE_GAP_EVENT_RESET_RESULT が通知されます。
最後に、実装する応用製品により決まるプロファイル機能の有効設定（図では FMP Target ロールのため、RBLE_FMP_Target_ Enable）が必要です。これに対し、プロファイル機能の有効化完了を通知するイベント（図では RBLE_FMP_EVENT_TARGET_ENABLE_COMP）で、プロファイル機能が使用可能です。

3. Common Definitions

このセクションは、rBLE API の共通定義について記載します。

3.1 Standard Typedef

- 型宣言

```
typedef   unsigned char      uint8_t;          符号なし 8bit 整数型
typedef   unsigned short     uint16_t;         符号なし 16bit 整数型
typedef   unsigned long      uint32_t;         符号なし 32bit 整数型
typedef   signed char        int8_t;           符号付き 8bit 整数型
typedef   signed short       int16_t;          符号付き 16bit 整数型
typedef   signed long        int32_t;          符号付き 32bit 整数型
typedef   unsigned char      bool;             ブール型
typedef   signed int         int_t;            符号付き int 型
typedef   unsigned int       uint_t;           符号なし int 型
typedef   char               char_t;           文字列型
```

3.2 Generic Definitions

- 定数定義

```
#define RBLE_BD_ADDR_LEN              0x06      Bluetooth デバイスアドレス長
#define RBLE_BD_NAME_SIZE             0x41      Bluetooth デバイス名長
#define RBLE_ADV_DATA_LEN             0x1F      Advertising データバイト数
#define RBLE_SCAN_RSP_DATA_LEN        0x1F      Scan レスポンスデータバイト数
#define RBLE_KEY_LEN                  0x10      Key 長
#define RBLE_LE_FEATS_LEN             0x08      フィーチャー長
#define RBLE_LE_CHNL_MAP_LEN          0x05      チャネルマップ長
#define RBLE_ATTM_MAX_VALUE           0x18      アトリビュート値最大長
#define RBLE_RAND_NB_LEN              0x08      乱数長
#define RBLE_MASTER                   0x00      マスタロール
#define RBLE_SLAVE                    0x01      スレーブロール
```

- rBLE ステータスの型宣言

```
typedef uint8_t         RBLE_STATUS;
```

- rBLE ステータス列挙型宣言

```
enum RBLE_STATUS_enum {
  RBLE_OK                                = 0x00,   正常動作
  RBLE_UNKNOWN_HCI_COMMAND               = 0x01,   不明なコマンドを受信
  RBLE_UNKNOWN_CONNECTION_ID             = 0x02,   不明なコネクション ID が指定された
  RBLE_HARDWARE_FAILURE                  = 0x03,   ハードウエアエラー発生
  RBLE_PAGE_TIMEOUT                      = 0x04,   ページタイムアウト発生
  RBLE_AUTH_FAILURE                      = 0x05,   認証失敗
  RBLE_PIN_MISSING                       = 0x06,   PIN コードが不明
  RBLE_MEMORY_CAPA_EXCEED                = 0x07,   メモリの容量を超えた
  RBLE_CON_TIMEOUT                       = 0x08,   接続タイムアウト発生
  RBLE_CON_LIMIT_EXCEED                  = 0x09,   接続数が上限に達した
  RBLE_COMMAND_DISALLOWED                = 0x0C,   コマンドは許可されない
  RBLE_CONN_REJ_LIMITED_RESOURCES        = 0x0D,   リソース制限により接続が拒否された
  RBLE_CONN_REJ_SECURITY_REASONS         = 0x0E,   セキュリティにより接続が拒否された
  RBLE_CONN_REJ_UNACCEPTABLE_BDADDR      = 0x0F,   不許可 BD ADDR のため接続が拒否された
  RBLE_CONN_ACCEPT_TIMEOUT_EXCEED        = 0x10,   接続許諾待ちタイムアウト発生
  RBLE_UNSUPPORTED                       = 0x11,   未サポートである
  RBLE_INVALID_HCI_PARAM                 = 0x12,   不正なパラメータが指定された
  RBLE_REMOTE_USER_TERM_CON              = 0x13,   リモートユーザにより切断
  RBLE_REMOTE_DEV_TERM_LOW_RESOURCES     = 0x14,   リソース不足により切断
  RBLE_REMOTE_DEV_POWER_OFF              = 0x15,   リモートデバイスの電源 OFF
  RBLE_CON_TERM_BY_LOCAL_HOST            = 0x16,   ローカルホストにより切断
  RBLE_REPEATED_ATTEMPTS                 = 0x17,   ペアリング・認証が最大試行数に達した
  RBLE_PAIRING_NOT_ALLOWED               = 0x18,   ペアリングは許可されない
  RBLE_UNSUPPORTED_REMOTE_FEATURE        = 0x1A,   リモートデバイスが未サポート
  RBLE_UNSPECIFIED_ERROR                 = 0x1F,   不明なエラー
  RBLE_LMP_RSP_TIMEOUT                   = 0x22,   LMP/LL のレスポンスタイムアウト発生
  RBLE_ENC_MODE_NOT_ACCEPT               = 0x25,   要求された Encryption モードは不許可
  RBLE_LINK_KEY_CANT_CHANGE              = 0x26,   リンクキーは変更できない
  RBLE_INSTANT_PASSED                    = 0x28,   実行時間を経過
  RBLE_PAIRING_WITH_UNIT_KEY_NOT_SUP     = 0x29,   UNIT キーによるペアリングは未サポート
  RBLE_DIFF_TRANSACTION_COLLISION        = 0x2A,   複数のトランザクションの衝突発生
  RBLE_CHANNEL_CLASS_NOT_SUP             = 0x2E,   チャネルアセスメントモードは未サポート
  RBLE_INSUFFICIENT_SECURITY             = 0x2F,   セキュリティ不足によるエラー
  RBLE_PARAM_OUT_OF_MAND_RANGE           = 0x30,   パラメータが必須サポート範囲外
  RBLE_SP_NOT_SUPPORTED_HOST             = 0x37,   ホストが SSP を未サポート
  RBLE_HOST_BUSY_PAIRING                 = 0x38,   別のペアリング中につきペアリング不可
  RBLE_CONTROLLER_BUSY                   = 0x3A,   別の処理中につき実行不可
  RBLE_UNACCEPTABLE_CONN_INT             = 0x3B,   指定された接続パラメータは受入れられない
  RBLE_DIRECT_ADV_TO                     = 0x3C,   Directed Advertising タイムアウト
  RBLE_TERMINATED_MIC_FAILURE            = 0x3D,   受信パケットのメッセージ不完全のため切断
  RBLE_CONN_FAILED_TO_BE_ES              = 0x3E,   接続確立に失敗

  RBLE_GAP_INVALID_PARAM_ERR             = 0x40,   GAP 不正パラメータエラー
  RBLE_GAP_AUTO_EST_ERR,                           GAP 自動接続エラー
```

```
  RBLE_GAP_SELECT_EST_ERR,                    GAP 選択接続エラー
  RBLE_GAP_SET_RECON_ADDR_ERR,                GAP 再接続アドレス設定エラー
  RBLE_GAP_SET_PRIVACY_FEAT_ERR,              GAP プライバシーフィーチャー設定エラー

  RBLE_GATT_INVALID_PARAM_ERR       = 0x50,   GATT 不正パラメータエラー
  RBLE_GATT_INDICATE_NOT_ALLOWED,             GATT 表示不可
  RBLE_GATT_NOTIFY_NOT_ALLOWED,               GATT 通知不可
  RBLE_GATT_INVALID_TYPE_IN_SVC_SEARCH,       GATT 不正サービスサーチタイプエラー
  RBLE_GATT_ATTRIBUTE_CLIENT_MISSING,         GATT ATT Client 無効エラー
  RBLE_GATT_ATTRIBUTE_SERVER_MISSING,         GATT ATT Server 無効エラー
  RBLE_GATT_RELIABLE_WRITE_ERR,               GATT 信頼性書き込みエラー
  RBLE_GATT_BUFF_OVER_ERR,                    GATT バッファオーバーエラー

  RBLE_ATT_INVALID_PARAM_ERR        = 0x60,   ATT 不正パラメータエラー

  RBLE_SM_INVALID_PARAM_ERR         = 0x70,   SM 不正パラメータエラー
  RBLE_SM_PAIR_ERR_PASSKEY_ENTRY_FAILED,      パスキー入力が不正
  RBLE_SM_PAIR_ERR_OOB_NOT_AVAILABLE,         OOB データが利用可能ではない
  RBLE_SM_PAIR_ERR_AUTH_REQUIREMENTS,         認証要件を満たさない
  RBLE_SM_PAIR_ERR_CFM_VAL_FAILED,            Confirm Value の不一致
  RBLE_SM_PAIR_ERR_PAIRING_NOT_SUPPORTED,     ペアリングは未サポート
  RBLE_SM_PAIR_ERR_ENCRYPTION_KEY_SIZE,       暗号化キーサイズが不正
  RBLE_SM_PAIR_ERR_CMD_NOT_SUPPORTED,         未サポートの SMP コマンドを受信
  RBLE_SM_PAIR_ERR_UNSPECIFIED_REASON,        不明なエラーによりペアリング失敗
  RBLE_SM_PAIR_ERR_REPEATED_ATTEMPTS,         短時間にペアリング試行回数の上限に達した
  RBLE_SM_PAIR_ERR_INVALID_PARAMS,            パラメータが不正

  RBLE_L2C_INVALID_PARAM_ERR        = 0x80,   L2CAP 不正パラメータエラー

  RBLE_ERR,                         = 0xF0,   エラー
  RBLE_TRANS_ERR                    = 0xF1,   通信エラー
  RBLE_STATUS_ERROR                 = 0xF2,   ステータスエラー
  RBLE_PARAM_ERR                    = 0xF3,   パラメータエラー
  RBLE_BUSY                         = 0xF4,   ビジーエラー発生
  RBLE_SHORTAGE_OF_RESOURCE         = 0xF5,   リソース不足
  RBLE_EXIT                         = 0xF6,   終了
  RBLE_VERSION_FAIL                 = 0xF7,   ライブラリ組み合わせエラー
  RBLE_TEST_VERSION                 = 0xF8    BLE ソフトウェアはテストバージョン
};
```

【注】プロファイル特有のステータスは、API リファレンスマニュアルの各プロファイル編に記載しています。

- ATT エラーコード列挙型宣言

```
enum RBLE_ATT_ERR_CODE_enum {
  RBLE_ATT_ERR_NO_ERROR             = 0x00,   正常終了
  RBLE_ATT_ERR_INVALID_HANDLE,                ハンドルが不正
  RBLE_ATT_ERR_READ_NOT_PERMITTED,            読み出しは許可されない
```

```
    RBLE_ATT_ERR_WRITE_NOT_PERMITTED,            書き込みは許可されない
    RBLE_ATT_ERR_INVALID_PDU,                    PDU が不正
    RBLE_ATT_ERR_INSUFF_AUTHEN,                  要求には認証が必要
    RBLE_ATT_ERR_REQUEST_NOT_SUPPORTED,          その要求は未サポート
    RBLE_ATT_ERR_INVALID_OFFSET,                 オフセットが不正
    RBLE_ATT_ERR_INSUFF_AUTHOR,                  要求には許可が必要
    RBLE_ATT_ERR_PREPARE_QUEUE_FULL,             キューが一杯
    RBLE_ATT_ERR_ATTRIBUTE_NOT_FOUND,            アトリビュートが見つからない
    RBLE_ATT_ERR_ATTRIBUTE_NOT_LONG,             アトリビュートが長くない
    RBLE_ATT_ERR_INSUFF_ENC_KEY_SIZE,            暗号化キーサイズが不十分
    RBLE_ATT_ERR_INVALID_ATTRIBUTE_VAL_LEN,      アトリビュート値サイズが不正
    RBLE_ATT_ERR_UNLIKELY_ERR,                   予期しないエラーが発生
    RBLE_ATT_ERR_INSUFF_ENC,                     要求には暗号化が必要
    RBLE_ATT_UNSUPP_GRP_TYPE,                    指定グループタイプは未サポート
    RBLE_ATT_INSUFF_RESOURCE,                    リソース不足
    RBLE_ATT_ERR_APP_ERROR             = 0x80,   アプリケーションエラー
    RBLE_ATT_ERR_IMPROPERLY_CONFIGURED= 0xFD,    Configuration 設定が不適切
    RBLE_ATT_ERR_ALREADY_IN_PROGRESS   = 0xFE,   手順が進行中
    RBLE_ATT_ERR_OUT_OF_RANGE          = 0xFF,   設定が範囲外
};
```

- Bluetooth デバイス名構造体宣言

```
typedef struct RBLE_BD_NAME_t {
    uint8_t             namelen;                    デバイス名称長
    uint8_t             name[RBLE_BD_NAME_SIZE];    Bluetooth デバイス名
} RBLE_BD_NAME;
```

- Bluetooth デバイスアドレス構造体宣言

```
typedef struct RBLE_BD_ADDR_t {
    uint8_t             addr[RBLE_BD_ADDR_LEN];     Bluetooth デバイスアドレス
} RBLE_BD_ADDR;
```

- Bluetooth チャネルマップ構造体宣言

```
typedef struct RBLE_LE_CHNL_MAP_t{
    uint8_t  map[RBLE_LE_CHNL_MAP_LEN];   チャネルマップ配列(5byte = 40ch/8bit)
                                          各ビットは 0:不使用、1:使用で設定
} RBLE_LE_CHNL_MAP;
```

3.3 GATT Definitions

- GATT アトリビュートタイプ UUID 定義

```
#define RBLE_DECL_PRIMARY_SERVICE     0x2800u     Primary Service Declaration
#define RBLE_DECL_SECONDARY_SERVICE   0x2801u     Secondary Service Declaration
#define RBLE_DECL_INCLUDE             0x2802u     Include Declaration
#define RBLE_DECL_CHARACTERISTIC      0x2803u     Characteristic Declaration
```

- 特性記述子 UUID 定義

```
#define RBLE_DESC_CHAR_EXT_PROPERTIES    0x2900u   Characteristic Extended
                                                   Properties
#define RBLE_DESC_CHAR_USER_DESCRIPTION  0x2901u   Characteristic User Description
#define RBLE_DESC_CLIENT_CHAR_CONF       0x2902u   Client Characteristic
                                                   Configuration
#define RBLE_DESC_SERVER_CHAR_CONF       0x2903u   Server Characteristic
                                                   Configuration
#define RBLE_DESC_CHAR_PRESENTATION_FMT  0x2904u   Characteristic Presentation
                                                   Format
#define RBLE_DESC_CHAR_AGGREGATE_FMT     0x2905u   Characteristic Aggregate Format
#define RBLE_DESC_VALID_RANGE            0x2906u   Valid Range
#define RBLE_DESC_EXT_REPORT_REFERENCE   0x2907u   External Report Reference
#define RBLE_DESC_REPORT_REFERENCE       0x2908u   Report Reference
```

- 特性 UUID 定義

```
#define RBLE_CHAR_GAP_DEVICE_NAME          0x2A00u    Device Name
#define RBLE_CHAR_GAP_APPEARANCE           0x2A01u    Appearance
#define RBLE_CHAR_GAP_PH_PRIV_FLAG         0x2A02u    Peripheral Privacy Flag
#define RBLE_CHAR_GAP_RECONN_ADDRESS       0x2A03u    Reconnection Address
#define RBLE_CHAR_GAP_PH_PREF_CONN_PARAM   0x2A04u    Peripheral Preferred
                                                      Connection Parameters
#define RBLE_CHAR_GATT_SERVICE_CHANGED     0x2A05u    Service Changed
#define RBLE_CHAR_ALERT_LEVEL              0x2A06u    Alert Level
#define RBLE_CHAR_TX_POWER_LEVEL           0x2A07u    Tx Power Level
#define RBLE_CHAR_DATE_TIME                0x2A08u    Date Time
#define RBLE_CHAR_DAY_OF_WEEK              0x2A09u    Day of Week
#define RBLE_CHAR_DAY_DATE_TIME            0x2A0Au    Day Date Time
#define RBLE_CHAR_EXACT_TIME_256           0x2A0Cu    Exact Time 256
#define RBLE_CHAR_DST_OFFSET               0x2A0Du    DST Offset
#define RBLE_CHAR_TIME_ZONE                0x2A0Eu    Time Zone
#define RBLE_CHAR_LOCAL_TIME_INFO          0x2A0Fu    Local Time Information
#define RBLE_CHAR_TIME_WITH_DST            0x2A11u    Time with DST
#define RBLE_CHAR_TIME_ACCURACY            0x2A12u    Time Accuracy
#define RBLE_CHAR_TIME_SOURCE              0x2A13u    Time Source
#define RBLE_CHAR_REF_TIME_INFO            0x2A14u    Reference Time Information
#define RBLE_CHAR_TIME_UPDATE_CTRL_POINT   0x2A16u    Time Update Control Point
#define RBLE_CHAR_TIME_UPDATE_STATE        0x2A17u    Time Update State
```

```
#define RBLE_CHAR_GLUCOSE_MEASUREMENT     0x2A18u    Glucose Measurement
#define RBLE_CHAR_BATTERY_LEVEL           0x2A19u    Battery Level
#define RBLE_CHAR_TEMPERATURE_MEAS        0x2A1Cu    Temperature Measurement
#define RBLE_CHAR_TEMPERATURE_TYPE        0x2A1Du    Temperature Type
#define RBLE_CHAR_INTERMEDIATE_TEMP       0x2A1Eu    Intermediate Temperature
#define RBLE_CHAR_MEAS_INTERVAL           0x2A21u    Measurement Interval
#define RBLE_CHAR_BOOT_KB_INPUT_REPORT    0x2A22u    Boot Keyboard Input Report
#define RBLE_CHAR_SYSTEM_ID               0x2A23u    System ID
#define RBLE_CHAR_MODEL_NUMBER_STRING     0x2A24u    Model Number String
#define RBLE_CHAR_SERIAL_NUMBER_STRING    0x2A25u    Serial Number String
#define RBLE_CHAR_FW_REVISION_STRING      0x2A26u    Firmware Revision String
#define RBLE_CHAR_HW_REVISION_STRING      0x2A27u    Hardware Revision String
#define RBLE_CHAR_SW_REVISION_STRING      0x2A28u    Software Revision String
#define RBLE_CHAR_MANUF_NAME_STRING       0x2A29u    Manufacturer Name String
#define RBLE_CHAR_IEEE_CERTIF             0x2A2Au    IEEE 11073-20601 Regulatory
                                                     Certification Data List
#define RBLE_CHAR_CURRENT_TIME            0x2A2Bu    Current Time
#define RBLE_CHAR_SCAN_REFRESH            0x2A31u    Scan Refresh
#define RBLE_CHAR_BOOT_KB_OUTPUT_REPORT   0x2A32u    Boot Keyboard Output Report
#define RBLE_CHAR_BOOT_MOUSE_INPUT_REPORT 0x2A33u    Boot Mouse Input Report
#define RBLE_CHAR_GLUCOSE_MEAS_CONTEXT    0x2A34u    Glucose Measurement Context
#define RBLE_CHAR_BLOOD_PRESSURE_MEAS     0x2A35u    Blood Pressure Measurement
#define RBLE_CHAR_INTERMEDIATE_BLOOD_PRESS  0x2A36u  Intermediate Cuff Pressure
#define RBLE_CHAR_HEART_RATE_MEAS         0x2A37u    Heart Rate Measurement
#define RBLE_CHAR_BODY_SENSOR_LOCATION    0x2A38u    Body Sensor Location
#define RBLE_CHAR_HEART_RATE_CTRL_POINT   0x2A39u    Heart Rate Control Point
#define RBLE_CHAR_ALERT_STATUS            0x2A3Fu    Alert Status
#define RBLE_CHAR_RINGER_CTRL_POINT       0x2A40u    Ringer Control Point
#define RBLE_CHAR_RINGER_SETTING          0x2A41u    Ringer Setting
#define RBLE_CHAR_AL_CATEGORY_ID_BIT_MASK 0x2A42u    Alert Category ID Bit Mask
#define RBLE_CHAR_AL_CATEGORY_ID          0x2A43u    Alert Category ID
#define RBLE_CHAR_AL_NOTIF_CTRL_POINT     0x2A44u    Alert Notification Control
                                                     Point
#define RBLE_CHAR_UNREAD_ALERT_STATUS     0x2A45u    Unread Alert Status
#define RBLE_CHAR_NEW_ALERT               0x2A46u    New Alert
#define RBLE_CHAR_SUPP_NEW_AL_CATEGORY    0x2A47u    Supported New Alert Category
#define RBLE_CHAR_SUPP_UNREAD_AL_CATEGORY 0x2A48u    Supported Unread Alert
                                                     Category
#define RBLE_CHAR_BLOOD_PRESSURE_FEAT     0x2A49u    Blood Pressure Feature
#define RBLE_CHAR_HID_INFO                0x2A4Au    HID Information
#define RBLE_CHAR_REPORT_MAP              0x2A4Bu    Report Map
#define RBLE_CHAR_HID_CTRL_POINT          0x2A4Cu    HID Control Point
#define RBLE_CHAR_REPORT                  0x2A4Du    Report
#define RBLE_CHAR_PROTOCOL_MODE           0x2A4Eu    Protocol Mode
#define RBLE_CHAR_SCAN_INTERVAL_WINDOW    0x2A4Fu    Scan Interval Window
#define RBLE_CHAR_PNP_ID                  0x2A50u    PnP ID
```

```
#define RBLE_CHAR_GLUCOSE_FEATURE         0x2A51u    Glucose Feature
#define RBLE_CHAR_RECORD_ACCESS_CTRL_POINT  0x2A52u Record Access Control Point
#define RBLE_CHAR_SC_CNTL_POINT           0x2A53u    RSC Measurement
#define RBLE_CHAR_CSC_MEAS                0x2A54u    RSC Feature
#define RBLE_CHAR_SC_CNTL_POINT           0x2A55u    SC Control Point
#define RBLE_CHAR_CSC_MEAS                0x2A5Bu    CSC Measurement
#define RBLE_CHAR_CSC_FEATURE             0x2A5Cu    CSC Feature
#define RBLE_CHAR_SENSOR_LOCATION         0x2A5Du    Sensor Location
#define RBLE_CHAR_CYCLING_POWER_MEAS      0x2A63u    Cycling Power Measurements
#define RBLE_CHAR_CYCLING_POWER_VECTOR    0x2A64u    Cycling Power Vector
#define RBLE_CHAR_CYCLING_POWER_FEATURE   0x2A65u    Cycling Power Feature
#define RBLE_CHAR_CYCLING_POWER_CNTL_POINT  0x2A66u Cycling Power Control Point
#define RBLE_CHAR_LOCATION_SPEED          0x2A67u    Location and Speed
#define RBLE_CHAR_NAVIGATION              0x2A68u    Navigation
#define RBLE_CHAR_POSITION_QUALITY        0x2A69u    Position Quality
#define RBLE_CHAR_LN_FEATURE              0x2A6Au    LN Feature
#define RBLE_CHAR_LN_CNTL_POINT           0x2A6Bu    LN Control Point
```

- サービス UUID 定義

```
#define RBLE_SVC_GENERIC_ACCESS           0x1800u    Generic Access
#define RBLE_SVC_GENERIC_ATTRIBUTE        0x1801u    Generic Attribute
#define RBLE_SVC_IMMEDIATE_ALERT          0x1802u    Immediate Alert
#define RBLE_SVC_LINK_LOSS                0x1803u    Link Loss
#define RBLE_SVC_TX_POWER                 0x1804u    Tx Power
#define RBLE_SVC_CURRENT_TIME             0x1805u    Current Time Service
#define RBLE_SVC_REFERENCE_TIME_UPDATE    0x1806u    Reference Time Update Service
#define RBLE_SVC_NEXT_DST_CHANGE          0x1807u    Next DST Change Service
#define RBLE_SVC_GLUCOSE                  0x1808u    Glucose
#define RBLE_SVC_HEALTH_THERMOMETER       0x1809u    Health Thermometer
#define RBLE_SVC_DEVICE_INFORMATION       0x180Au    Device Information
#define RBLE_SVC_HEART_RATE               0x180Du    Heart Rate
#define RBLE_SVC_PHONE_ALERT_STATUS       0x180Eu    Phone Alert Status Service
#define RBLE_SVC_BATTERY_SERVICE          0x180Fu    Battery Service
#define RBLE_SVC_BLOOD_PRESSURE           0x1810u    Blood Pressure
#define RBLE_SVC_ALERT_NOTIFICATION       0x1811u    Alert Notification Service
#define RBLE_SVC_HUMAN_INTERFACE_DEVICE   0x1812u    Human Interface Device
#define RBLE_SVC_SCAN_PARAMETERS          0x1813u    Scan Parameters
#define RBLE_SVC_RUNNING_SPEED            0x1814u    Running Speed and Cadence
#define RBLE_SVC_CYCLING_SPEED            0x1816u    Cycling Speed and Cadence
#define RBLE_SVC_CYCLING_POWER            0x1818u    Cycling Power
#define RBLE_SVC_LOCATION_NAVIGATION      0x1819u    Location and Navigation
```

4. Initialization

このセクションは、rBLE の初期化機能に関連する API について記載します。

4.1 Definitions

このセクションは、rBLE の初期化機能 API で使用される定義について記載します。

- rBLE モード変化通知用コールバック関数型宣言

```
typedef void ( *RBLE_INIT_CB )( RBLE_MODE mode )
```

- rBLE モード列挙型宣言

```
enum RBLE_MODE_enum {
   RBLE_MODE_INITIALIZE        = 0,            rBLE 初期化処理中
   RBLE_MODE_ACTIVE,                           rBLE がアクティブ(動作可能)状態
   RBLE_MODE_RESET,                            rBLE が Reset 処理中
   RBLE_MODE_ERROR                             rBLE 初期化処理にてエラー発生
};
```

- rBLE モード型宣言

```
typedef uint8_t             RBLE_MODE;
```

4.2 Functions

以下に、rBLE の初期化機能で定義されている API 関数を表にまとめ、次節よりその API 関数の詳細について説明します。

表 4-1 rBLE 初期化機能 API 関数一覧

RBLE_Init	rBLE の初期化

4.2.1 RBLE_Init

RBLE_STATUS RBLE_Init (RBLE_INIT_CB call_back)		
このファンクションは、BLE ソフトウェアの初期化を行います。全ての rBLE 機能をご使用になる前に呼び出す必要があります。 Embedded 構成や Modem 構成の BLE_MCU では、rBLE の GAP・SM・GATT・VS のメモリ初期化を行います。 Modem 構成の APP_MCU では、rBLE_Host の管理するメモリの初期化、RSCIP の初期化、UART ドライバの初期化を行い、BLE_MCU とリンクを確立します。(「Bluetooth Low Energy プロトコルスタック rBLE コマンド仕様書」(R01AN1376)「4.6 リンク確立」を参照ください) rBLE モード変化通知用コールバック RBLE_INIT_CB にてアクティブ状態（動作可能）への状態変化が通知されることで、初期化処理が正常に終了したことを示します。		
Parameters:		
	call_back	rBLE ソフトウェアのモード変化通知用コールバックファンクションを指定
Return:		
	RBLE_OK	正常終了
	RBLE_ERR	初期化処理でエラー発生
	RBLE_PARAM_ERR	パラメータ異常

4.3 Events

以下に、rBLE の初期化機能で定義されているイベントを表にまとめ、次節よりそのイベントの詳細について説明します。

表 4-2 rBLE 初期化機能イベント一覧

RBLE_INIT_EVENT_MODE_CHANGE	rBLE の状態変化通知

4.3.1 RBLE_INIT_EVENT_MODE_CHANGE

void (*RBLE_INIT_CB)(RBLE_MODE mode)			
このコールバックファンクションは、rBLE のモード変化を通知します。			
Parameters:			
	mode	RBLE_MODE_INITIALIZE	rBLE ソフトウェアの初期化中を意味します。RBLE_Init()関数コールにより RBLE_MODE_INITIALIZE に移行します。初期化が完了すると、RBLE_MODE_ACTIVE に移行します。
		RBLE_MODE_ACTIVE	rBLE ソフトウェアがアクティブ状態（動作可能）であることを意味します。
		RBLE_MODE_RESET	RSCIP 接続がリセットされ、そのリセット処理中であることを意味します。リセットが完了すると、RBLE_MODE_ACTIVE に移行します。
		RBLE_MODE_ERROR	rBLE の初期化処理でエラーが発生したことを示します。
Return:			
	none		

5. Generic Access Profile

このセクションは、Bluetoothデバイスの検索・接続、ボンディング等の一般的なAPIについて記載します。

5.1 Definitions

このセクションは、Bluetooth デバイスの検索・接続、ボンディング等の一般的な API で使用される定義について記載します。

- GAP イベントタイプ列挙型宣言

```
enum RBLE_GAP_EVENT_TYPE_enum {
    RBLE_GAP_EVENT_RESET_RESULT = 1,          リセット完了イベント
                                              (Parameters : reset_result)
    RBLE_GAP_EVENT_SET_NAME_COMP,             デバイスネーム設定完了イベント
                                              (Parameters : status)
    RBLE_GAP_EVENT_OBSERVATION_ENABLE_COMP,   オブザービングの有効設定イベント
                                              (Parameters : status)
    RBLE_GAP_EVENT_OBSERVATION_DISABLE_COMP,  オブザービングの無効設定イベント
                                              (Parameters : status)
    RBLE_GAP_EVENT_BROADCAST_ENABLE_COMP,     ブロードキャストの有効設定イベント
                                              (Parameters : status)
    RBLE_GAP_EVENT_BROADCAST_DISABLE_COMP,    ブロードキャストの無効設定イベント
                                              (Parameters : status)
    RBLE_GAP_EVENT_SET_BONDING_MODE_COMP,     ボンディングモードの設定イベント
                                              (Parameters : status)
    RBLE_GAP_EVENT_SET_SECURITY_REQUEST_COMP, セキュリティモードの設定イベント
                                              (Parameters : set_sec_req)
    RBLE_GAP_EVENT_GET_DEVICE_INFO_COMP,      デバイス情報の取得完了イベント
                                              (Parameters : get_dev_ver)
    RBLE_GAP_EVENT_GET_WHITE_LIST_SIZE_COMP,  ローカルデバイスの White list サイズ
                                              読み出し完了イベント
                                              (Parameters : get_wlst_size)
    RBLE_GAP_EVENT_ADD_TO_WHITE_LIST_COMP,    White List デバイス追加完了イベント
                                              (Parameters : status)
    RBLE_GAP_EVENT_DEL_FROM_WHITE_LIST_COMP,  White List デバイス削除完了イベント
                                              (Parameters : status)
    RBLE_GAP_EVENT_GET_REMOTE_DEVICE_NAME_COMP,
                                              リモートデバイス名取得完了イベント
                                              (Parameters : get_remote_dev_name)
    RBLE_GAP_EVENT_GET_REMOTE_DEVICE_INFO_COMP,
                                              リモートデバイス情報取得完了イベント
                                              (Parameters : get_remote_dev_info)
    RBLE_GAP_EVENT_DEVICE_SEARCH_COMP,        デバイスサーチコマンド完了イベント

                                                 (Parameters : status)
    RBLE_GAP_EVENT_DEVICE_SEARCH_RESULT_IND,     デバイスサーチ結果通知イベント
```

```
                                                    (Parameters : dev_search_result)
    RBLE_GAP_EVENT_RPA_RESOLVED,
                                                    アドレス解決完了結果通知イベント
                                                (Parameters : rpa_resolved)
    RBLE_GAP_EVENT_SET_RANDOM_ADDRESS_COMP, ランダムアドレス設定コマンド完了イベント
                                                    (Parameters : set_rand_adr)
    RBLE_GAP_EVENT_SET_PRIVACY_FEATURE_COMP,プライバシーフィーチャー設定完了イベント
                                                    (Parameters : status)
    RBLE_GAP_EVENT_CONNECTION_COMP,             LE リンク確立イベント
                                                    (Parameters : conn_comp)
    RBLE_GAP_EVENT_CONNECTION_CANCEL_COMP,      LE リンク確立キャンセル完了イベント
                                                    (Parameters : status)
    RBLE_GAP_EVENT_DISCONNECT_COMP,             LE リンク切断完了イベント
                                                    (Parameters : disconnect)
    RBLE_GAP_EVENT_ADVERTISING_REPORT_IND,      アドバタイジングレポートおよび
                                                    データ通知イベント
                                                    (Parameters : adv_report)
    RBLE_GAP_EVENT_BONDING_COMP,                ボンディング完了イベント
                                                    (Parameters : bonding_comp)
    RBLE_GAP_EVENT_BONDING_REQ_IND,         ピアデバイスからのボンディング要求通知
                                                    イベント
                                                    (Parameters : bonding_req)
    RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_REQ_IND,
                                                    接続パラメータ変更要求通知イベント
                                                (Parameters : chg_connect_param_req)
    RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_COMP,接続パラメータ変更完了イベント
                                                    (Parameters : chg_connect_param)
    RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_RESPONSE,
                                                接続パラメータ変更要求応答通知イベント
                                            (Parameters : chg_connect_param_resp)
    RBLE_GAP_EVENT_CHANNEL_MAP_REQ_COMP,        チャネルマップ設定/取得完了イベント
                                                (Parameters : channel_map_req_cmp)
    RBLE_GAP_EVENT_READ_RSSI_COMP,              RSSI 取得完了イベント
                                                    (Parameters : read_rssi)
    RBLE_GAP_EVENT_WR_CHAR_IND,                 GAP 特性値書き込み通知イベント
                                                    (Parameters : wr_char)
    RBLE_GAP_EVENT_COMMAND_DISALLOWED_IND       GAP コマンド拒否通知イベント
                                                (Parameters : cmd_disallowed_ind)
};
```

- GAP イベントタイプ型宣言

```
typedef uint8_t                         RBLE_GAP_EVENT_TYPE;
```

- GAP イベントコールバック関数型宣言

```
typedef void ( *RBLE_GAP_EVENT_HANDLER )( RBLE_GAP_EVENT *event );
```

- GAP Observation および接続確立プロシージャ列挙型宣言

```
enum RBLE_GAP_OBSERV_MODE_enum {
    RBLE_GAP_OBSERVER           = 0x0800,       Observation プロシージャ
    RBLE_GAP_AUTO_CONNECT       = 0x1000,       Auto connection プロシージャ
    RBLE_GAP_SELECT_CONNECT     = 0x2000        Selective connection プロシージャ
};
```

- GAP Discovery モード列挙型宣言

```
enum RBLE_GAP_DISCOVERABLE_MODE_enum {
    RBLE_GAP_NON_DISCOVERABLE   = 0x0001,       Non-discoverable モード
    RBLE_GAP_GEN_DISCOVERABLE   = 0x0002,       General discoverable モード
    RBLE_GAP_LIM_DISCOVERABLE   = 0x0004        Limited discoverable モード
};
```

- GAP Bondable モード列挙型宣言

```
enum RBLE_GAP_BONDABLE_MODE_enum {
    RBLE_GAP_NON_BONDABLE       = 0x0100,       Non-bondable モード
    RBLE_GAP_BONDABLE           = 0x0200        Bondable モード
};
```

- GAP Broadcast モード列挙型宣言

```
enum RBLE_GAP_BROADCAST_MODE_enum {
    RBLE_GAP_BROADCASTER        = 0x0400        Broadcast モード
};
```

- GAP Connectable モード列挙型宣言

```
enum RBLE_GAP_CONNECTABLE_MODE_enum {
    RBLE_GAP_NON_CONNECTABLE     = 0x0010,        Non-connectable モード
    RBLE_GAP_UND_CONNECTABLE     = 0x0020,        Undirected connectable モード
    RBLE_GAP_DIR_CONNECTABLE     = 0x0040         Directed connectable モード
};
```

- GAP セキュリティモード列挙型宣言

```
enum RBLE_GAP_SECURITY_MODE_enum {
    RBLE_GAP_NO_SEC          = 0x00,       セキュリティモード 1　レベル 1
                                           (セキュリティ (認証・暗号化) なし)
    RBLE_GAP_SEC1_NOAUTH_PAIR_ENC,         セキュリティモード 1　レベル 2
                                           (Unauthenticated ペアリングによる暗号化)
    RBLE_GAP_SEC1_AUTH_PAIR_ENC,           セキュリティモード 1　レベル 3
                                           (Authenticated ペアリングによる暗号化)
    RBLE_GAP_SEC2_NOAUTH_DATA_SGN,         セキュリティモード 2　レベル 1
                                           (Unauthenticated ペアリングによるデータ署名)
    RBLE_GAP_SEC2_AUTH_DATA_SGN            セキュリティモード 2　レベル 2
                                           (Authentication ペアリングによるデータ署名)
};
```

- GAP Advertising タイプ列挙型宣言

```
enum RBLE_GAP_ADV_TYPE_enum {
    RBLE_GAP_ADV_CONN_UNDIR      = 0x00,   Connectable Undirected advertising
                                           (CONNECT_REQ または SCAN_REQ に応答可能)
    RBLE_GAP_ADV_CONN_DIR_HIGH_DUTY,       Connectable high duty cycle directed
                                           advertising
                                           (指定デバイスとのみ接続可能)
    RBLE_GAP_ADV_DISC_UNDIR,               Discoverable undirected advertising
                                           (SCAN_REQ に応答可能)
    RBLE_GAP_ADV_NONCONN_UNDIR,            Non-connectable undirected advertising
                                           (Advertiser からの情報送信のみ)
    RBLE_GAP_ADV_CONN_DIR_LOW_DUTY         Connectable low duty cycle directed
                                           advertising
                                           (指定デバイスとのみ接続可能)
};
```

- GAP イニシエータフィルタポリシー列挙型宣言

```
enum RBLE_GAP_INIT_FILTER_enum {
    RBLE_GAP_INIT_FILT_IGNORE_WLST     = 0x00,        White List を無視する
    RBLE_GAP_INIT_FILT_USE_WLST                       White List を使用する
};
```

- GAP Advertising チャネル列挙型宣言

```
enum RBLE_GAP_ADV_CH_enum {
    RBLE_ADV_CHANNEL_37                 = 0x01,     37ch を使用する
    RBLE_ADV_CHANNEL_38                 = 0x02,     38ch を使用する
    RBLE_ADV_CHANNEL_39                 = 0x04,     39ch を使用する
    RBLE_ADV_ALL_CHANNELS               = 0x07      全チャネル(37,38,39)を使用する
};
```

- GAP Advertising フィルタポリシー列挙型宣言

```
enum RBLE_GAP_ADV_FILTER_enum {
    RBLE_ADV_ALLOW_SCAN_ANY_CON_ANY     = 0x00,   SCAN_REQ：全て許可
                                                  CONNECT_REQ：全て許可
    RBLE_ADV_ALLOW_SCAN_WLST_CON_ANY,             SCAN_REQ：White List のみ許可
                                                  CONNECT_REQ：全て許可
    RBLE_ADV_ALLOW_SCAN_ANY_CON_WLST,             SCAN_REQ：全て許可
                                                  CONNECT_REQ：White List のみ許可
    RBLE_ADV_ALLOW_SCAN_WLST_CON_WLST             SCAN_REQ：White List のみ許可
                                                  CONNECT_REQ：White List のみ許可
};
```

- GAP アドレスタイプ列挙型宣言

```
enum RBLE_GAP_ADDR_TYPE_enum {
    RBLE_ADDR_PUBLIC                    = 0x00,   パブリックタイプ
    RBLE_ADDR_RAND                                ランダムタイプ
};
```

- GAP Scan タイプ列挙型宣言

```
enum RBLE_GAP_SCAN_TYPE_enum {
    RBLE_SCAN_PASSIVE                   = 0x00,   パッシブ Scan(受信するのみ)
    RBLE_SCAN_ACTIVE                              アクティブ Scan(SCAN_REQ を送信)
};
```

- GAP Scan フィルタポリシー列挙型宣言

```
enum RBLE_GAP_SCAN_FILTER_enum {
    RBLE_SCAN_ALLOW_ADV_ALL             = 0x00,   全ての Advertising パケットを受信
    RBLE_SCAN_ALLOW_ADV_WLST                      White List のデバイスからのみ受信
};
```

- GAP Scan 重複フィルタポリシー列挙型宣言

```
enum RBLE_GAP_SCAN_DUPLIC_enum {
    RBLE_SCAN_FILT_DUPLIC_DIS           = 0x00,   重複する受信データをフィルタリングしない
    RBLE_SCAN_FILT_DUPLIC_EN                      重複する受信データをフィルタリングする
};
```

- GAP プライバシー設定列挙型宣言

```
enum RBLE_GAP_PRIV_SETTING_enum {
    RBLE_DEVICE_PRIV_DISABLE          = 0x00,   プライバシー無効
    RBLE_CENTRAL_PRIV_ENABLE,                   Central プライバシー有効
    RBLE_PH_PRIV_ENABLE,                        Peripheral プライバシー有効
    RBLE_BCST_PRIV_ENABLE,                      Broadcaster プライバシー有効
    RBLE_OBSERV_PRIV_ENABLE,                    Observer プライバシー有効
    RBLE_OBSERV_PRIV_RESOLVE                    Observer アドレス解決
};
```

- GAP キー配布フラグ列挙型宣言

```
enum RBLE_GAP_KEY_DIST_enum {
    RBLE_KEY_DIST_NONE        = 0x00,     キーを配布しない
    RBLE_KEY_DIST_ENCKEY      = 0x01,     Encryption キーを配布
    RBLE_KEY_DIST_IDKEY       = 0x02,     IRK(Identity Resolving Key)を配布
    RBLE_KEY_DIST_SIGNKEY     = 0x04      CSRK(Connection Signature Resolving Key)
                                          を配布
};
```

- GAP OOB データフラグ列挙型宣言

```
enum RBLE_GAP_OOB_PRESENT_enum {
    RBLE_OOB_AUTH_DATA_NOT_PRESENT = 0x00,      OOB データなし
    RBLE_OOB_AUTH_DATA_FROM_REMOTE_DEV_PRESENT  リモートデバイスの OOB データ有り
};
```

- GAP 入出力能力列挙型宣言

```
enum RBLE_GAP_IO_CAP_enum {
    RBLE_IO_CAP_DISPLAY_ONLY       = 0x00,    入力：なし、       出力：ディスプレイ
    RBLE_IO_CAP_DISPLAY_YES_NO,               入力：Yes,No、    出力：ディスプレイ
    RBLE_IO_CAP_KB_ONLY,                      入力：キーボード、出力：なし
    RBLE_IO_CAP_NO_INPUT_NO_OUTPUT,           入力：なし、       出力：なし
    RBLE_IO_CAP_KB_DISPLAY                    入力：キーボード、出力：ディスプレイ
};
```

- 認証要件列挙型宣言

```
enum RBLE _AUTH_REQ_enum {
    RBLE_AUTH_REQ_NO_MITM_NO_BOND  = 0x00,      MITM から保護されない。Bonding しない。
    RBLE_AUTH_REQ_NO_MITM_BOND     = 0x01,      MITM から保護されない。Bonding する。
    RBLE_AUTH_REQ_MITM_NO_BOND     = 0x04,      MITM から保護される。Bonding しない。
    RBLE_AUTH_REQ_MITM_BOND        = 0x05       MITM から保護される。Bonding する。
};
```

- GAP デバイス検索列挙型宣言

```
enum RBLE_GAP_DISCOVERY_TYPE_enum {
    RBLE_GAP_GEN_DISCOVERY_TYPE    = 0x00,      一般的な検索(General および Limited
                                                Discoverable モードのデバイスを発見)
    RBLE_GAP_LIM_DISCOVERY_TYPE,                限定的な検索(Limited Discoverable
                                                モードのデバイスのみ発見)
    RBLE_GAP_CANCEL_DISCOVERY,                  デバイス検索停止
};
```

- GAP ボンディング情報列挙型宣言

```
enum RBLE_GAP_BOND_INFO_enum {
    RBLE_GAP_BOND_ADDED,                        ボンディング情報追加
    RBLE_GAP_BOND_REMOVED                       ボンディング情報削除
};
```

- GAP 特性値書き込み通知コード列挙型宣言

```
enum RBLE_GAP_WR_CHAR_CODE_enum {
    RBLE_GAP_WR_CHAR_NAME,                      デバイス名特性値書き込み
    RBLE_GAP_WR_CHAR_APPEARANCE                 アピアランス特性値書き込み
};
```

- クロック精度列挙型宣言

```
enum RBLE_SAC_CLOCK_ACCURACY_enum {
    RBLE_SCA_500PPM,                            クロック精度 500ppm
    RBLE_SCA_250PPM,                            クロック精度 250ppm
    RBLE_SCA_150PPM,                            クロック精度 150ppm
    RBLE_SCA_100PPM,                            クロック精度 100ppm
    RBLE_SCA_75PPM,                             クロック精度 75ppm
    RBLE_SCA_50PPM,                             クロック精度 50ppm
    RBLE_SCA_30PPM,                             クロック精度 30ppm
    RBLE_SCA_20PPM                              クロック精度 20ppm
};
```

- Advertising パラメータ構造体

```
typedef struct RBLE_SET_ADV_PARAM_t {
    uint16_t        adv_intv_min;               Advertising Minimum インターバル
    uint16_t        adv_intv_max;               Advertising Maximum インターバル
    uint8_t         adv_type;                   Advertising タイプ
    uint8_t         own_addr_type;              ローカルデバイスアドレスタイプ
    uint8_t         direct_addr_type;           ダイレクトアドレスタイプ
    RBLE_BD_ADDR    direct_addr;                ダイレクト接続 Bluetooth アドレス
    uint8_t         adv_chnl_map;               Advertising チャネルマップ
    uint8_t         adv_filt_policy;            Advertising フィルタポリシー
    uint8_t         reserved;                   予約
} RBLE_SET_ADV_PARAM;
```

- Advertising データ構造体

```
typedef struct RBLE_ADV_DATA_t {
    uint8_t       data[RBLE_ADV_DATA_LEN];      Advertising データ
} RBLE_ADV_DATA;
```

- Advertising データ設定構造体

```
typedef struct RBLE_SET_ADV_DATA_t {
    uint8_t           adv_data_len;             Advertising データ長
    RBLE_ADV_DATA     adv_data;                 Advertising データ
} RBLE_SET_ADV_DATA;
```

- Scan レスポンスデータ構造体

```
typedef struct RBLE_SCAN_RSP_DATA_t {
    uint8_t       data[RBLE_SCAN_RSP_DATA_LEN];     Scan レスポンスデータ
} RBLE_SCAN_RSP_DATA;
```

- Scan レスポンスデータ設定構造体

```
typedef struct RBLE_SET_SCAN_RSP_DATA_t {
    uint8_t             scan_rsp_data_len;          Scan レスポンスデータ長
    RBLE_SCAN_RSP_DATA  data;                       Scan レスポンスデータ
} RBLE_SET_SCAN_RSP_DATA;
```

- Advertising 情報構造体

```
typedef struct RBLE_ADV_INFO_t {
    RBLE_SET_ADV_PARAM        adv_param;        Advertising パラメータ
    RBLE_SET_ADV_DATA         adv_data;         Advertising データ
    RBLE_SET_SCAN_RSP_DATA    scan_rsp_data;    Scan レスポンスデータ
} RBLE_ADV_INFO;
```

- Scan パラメータ構造体

```
typedef struct RBLE_SET_SCAN_PARAMETER_t {
    uint8_t        scan_type;                  Scan タイプ
    uint8_t        reserved;                   予約
    uint16_t       scan_intv;                  Scan インターバル
    uint16_t       scan_window;                Scan ウィンドウ
    uint8_t        own_addr_type;              ローカルデバイスアドレスタイプ
    uint8_t        scan_filt_policy;           Scan フィルタポリシー
} RBLE_SET_SCAN_PARAMETER;
```

- Scan 情報構造体

```
typedef struct RBLE_SCANNING_INFO_t {
    RBLE_SET_SCAN_PARAMETER   set_scan;        Scan パラメータ
    uint8_t                   filter_dup;      重複フィルタリングポリシー
    uint8_t                   reserved;        予約
} RBLE_SCANNING_INFO;
```

- White List 追加・削除パラメータ構造体

```
typedef struct RBLE_DEV_ADDR_INFO_t {
    uint8_t             dev_addr_type;         デバイスアドレスタイプ
    RBLE_BD_ADDR        dev_addr;              デバイスアドレス
} RBLE_DEV_ADDR_INFO;
```

- 接続パラメータ構造体

```
typedef struct RBLE_CREATE_CONNECT_PARAM_t {
    uint16_t         scan_intv;                Scan インターバル
    uint16_t         scan_window;              Scan ウィンドウ
    uint8_t          init_filt_policy;         イニシエータフィルタポリシー
    uint8_t          peer_addr_type;           ピアデバイスアドレスタイプ
    RBLE_BD_ADDR     peer_addr;                ピアデバイスアドレス
    uint8_t          own_addr_type;            ローカルデバイスアドレスタイプ
    uint8_t          reserved;                 予約
    uint16_t         con_intv_min;             最小コネクションインターバル
    uint16_t         con_intv_max;             最大コネクションインターバル
    uint16_t         con_latency;              コネクションレイテンシー
    uint16_t         superv_to;                スーパービジョンタイムアウト
    uint16_t         ce_len_min;               最小コネクションイベントレングス
    uint16_t         ce_len_max;               最大コネクションイベントレングス
} RBLE_CREATE_CONNECT_PARAM;
```

- 接続完了パラメータ構造体

```
typedef struct RBLE_CONNECT_INFO_t {
    uint8_t          status;                接続確立結果
    uint8_t          role;                  ロール
    uint16_t         conhdl;                コネクションハンドル
    uint8_t          peer_addr_type;        ピアデバイスアドレスタイプ
    RBLE_BD_ADDR     peer_addr;             ピアデバイスアドレス
    uint8_t          idx;                   コネクションインデックス
    uint16_t         con_interval;          コネクションインターバル
    uint16_t         con_latency;           コネクションレイテンシー
    uint16_t         sup_to;                スーパービジョンタイムアウト
    uint8_t          clk_accuracy;          マスタクロック精度
    uint8_t          reserved3;             予約
}RBLE_CONNECT_INFO;
```

- Scan 有効・無効設定構造体

```
typedef struct RBLE_SET_SCAN_EN_t {
    uint8_t          scan_en;               Scan 有効・無効設定
    uint8_t          filter_duplic_en;      重複フィルタリング有効・無効設定
} RBLE_SET_SCAN_EN;
```

- ボンディングパラメータ構造体

```
typedef struct RBLE_BOND_PARAM_t {
    RBLE_BD_ADDR     addr;                  デバイスアドレス
    uint8_t          oob;                   OOB 情報
    uint8_t          iocap;                 I/O capabilities
    uint8_t          auth;                  認証要件
    uint8_t          key_size;              暗号化キーサイズ
    uint8_t          ikey_dist;             イニシエータ キー配布フラグ
    uint8_t          rkey_dist;             レスポンダ キー配布フラグ
} RBLE_BOND_PARAM;
```

- ボンディングレスポンスパラメータ構造体

```
typedef struct RBLE_BOND_RESP_PARAM_t {
    uint16_t        conhdl;          コネクションハンドル
    uint8_t         accept;          許可・拒否フラグ
    uint8_t         io_cap;          I/O capabilities
    uint8_t         oob;             OOB 情報
    uint8_t         auth_req;        認証要件
    uint8_t         max_key_size;    最大キーサイズ
    uint8_t         ikeys;           イニシエータ キー配布フラグ
    uint8_t         rkeys;           レスポンダ キー配布フラグ
    uint8_t         reserved;        予約
} RBLE_BOND_RESP_PARAM;
```

- コネクションアップデートパラメータ構造体

```
typedef struct RBLE_CONN_PARAM_t {
    uint16_t        intv_min;        最小コネクションインターバル
    uint16_t        intv_max;        最大コネクションインターバル
    uint16_t        latency;         コネクションレイテンシー
    uint16_t        time_out;        スーパービジョンタイムアウト
} RBLE_CONN_PARAM;
```

- デバイスバージョン情報構造体

```
typedef struct RBLE_DEVICE_VER_INFO_t {
    uint8_t         hci_ver;         HCI バージョン
    uint8_t         lmp_ver;         LMP バージョン
    uint8_t         host_ver;        Host バージョン
    uint8_t         reserved;        予約
    uint16_t        hci_subver;      HCI サブバージョン
    uint16_t        lmp_subver;      LMP サブバージョン
    uint16_t        host_subver;     Host サブバージョン
    uint16_t        company_id;      カンパニーID
} RBLE_DEVICE_VER_INFO;
```

- LE Features 構造体

```
typedef struct RBLE_FEATURES_t {
    uint8_t         feats[RBLE_LE_FEATS_LEN];    LE Features
} RBLE_FEATURES;
```

- Advertising レポート構造体

```
typedef struct RBLE_ADV_REPORT_t {
    uint8_t          evt_type;                    Advertising イベントタイプ
    uint8_t          adv_addr_type;               Advertising アドレスタイプ
    RBLE_BD_ADDR     adv_addr;                    Advertising デバイスアドレス
    uint8_t          data_len;                    Advertising データ長
    uint8_t          data[RBLE_ADV_DATA_LEN];     Advertising データ
    uint8_t          rssi;                        RSSI 値
} RBLE_ADV_REPORT;
```

- Advertising レポートイベント構造体

```
typedef struct RBLE_ADV_REPORT_EVT_t {
    RBLE_ADV_REPORT      adv_rep;                 Advertising レポート
} RBLE_ADV_REPORT_EVT;
```

- GAP イベントパラメータ構造体

```
typedef struct RBLE_GAP_EVENT_t {
    RBLE_GAP_EVENT_TYPE    type;                    GAP イベントタイプ
    uint8_t                reserved;                予約
    union Event_Parameter_u {
        Generic イベント
        RBLE_STATUS      status;                    ステータス

        リセット完了イベント
        struct RBLE_GAP_Reset_Result_t {
            RBLE_STATUS       status;               リセット結果
            uint8_t           rBLE_major_ver;       rBLE メジャーバージョン
            uint8_t           rBLE_minor_ver;       rBLE マイナーバージョン
        } reset_result;

        セキュリティモード設定完了イベント
        struct RBLE_GAP_Set_Security_Request_t{
            RBLE_STATUS       status;               ステータス
            uint8_t           sec;                  セキュリティモード
        } set_sec_req;

        デバイス情報取得完了イベント
        struct RBLE_GAP_Get_Device_Info_t {
            RBLE_STATUS           status;           ステータス
            RBLE_BD_ADDR          addr;             デバイスアドレス
            uint8_t               reserved;         予約
            RBLE_DEVICE_VER_INFO  ver_info;         バージョン情報
        } get_dev_ver;

            ローカルデバイスの White list サイズ読み出し完了イベント
        struct RBLE_GAP_Get_Wlst_size_t {
            RBLE_STATUS       status;               ステータス
            uint8_t           wlist_size;           White list サイズ
        } get_wlst_size;

        リモートデバイス名取得完了イベント
        struct RBLE_GAP_Get_Remote_Device_Name_t {
            RBLE_STATUS       status;               ステータス
            RBLE_BD_NAME      bd_name;              デバイス名称
            uint8_t           reserved;             予約
        } get_remote_dev_name;

        リモートデバイス情報取得完了イベント
        struct RBLE_GAP_GET_Remote_Device_Info_t {
            RBLE_STATUS       status;               ステータス
            uint8_t           reserved;             予約
```

```
    uint16_t          conhdl;                コネクションハンドル
    uint16_t          vers;                  LMP バージョン
    uint16_t          compid;                カンパニーID
    uint16_t          subvers;               LMP サブバージョン
    RBLE_FEATURES     feats_used;            LE Features
} get_remote_dev_info;
```

デバイスサーチ結果通知イベント

```
struct RBLE_GAP_Device_Search_Result_t {
    RBLE_ADV_REPORT   adv_resp;              Advertising レポート
} dev_search_result;
```

アドレス解決完了結果通知イベント

```
struct RBLE_GAP_RPA_Resolved_Evt_t {
    RBLE_BD_ADDR      res_addr;              解決済みデバイスアドレス
    uint8_t           res_addr_type;         解決済みアドレスタイプ
    RBLE_BD_ADDR      addr;                  以前のデバイスアドレス
    uint8_t           addr_type;             以前のアドレスタイプ
} rpa_resolved;
```

ランダムアドレス設定完了イベント

```
struct RBLE_GAP_Set_Random_Address_t {
    RBLE_STATUS       status;                ステータス
    RBLE_BD_ADDR      addr;                  デバイスアドレス
} set_rand_adr;
```

LE リンク接続完了イベント

```
struct RBLE_GAP_Connection_t {
    RBLE_CONNECT_INFO  connect_info;         接続完了パラメータ
} conn_comp;
```

LE リンク切断完了イベント

```
struct RBLE_GAP_Disconnect_t {
    uint8_t          reason;                 切断理由
    RBLE_STATUS      status;                 ステータス
    uint16_t         conhdl;                 コネクションハンドル
} disconnect;
```

Advertising レポート通知イベント

```
struct RBLE_GAP_Advertising_Report_t {
    RBLE_ADV_REPORT_EVT        evt;          Advertising イベント
    uint8_t                    reserved;     予約
} adv_report;
```

ボンディング完了イベント

```
struct RBLE_GAP_Bonding_Comp_t {
```

```
    uint16_t        conhdl;                 コネクションハンドル
    uint8_t         idx;                    コネクションインデックス
    RBLE_STATUS     status;                 ステータス
    uint8_t         key_size;               キーサイズ
    uint8_t         sec_prop;               セキュリティプロパティ
} bonding_comp;
```

ボンディング要求通知イベント

```
struct RBLE_GAP_Bonding_Req_t {
    RBLE_BD_ADDR    bd_addr;                デバイスアドレス
    uint8_t         index;                  コネクションインデックス
    uint8_t         auth_req;               認証要件
    uint8_t         io_cap;                 I/O Capability
    uint8_t         oob_data_flg;           OOB データフラグ
    uint8_t         max_enc_size;           最大キーサイズ
    uint8_t         ikey_dist;              イニシエータ キー配布フラグ
    uint8_t         rkey_dist;              レスポンダ キー配布フラグ
} bonding_req;
```

接続パラメータ変更要求通知イベント

```
struct RBLE_GAP_Change_Connection_Param_Req_Ind_t {
    uint16_t        conhdl;                 コネクションハンドル
    RBLE_CONN_PARAM conn_param;             コネクションパラメータ
} chg_connect_param_req;
```

接続パラメータ変更完了イベント

```
struct RBLE_GAP_Change_Connection_Param_t {
    RBLE_STATUS     status;                 ステータス
    uint8_t         reserved;               予約
    uint16_t        con_interval;           コネクションインターバル
    uint16_t        con_latency;            コネクションレイテンシー
    uint16_t        sup_to;                 スーパービジョンタイムアウト
} chg_connect_param;
```

接続パラメータ変更要求応答通知イベント

```
struct RBLE_GAP_Change_Connection_Param_Response_t {
    RBLE_STATUS     status;                 ステータス
    uint8_t         reserved;               予約
    uint16_t        result;                 変更結果
    uint16_t        conhdl;                 コネクションハンドル
} chg_connect_param_resp;
```

RSSI 取得完了イベント

```
struct RBLE_GAP_Read_RSSI_Cmp_Evt_t{
    uint16_t         conhdl;                コネクションハンドル
    RBLE_STATUS      status;                ステータス
    uint8_t          rssi;                  RSSI 値
} read_rssi;
```

GAP 特性値書き込み通知イベント

```
struct RBLE_GAP_Wr_Char_Ind_Evt_t{
    uint16_t         conhdl;                コネクションハンドル
    uint16_t         type;                  書き込まれた特性値
    union {
        RBLE_BD_NAME name;                  デバイス名特性値
        uint16_t     appearance;            アピアランス特性値
    } param;
} wr_char;
```

GAP コマンドエラー通知イベント

```
struct RBLE_GAP_Command_Error_Ind_t {
    RBLE_STATUS      status;                ステータス
    uint8_t          reserved;              予約
    uint16_t         opcode;                オペコード
} cmd_disallowed_ind;
  } param;
} RBLE_GAP_EVENT;
```

5.2 Function

rBLE の GAP 機能で定義されている API 関数を表 5-1 に纏めます。次節より、その API 関数の詳細について説明します。

表 5-1 GAP 機能 API 関数一覧

RBLE_GAP_Reset	GAP のリセットを行う
RBLE_GAP_Set_Name	ローカルデバイス名を設定する
RBLE_GAP_Observation_Enable	オブザービングの有効設定
RBLE_GAP_Observation_Disable	オブザービングの無効設定
RBLE_GAP_Broadcast_Enable	ブロードキャストの有効設定
RBLE_GAP_Broadcast_Disable	ブロードキャストの無効設定
RBLE_GAP_Set_Bonding_Mode	ボンディングモードの設定
RBLE_GAP_Set_Security_Request	セキュリティモードの設定
RBLE_GAP_Get_Device_Info	ローカルデバイス情報の取得を行う
RBLE_GAP_Get_White_List_Size	ローカルデバイスの White List サイズを読み出す
RBLE_GAP_Add_To_White_List	デバイスを White List へ追加する
RBLE_GAP_Del_From_White_List	White List からデバイスを削除する
RBLE_GAP_Get_Remote_Device_Name	リモートデバイス名を取得する
RBLE_GAP_Get_Remote_Device_Info	リモートデバイス情報を取得する
RBLE_GAP_Device_Search	リモートデバイスをサーチする
RBLE_GAP_Set_Random_Address	リンクレイヤーへランダムアドレス設定を行う
RBLE_GAP_Set_Privacy_Feature	プライバシーフィーチャーの設定を行う
RBLE_GAP_Create_Connection	LE リンク接続開始
RBLE_GAP_Connection_Cancel	LE リンク接続をキャンセル
RBLE_GAP_Disconnect	LE リンク切断
RBLE_GAP_Start_Bonding	ボンディング開始
RBLE_GAP_Bonding_Info_Ind	ボンディング情報通知
RBLE_GAP_Bonding_Response	ボンディング要求に対する応答
RBLE_GAP_Change_Connection_Param	リンクパラメータ変更
RBLE_GAP_Channel_Map_Req	チャネルマップの設定・取得
RBLE_GAP_Read_RSSI	RSSI 取得
RBLE_GAP_Authorized_Ind	承認通知

5.2.1 RBLE_GAP_Reset

RBLE_STATUS RBLE_GAP_Reset (RBLE_GAP_EVENT_HANDLER gap_call_back, RBLE_SM_EVENT_HANDLER sm_call_back)		
このファンクションは、GAP のリセットを行います。また、RWKE のメッセージやタイマで使用したヒープ領域を解放します。全ての Bluetooth 機能をご使用になる前に呼び出す必要があります。 ※再度本ファンクションを呼び出す場合、ユーザアプリケーションで ke_malloc を直接コールして確保したメモリは解放されませんので ke_free で解放した後に呼び出してください。 結果は GAP リセット完了イベント RBLE_GAP_EVENT_RESET_RESULT で通知されます。 ※このファンクションを呼び出す前に Bluetooth 機能をご使用(他ファンクションの呼び出し)した場合、イベントは通知されません。またこの時の動作は保証しません。		
Parameters:		
gap_call_back	GAP のイベントを通知するコールバックファンクションを指定	
sm_call_back	SM のイベントを通知するコールバックファンクションを指定	
Return:		
RBLE_OK	正常終了	
RBLE_PARAM_ERR	パラメータ異常	
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可	

5.2.2 RBLE_GAP_Set_Name

RBLE_STATUS RBLE_GAP_Set_Name(RBLE_BD_NAME *dev_name)		
このファンクションは、GAP Device Name Characteristic にローカルデバイスのデバイス名称を設定します。デバイス名称として設定可能な最大文字列サイズは 64byte 長です。結果はデバイス名称設定完了イベント RBLE_GAP_EVENT_SET_NAME_COMP で通知されます。 ※このファンクションで設定したデバイス名称は、電源 ON の間、次に GAP をリセット（RBLE_GAP_Reset）するまで保持されます。		
Parameters:		
*dev_name	*namelen*	デバイス名称のデータ長
	name	デバイス名称データ
Return:		
RBLE _OK	正常終了	
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可	

5.2.3 RBLE_GAP_Observation_Enable

RBLE_STATUS RBLE_GAP_Observation_Enable(uint16_t mode, RBLE_SCANNING_INFO *set_scan)			
このファンクションは、オブザーベーションプロシージャまたは接続プロシージャを有効に設定します。結果はオブザービングの有効設定イベント RBLE_GAP_EVENT_OBSERVATION_ENABLE_COMP で通知されます。			
Parameters:			
mode	RBLE_GAP_OBSERVER		Observation プロシージャを実行 Observer として動作し、set_scan で指定されたパラメータで Scan を行います。 受信した Advetising Report は、RBLE_GAP_EVENT_ADVERTISING_REPORT_IND で通知されます。
	RBLE_GAP_AUTO_CONNECT		Auto connection プロシージャを実行 White List に登録されたデバイスに接続を行います。set_scan のパラメータではなく固定値を使用します。「Bluetooth Low Energy プロトコルスタック ユーザーズマニュアル」(R01UW0095)「6.1.11.2 GAP パラメータの設定 表 6-23 GAP パラメータ設定マクロ」を参照ください。 キャンセルする場合は RBLE_GAP_Connection_Cancel をコールします。
	RBLE_GAP_SELECT_CONNECT		Selective connection プロシージャを実行 White List を使用して SCAN し、見つかったデバイスに接続を行います。set_scan のパラメータではなく固定値を使用します。「Bluetooth Low Energy プロトコルスタック ユーザーズマニュアル」(R01UW0095)「6.1.11.2 GAP パラメータの設定 表 6-23 GAP パラメータ設定マクロ」を参照ください。 キャンセルする場合は RBLE_GAP_Observation_Disable をコールします。
set_scan	*scan_type*	RBLE_SCAN_PASSIVE	パッシブスキャン(受信するのみ)を実行
		RBLE_SCAN_ACTIVE	アクティブスキャン(SCAN_REQ を送信)を実行
	scan_intv	スキャンインターバル N=0x0004～0x4000 (Time=N×0.625msec(2.5msec～10.24sec))	
	scan_window	スキャンウィンドウサイズ N=0x0004～0x4000 (Time=N×0.625msec(2.5msec～10.24sec)) ※スキャンインターバル＞スキャンウィンドウサイズ	
	own_addr_type	RBLE_ADDR_PUBLIC	パブリック BD address
		RBLE_ADDR_RAND	ランダム BD Address
	scan_filt_policy	RBLE_SCAN_ALLOW_ADV_ALL	全ての Advertising パケットを受信
		RBLE_SCAN_ALLOW_ADV_WLST	White List のデバイスからのみ受信
	filter_dup	RBLE_SCAN_FILT_DUPLIC_DIS	重複する受信データをフィルタリングしない

			RBLE_SCAN_FILT_DUPLIC_EN	重複する受信データをフィルタリングする

Return:

RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

5.2.4 RBLE_GAP_Observation_Disable

RBLE_STATUS RBLE_GAP_Observation_Disable(void)

このファンクションは、RBLE_GAP_Observation_Enable 関数で有効設定されたモードを無効に設定します。結果はオブザービングの無効設定イベント RBLE_GAP_EVENT_OBSERVATION_DISABLE_COMP で通知されます。
RBLE_GAP_Observation_Enable で RBLE_GAP_OBSERVER または RBLE_GAP_SELECT_CONNECT を指定したときに本ファンクションが使用できます。

Parameters:

none

Return:

RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

5.2.5 RBLE_GAP_Broadcast_Enable

RBLE_STATUS RBLE_GAP_Broadcast_Enable(uint16_t disc_mode, uint16_t conn_mode, RBLE_ADV_INFO *adv_info)

このファンクションは、Discoverable モードおよび Connectable モードを設定します。
ビーコンなどのように Broadcaster として動作する場合や、接続中にアドバタイジングを行う場合は、下記パラメータのみ有効です。それ以外を指定するとイベントで RBLE_COMMAND_DISALLOWED が通知されます。

disc_mode = RBLE_GAP_BROADCASTER
conn_mode = 0
adv_type = RBLE_GAP_ADV_DISC_UNDIR または RBLE_GAP_ADV_NONCONN_UNDIR

結果はブロードキャストの有効設定イベント RBLE_GAP_EVENT_BROADCAST_ENABLE_COMP で通知されます。
※*disc_mode* に RBLE_GAP_LIM_DISCOVERABLE を設定した場合は、RBLE_GAP_Broadcast_Disable 関数を呼び出さないでください。
※*disc_mode* に RBLE_GAP_LIM_DISCOVERABLE または RBLE_GAP_GEN_DISCOVERABLE を設定した場合は、Advetising Data に AD Types の Flags を含めてください。

設定可能なパラメータの組み合わせは下記のとおり。

GAP で規定される Mode	disc_mode	conn_mode	adv_type
Broadcast	RBLE_GAP_BROADCASTER	0	RBLE_GAP_ADV_NONCONN_UNDIR or RBLE_GAP_ADV_DISC_UNDIR
Non-Discoverable Non-Connectable	RBLE_GAP_NON_DISCOVERABLE	RBLE_GAP_NON_CONNECTABLE	RBLE_GAP_ADV_NONCONN_UNDIR
Non-Discoverable Undirected Connectable	RBLE_GAP_NON_DISCOVERABLE	RBLE_GAP_UND_CONNECTABLE	RBLE_GAP_ADV_CONN_UNDIR
Limited Discoverable Undirected Connectable	RBLE_GAP_LIM_DISCOVERABLE	RBLE_GAP_UND_CONNECTABLE	RBLE_GAP_ADV_CONN_UNDIR
General Discoverable Undirected Connectable	RBLE_GAP_GEN_DISCOVERABLE	RBLE_GAP_UND_CONNECTABLE	RBLE_GAP_ADV_CONN_UNDIR
Directed Connectable	RBLE_GAP_NON_DISCOVERABLE 以外	RBLE_GAP_DIR_CONNECTABLE	RBLE_GAP_ADV_CONN_DIR_HIGH_DUTY or RBLE_GAP_ADV_CONN_DIR_LOW_DUTY

Parameters:

disc_mode	RBLE_GAP_NON_DISCOVERABLE	General または Limited Discovery プロシージャを実行するデバイスから発見されない
	RBLE_GAP_GEN_DISCOVERABLE	General Discovery をプロシージャ実行するデバイスから発見されることが可能
	RBLE_GAP_LIM_DISCOVERABLE	General または Limited Discovery プロシージャを実行するデバイスから限られた期間発見されることが可能 ※デフォルト設定では 30.72 秒後に Advertising

RBLE_STATUS RBLE_GAP_Broadcast_Enable(uint16_t disc_mode, uint16_t conn_mode, RBLE_ADV_INFO *adv_info)				
				は停止しイベントは発生しません。この値は GAP_LIM_ADV_TIMEOUT 定義で変更することができます
		RBLE_GAP_BROADCASTER		Advertising イベントにてデータをブロードキャスト
	conn_mode	0		Broadcaster として動作
		RBLE_GAP_NON_CONNECTABLE		接続の確立を許可しない
		RBLE_GAP_UND_CONNECTABLE		接続可能
		RBLE_GAP_DIR_CONNECTABLE		既知のデバイスからのみ接続可能
	**adv_info*	*adv_intv_min*		最小 Advertising インターバル N=0x0020～0x4000 (Time=N×0.625msec(20msec～10.24sec)) ※*adv_type* に RBLE_GAP_ADV_DISC_UNDIR または RBLE_GAP_ADV_NONCONN_UNDIR を選択する(Broadcaster として動作する)場合、0x00A0(100msec)未満は設定できません。
		adv_intv_max		最大 Advertising インターバル N=0x0020～0x4000 (Time=N×0.625msec(20msec～10.24sec)) ※*adv_type* に RBLE_GAP_ADV_DISC_UNDIR または RBLE_GAP_ADV_NONCONN_UNDIR を選択する(Broadcaster として動作する)場合、0x00A0(100msec)未満は設定できません。
		adv_type	RBLE_GAP_ADV_CONN_UNDIR	CONNECT_REQ または SCAN_REQ に応答可能
			RBLE_GAP_ADV_CONN_DIR_HIGH_DUTY	指定デバイスとのみ接続可能
			RBLE_GAP_ADV_DISC_UNDIR	SCAN_REQ に応答可能
			RBLE_GAP_ADV_NONCONN_UNDIR	Advertiser からの情報送信のみ
			RBLE_GAP_ADV_CONN_DIR_LOW_DUTY	指定デバイスとのみ接続可能
		own_addr_type		ローカルデバイスのアドレスタイプ パブリックアドレス：RBLE_ADDR_PUBLIC ランダムアドレス：RBLE_ADDR_RAND
		direct_addr_type		ダイレクト接続アドレスタイプ（Initiator Address Type） パブリックアドレス：RBLE_ADDR_PUBLIC ランダムアドレス：RBLE_ADDR_RAND
		direct_addr		ダイレクト接続アドレス（Initiator Address）
		adv_chnl_map	RBLE_ADV_CHANNEL_37	37ch を使用する
			RBLE_ADV_CHANNEL_38	38ch を使用する
			RBLE_ADV_CHANNEL_39	39ch を使用する
			RBLE_ADV_ALL_CHANNELS	全チャネルを使用する
		adv_filt_policy	RBLE_ADV_ALLOW_SCAN_ANY_CON_ANY	SCAN_REQ：全て許可 CONNECT_REQ：全て許可
			RBLE_ADV_ALLOW_SCAN_WLST_CON_ANY	SCAN_REQ：White List のみ許可 CONNECT_REQ：全て許可
			RBLE_ADV_ALLOW_SCAN	SCAN_REQ：全て許可

RBLE_STATUS RBLE_GAP_Broadcast_Enable(uint16_t disc_mode, uint16_t conn_mode, RBLE_ADV_INFO *adv_info)				
			_ANY_CON_WLST	CONNECT_REQ：White List のみ許可
			RBLE_ADV_ALLOW_SCAN _WLST_CON_WLST	SCAN_REQ : White List のみ許可 CONNECT_REQ：White List のみ許可
		adv_data_len	Advertising のデータ長	
		adv_data	Advertising データ 【注】Advertising データフォーマットにつきましては Bluetooth Low Energy プロトコルスタック ・ユーザーズマニュアルを参照ください。	
		scan_rsp_data_len	Scan レスポンスデータ長	
		data	Scan レスポンスデータ 【注】Scan レスポンスデータフォーマットにつきましては Bluetooth Low Energy プロトコルスタック ・ユーザーズマニュアルを参照ください。	
Return:				
	RBLE_OK	正常終了		
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可		

5.2.6 RBLE_GAP_Broadcast_Disable

RBLE_STATUS RBLE_GAP_Broadcast_Disable(void)		
このファンクションは、RBLE_GAP_Broadcast_Enable 関数で有効設定されたモードを無効に設定します。結果はブロードキャストの無効設定イベント RBLE_GAP_EVENT_BROADCAST_DISABLE_COMP で通知されます。 ※RBLE_GAP_Broadcast_Enable 関数で RBLE_GAP_LIM_DISCOVERABLE を設定した場合は、本ファンクションを呼び出さないでください。		
Parameters:		
	none	
Return:		
	RBLE_OK	正常終了
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

5.2.7 RBLE_GAP_Set_Bonding_Mode

RBLE_STATUS RBLE_GAP_Set_Bonding_Mode(uint16_t mode)		
このファンクションは、ボンディングモードを BLE スタックに設定します。本ファンクションの呼び出しがない場合は Non-Bondable モードとなります。結果はボンディングモードの設定イベント RBLE_GAP_EVENT_SET_BONDING_MODE_COMP で通知されます。		
Parameters:		
mode	RBLE_GAP_NON_BONDABLE	Non-bondable モード
	RBLE_GAP_BONDABLE	Bondable モード(Bond 情報を保存する機能を有する)
Return:		
RBLE _OK	正常終了	
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可	

5.2.8 RBLE_GAP_Set_Security_Request

RBLE_STATUS RBLE_GAP_Set_Security_Request(uint8_t sec)		
このファンクションは、自デバイスのセキュリティモード・レベルを BLE スタックに設定します。結果はセキュリティモードの設定イベント RBLE_GAP_EVENT_SET_SECURITY_REQUEST_COMP で通知されます。 設定されたセキュリティモード・レベルは、BLE スタック内部で以下の用途で使用されます。 • 接続完了時 セキュリティモード・レベルが RBLE_GAP_NO_SEC 以外の場合、接続したデバイスとの(前回までの)セキュリティ状態を確認するため、RBLE_SM_CHK_BD_ADDR_REQ または RBLE_SM_IRK_REQ_IND イベントを通知します。 • ペアリング時 セキュリティモード・レベルが RBLE_GAP_SEC1_AUTH_PAIR_ENC または RBLE_GAP_SEC2_AUTH_DATA_SGN で、ペアリング方法が JustWorks に決定した場合、認証要件を満たせないため Authentication Requirements エラー(RBLE_SM_PAIR_ERR_AUTH_REQUIREMENTS)にてペアリングを中断します。		
Parameters:		
sec	RBLE_GAP_NO_SEC	セキュリティなし
	RBLE_GAP_SEC1_NOAUTH_PAIR_ENC	Unauthenticated ペアリングによる暗号化
	RBLE_GAP_SEC1_AUTH_PAIR_ENC	Authenticated ペアリングによる暗号化
	RBLE_GAP_SEC2_NOAUTH_DATA_SGN	Unauthenticated ペアリングによるデータ署名
	RBLE_GAP_SEC2_AUTH_DATA_SGN	Authenticated ペアリングによるデータ署名
Return:		
RBLE _OK	正常終了	
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可	

5.2.9 RBLE_GAP_Get_Device_Info

RBLE_STATUS RBLE_GAP_Get_Device_Info(void)	
このファンクションは、ローカルデバイス情報(デバイスアドレス・BLE スタックバージョン)の取得を行います。結果はデバイス情報の取得完了イベント RBLE_GAP_EVENT_GET_DEVICE_INFO_COMP で通知されます。	
Parameters:	
none	
Return:	
RBLE _OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

5.2.10 RBLE_GAP_Get_White_List_Size

RBLE_STATUS RBLE_GAP_Get_White_List_Size(void)	
このファンクションは、ローカルデバイスの White list サイズを読み出します。結果はローカルデバイスの White list サイズ読み出し完了イベント RBLE_GAP_EVENT_GET_WHITE_LIST_SIZE_COMP で通知されます。	
Parameters:	
none	
Return:	
RBLE _OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

5.2.11 RBLE_GAP_Add_To_White_List

RBLE_STATUS RBLE_GAP_Add_To_White_List(RBLE_DEV_ADDR_INFO *dev_info)			
このファンクションは、ボンディング済みなどの指定した既知デバイスを White List へ追加します。結果は White List デバイス追加完了イベント RBLE_GAP_EVENT_ADD_TO_WHITE_LIST_COMP で通知されます。 ※本ファンクションは、White List を使用したアドバタイジング中、またはスキャン中、またはイニシエーティング中に使用することはできません。			
Parameters:			
**dev_info*	*dev_addr_type*	RBLE_ADDR_PUBLIC	パブリック BD Address
		RBLE_ADDR_RAND	ランダム BD Address
	dev_addr	White List へ追加するデバイスの BD Address	
Return:			
RBLE _OK		正常終了	
RBLE_STATUS_ERROR		rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可	

5.2.12 RBLE_GAP_Del_From_White_List

RBLE_STATUS RBLE_GAP_Del_From_White_List(bool all_dev, RBLE_DEV ADDR_INFO *dev_info)			
このファンクションは、指定デバイスを White List から削除します。結果は White List デバイス削除完了イベント RBLE_GAP_EVENT_DEL_FROM_WHITE_LIST_COMP で通知されます。 ※本ファンクションは、White List を使用したアドバタイジング中、またはスキャン中、またはイニシエーティング中に使用することはできません。			
Parameters:			
all_dev	White List から全デバイスを削除するフラグ(TRUE : 全削除、FALSE : 指定デバイスのみ削除) ※all_dev が TRUE の場合、以下のパラメータは無効です。		
**dev_info*	*dev_addr_type*	RBLE_ADDR_PUBLIC	パブリック BD Address
		RBLE_ADDR_RAND	ランダム BD Address
	dev_addr	White List から削除するデバイスの BD Address	
Return:			
RBLE _OK		正常終了	
RBLE_STATUS_ERROR		rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可	

5.2.13 RBLE_GAP_Get_Remote_Device_Name

RBLE_STATUS RBLE_GAP_Get_Remote_Device_Name(
RBLE_CREATE_CONNECT_PARAM *connect_param)

このファンクションは、指定リモートデバイスの名前を取得します。結果はリモートデバイス名取得完了イベント RBLE_GAP_EVENT_GET_REMOTE_DEVICE_NAME_COMP で通知されます。

※既に接続中の場合は、下記パラメータの *peer_addr* に接続中デバイスの BD アドレスを設定してください。

Parameters:

*connect_param	*scan_intv*	スキャンインターバル N=0x0004～0x4000 (Time=N×0.625msec(2.5msec～10.24sec))	
	scan_window	スキャンウィンドウサイズ N=0x0004～0x4000 (Time=N×0.625msec(2.5msec～10.24sec)) ※スキャンインターバル＞スキャンウィンドウサイズ	
	init_filt_policy	RBLE_GAP_INIT_FILT_IGNORE_WLST	White list を使用せず、*peer_addr_type, peer_addr* で指定されたデバイスと接続する
		RBLE_GAP_INIT_FILT_USE_WLST	White list を使用し、White list に登録されているデバイスと接続する。(*peer_addr_type, peer_addr* は無視される)
	peer_addr_type	ピアデバイスのデバイスアドレスタイプ パブリックアドレス：RBLE_ADDR_PUBLIC ランダムアドレス：RBLE_ADDR_RAND ※本パラメータ *init_filt_policy* が RBLE_GAP_INIT_FILT_IGNORE_WLST の時のみ有効です。	
	peer_addr	ピアデバイスのデバイスアドレス※*init_filt_policy* が RBLE_GAP_INIT_FILT_IGNORE_WLST の時のみ有効	
	own_addr_type	自デバイスのデバイスアドレスタイプ パブリックアドレス：RBLE_ADDR_PUBLIC ランダムアドレス：RBLE_ADDR_RAND	
	con_intv_min	最小コネクションインターバル N=0x0006～0x0C80 (Time=N×1.25msec(7.5msec～4.0sec))	
	con_intv_max	最大コネクションインターバル N=0x0006～0x0C80 (Time=N×1.25msec(7.5msec～4.0sec))	
	con_latency	コネクションスレーブレイテンシー(0x0000～0x01F3)	
	superv_to	スーパービジョンタイムアウト N=0x000A～0x0C80 (Time=N×10msec(100msec～32sec))	
	ce_len_min	最小コネクションイベントレングス(0x0000～0xFFFF) ※このパラメータは将来のために予約されており、現在は未使用です。	
	ce_len_max	最大コネクションイベントレングス(0x0000～0xFFFF) ※このパラメータは将来のために予約されており、現在は未使用です。	

Return:

RBLE _OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

5.2.14 RBLE_GAP_Get_Remote_Device_Info

RBLE_STATUS RBLE_GAP_Get_Remote_Device_Info(unit16_t conhdl)		
このファンクションは、指定リモートデバイスの情報(BLE スタックバージョン、LE サポート機能)を取得します。結果はリモートデバイス情報取得完了イベント RBLE_GAP_EVENT_GET_REMOTE_DEVICE_INFO_COMP で通知されます。		
Parameters:		
conhdl	コネクションハンドル	
Return:		
RBLE _OK	正常終了	
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可	

5.2.15 RBLE_GAP_Device_Search

RBLE_STATUS RBLE_GAP_Device_Search(uint8_t discovery_type, uint8_t addr_type)

このファンクションは、スキャンを実行し周辺デバイスを検索します。デバイス検索は 7.68 秒で自動的に終了します。下記パラメータにてスキャンを実行します。

Scan Type	Active scan
Scan Interval	11.25msec
Scan Window	11.25msec
Duplicate filtering	Enable

本パラメータは下記の定義で変更することができます。

- GAP_DEV_SEARCH_TIME : スキャン時間
- GAP_DEV_SEARCH_SCAN_INTV : スキャンインターバル
- GAP_DEV_SEARCH_SCAN_WINDOW : スキャンウインドウ

検索の完了はデバイスサーチコマンド完了イベント RBLE_GAP_EVENT_DEVICE_SEARCH_COMP で通知されます。

受信したアドバタイジングデータの Flags AD Type をチェックし、LE General Discoverable Mode Flag または LE Limited Discoverable Flag がセットされているデバイスを発見する毎にデバイスサーチ結果通知イベント RBLE_GAP_EVENT_DEVICE_SEARCH_RESULT_IND が通知されます。

Parameters:			
discovery_type	RBLE_GAP_GEN_DISCOVERY_TYPE		一般的な検索(General および Limited Discoverable モードのデバイスを発見)
	RBLE_GAP_LIM_DISCOVERY_TYPE		限定的な検索(Limited Discoverable モードのデバイスのみを発見)
	RBLE_GAP_CANCEL_DISCOVERY		デバイス検索をキャンセル
addr_type	RBLE_ADDR_PUBLIC	パブリック BD Address	
	RBLE_ADDR_RAND	ランダム BD Address	
Return:			
RBLE _OK	正常終了		
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可		

5.2.16 RBLE_GAP_Set_Random_Address

RBLE_STATUS RBLE_GAP_Set_Random_Address(RBLE_BD_ADDR *bd_addr)	
このファンクションは、自デバイスアドレスを指定のランダムアドレスに設定します。結果はランダムアドレス設定コマンド完了イベント RBLE_GAP_EVENT_SET_RANDOM_ADDRESS_COMP で通知されます。 ※このファンクションで設定したランダムアドレスは、電源 ON の間、次に GAP をリセット（RBLE_GAP_Reset）するまで保持されます。 設定したランダムアドレスは下記 API で使用できます。API の own_addr_type に RBLE_ADDR_RAND を設定してください。 • RBLE_GAP_Observation_Enable • RBLE_GAP_Broadcast_Enable • RBLE_GAP_Create_Connection	
Parameters:	
*bd_addr	設定するランダムアドレス
Return:	
RBLE _OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

5.2.17 RBLE_GAP_Set_Privacy_Feature

RBLE_STATUS RBLE_GAP_Set_Privacy_Feature(uint8_t priv_flag, uint8_t set_to_ll)

このファンクションは、自デバイスのプライバシーフィーチャーを設定します。
各ロールのプライバシーフィーチャーを有効にする場合、Resolvable Private Address(RPA)が生成されます。事前に RBLE_SM_Set_Key により IRK を設定してください。
プライバシーフィーチャーを有効にせず、アドレス解決手続きのみ有効にする場合は *priv_flag* に RBLE_OBSERV_PRIV_RESOLVE を指定してください。
結果はプライバシーフィーチャー設定完了イベント RBLE_GAP_EVENT_SET_PRIVACY_FEATURE_COMP で通知されます。
RPA を使用する場合、2 分 30 秒ごとに RPA が更新されます。更新ごとに RBLE_GAP_EVENT_SET_RANDOM_ADDRESS_COMP イベントが発生し RPA が通知されます。

RPA を更新する時間は下記定義で変更することができます。

- GAP_RESOLVBLE_PRIVATE_ADDR_INTV ：RPA 更新時間

Parameters:

priv_flag	RBLE_DEVICE_PRIV_DISABLE	プライバシー無効
	RBLE_CENTRAL_PRIV_ENABLE	Central プライバシー有効 (スキャン、または接続で RPA を使用)
	RBLE_PH_PRIV_ENABLE	Peripheral プライバシー有効 (アドバタイジングで RPA を使用)
	RBLE_BCST_PRIV_ENABLE	Broadcaster プライバシー有効 (アドバタイジングで RPA を使用)
	RBLE_OBSERV_PRIV_ENABLE	Observer プライバシー有効 (スキャンで RPA を使用)
	RBLE_OBSERV_PRIV_RESOLVE	アドレス解決手続き有効
set_to_ll	ランダムアドレスを生成し、生成したアドレスを Link Layer に設定するかどうかのフラグ(TRUE：Link Layer に設定する、FALSE：Link Layer に設定しない)	

Return:

RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

5.2.18 RBLE_GAP_Create_Connection

RBLE_STATUS RBLE_GAP_Create_Connection(RBLE_CREATE_CONNECT_PARAM *connect_param)			
このファンクションは、指定リモートデバイスとのリンクを確立します。結果は LE リンク確立イベント RBLE_GAP_EVENT_CONNECTION_COMP で通知されます。			
Parameters:			
*connect_param	*scan_intv*	スキャンインターバル N=0x0004～0x4000 (Time=N×0.625msec(2.5msec～10.24sec))	
	scan_window	スキャンウィンドウサイズ N=0x0004～0x4000 (Time=N×0.625msec(2.5msec～10.24sec)) ※スキャンインターバル＞スキャンウィンドウサイズ	
	init_filt_policy	RBLE_GAP_INIT_FILT_IGNORE_WLST	White list を使用せず、*peer_addr_type, peer_addr* で指定されたデバイスと接続する
		RBLE_GAP_INIT_FILT_USE_WLST	White list を使用し、White list に登録されているデバイスと接続する。(*peer_addr_type, peer_addr* は無視される)
	peer_addr_type	ピアデバイスのデバイスアドレスタイプ パブリックアドレス：RBLE_ADDR_PUBLIC ランダムアドレス：RBLE_ADDR_RAND ※このパラメータは *init_filt_policy* が RBLE_GAP_INIT_FILT_IGNORE_WLST の時のみ有効です。	
	peer_addr	ピアデバイスのデバイスアドレス ※このパラメータは *init_filt_policy* が RBLE_GAP_INIT_FILT_IGNORE_WLST の時のみ有効です。	
	own_addr_type	自デバイスのデバイスアドレスタイプ パブリックアドレス：RBLE_ADDR_PUBLIC ランダムアドレス：RBLE_ADDR_RAND	
	con_intv_min	最小コネクションインターバル N=0x0006～0x0C80 (Time=N×1.25msec(7.5msec～4.0sec))	
	con_intv_max	最大コネクションインターバル N=0x0006～0x0C80 (Time=N×1.25msec(7.5msec～4.0sec))	
	con_latency	コネクションスレーブレイテンシー(0x0000～0x01F3)	
	superv_to	スーパービジョンタイムアウト N=0x000A～0x0C80 (Time=N×10msec(100msec～32sec)) ※ Time > (1+*con_latency*)×*con_intv_max*×2 [msec] となるように指定してください。	
	ce_len_min	最小コネクションイベントレングス(0x0000～0xFFFF) ※このパラメータは将来のために予約されており、現在は未使用です。	
	ce_len_max	最大コネクションイベントレングス(0x0000～0xFFFF) ※このパラメータは将来のために予約されており、現在は未使用です。	

Return:	
RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

5.2.19 RBLE_GAP_Connection_Cancel

RBLE_STATUS RBLE_GAP_Connection_Cancel(void)		
このファンクションは、リモートデバイスとのリンクの確立要求をキャンセルします。結果は LE リンク確立キャンセル完了イベント RBLE_GAP_EVENT_CONNECTION_CANCEL_COMP が通知されます。続いてリンク確立結果イベント RBLE_GAP_EVENT_CONNECTION_COMP が status=UNKNOWN_CONNECTION_ID で通知されます。 イニシエーティング以外の状態で本ファンクションを呼び出した場合、イベントは発生しません。		
Parameters:		
	none	
Return:		
	RBLE_OK	正常終了
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

5.2.20 RBLE_GAP_Disconnect

RBLE_STATUS RBLE_GAP_Disconnect(uint16_t conhdl)		
このファンクションは、指定リモートデバイスとのリンクを切断します。結果は LE リンク切断完了イベント RBLE_GAP_EVENT_DISCONNECT_COMP で通知されます。		
Parameters:		
	conhdl	コネクションハンドル
Return:		
	RBLE_OK	正常終了
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

5.2.21 RBLE_GAP_Start_Bonding

RBLE_STATUS RBLE_GAP_Start_Bonding(RBLE_BOND_PARAM *bond_param)

このファンクションは、指定リモートデバイスとのボンディングを開始します。結果はボンディング完了イベント RBLE_GAP_EVENT_BONDING_COMP で通知されます。

- 自デバイスがマスタの時、リモートデバイスに Pairing Request コマンドを送信します。
- 自デバイスがスレーブの時、リモートデバイスに Security Request コマンドを送信します。

Parameters:

*bond_param	*addr*	ボンディングを開始するリモートデバイスの BD Adddress	
	oob	RBLE_OOB_AUTH_DATA_NOT_PRESENT	OOB データなし
		RBLE_OOB_AUTH_DATA_FROM_REMOTE_DEV_PRESENT	リモートデバイスの OOB データ有り
	iocap	RBLE_IO_CAP_DISPLAY_ONLY	入力：なし 出力：ディスプレイ
		RBLE_IO_CAP_DISPLAY_YES_NO	入力：Yes,No 出力：ディスプレイ
		RBLE_IO_CAP_KB_ONLY	入力：キーボード 出力：なし
		RBLE_IO_CAP_NO_INPUT_NO_OUTPUT	入力：なし 出力：なし
		RBLE_IO_CAP_KB_DISPLAY	入力：キーボード 出力：ディスプレイ
	auth	RBLE_AUTH_REQ_NO_MITM_NO_BOND	MITM から保護されない。 Bonding しない。
		RBLE_AUTH_REQ_NO_MITM_BOND	MITM から保護されない。 Bonding する。
		RBLE_AUTH_REQ_MITM_NO_BOND	MITM から保護される。 Bonding しない。
		RBLE_AUTH_REQ_MITM_BOND	MITM から保護される。 Bonding する。
	key_size	最大暗号化キーサイズ	
	ikey_dist	イニシエータが配布を行うキーの種類（以下から論理和で選択） RBLE_KEY_DIST_NONE ：キーを配布しない RBLE_KEY_DIST_ENCKEY ：LTK を配布 RBLE_KEY_DIST_IDKEY ：IRK を配布 RBLE_KEY_DIST_SIGNKEY ：CSRK を配布	
	rkey_dist	レスポンダが配布を行うキーの種類（以下から論理和で選択） RBLE_KEY_DIST_NONE ：キーを配布しない RBLE_KEY_DIST_ENCKEY ：LTK を配布 RBLE_KEY_DIST_IDKEY ：IRK を配布 RBLE_KEY_DIST_SIGNKEY ：CSRK を配布	

Return:

RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

5.2.22 RBLE_GAP_Bonding_Info_Ind

RBLE_STATUS RBLE_GAP_Bonding_Info_Ind(uint8_t bond_op, RBLE_BD_ADDR *addr)			
このファンクションは、GAP レイヤに対してボンディング情報の通知を行います。			
Parameters:			
	bond_op	RBLE_GAP_BOND_ADDED	ボンディング情報の追加
		RBLE_GAP_BOND_REMOVED	ボンディング情報の削除
	**addr*	追加または削除対象となるリモートデバイスの BD Address	
Return:			
	RBLE_OK	正常終了	
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可	

5.2.23 RBLE_GAP_Bonding_Response

RBLE_STATUS RBLE_GAP_Bonding_Response(RBLE_BOND_RESP_PARAM *res_bond_param)

このファンクションは、リモートデバイスからのボンディング要求 RBLE_GAP_EVENT_BONDING_REQ_IND イベントに応答します。結果はボンディング完了イベント RBLE_GAP_EVENT_BONDING_COMP で通知されます。

- 自デバイスがマスタの時、リモートデバイスに Pairing Request コマンドを送信します。
- 自デバイスがスレーブの時、リモートデバイスに Pairing Response コマンドを送信します。

Parameters:

*res_bond_param	*conhdl*	コネクションハンドル	
	accept	ボンディング要求応答フラグ RBLE_OK：許諾 RBLE_CONN_REJ_UNACCEPTABLE_BDADDR：拒否	
	iocap	RBLE_IO_CAP_DISPLAY_ONLY	入力：なし 出力：ディスプレイ
		RBLE_IO_CAP_DISPLAY_YES_NO	入力：Yes,No 出力：ディスプレイ
		RBLE_IO_CAP_KB_ONLY	入力：キーボード 出力：なし
		RBLE_IO_CAP_NO_INPUT_NO_OUTPUT	入力：なし 出力：なし
		RBLE_IO_CAP_KB_DISPLAY	入力：キーボード 出力：ディスプレイ
	oob	RBLE_OOB_AUTH_DATA_NOT_PRESENT	OOB データなし
		RBLE_OOB_AUTH_DATA_FROM_REMOTE_DEV_PRESENT	リモートデバイスの OOB データ有り
	auth_req	RBLE_AUTH_REQ_NO_MITM_NO_BOND	MITM から保護されない。 Bonding しない。
		RBLE_AUTH_REQ_NO_MITM_BOND	MITM から保護されない。 Bonding する。
		RBLE_AUTH_REQ_MITM_NO_BOND	MITM から保護される。 Bonding しない。
		RBLE_AUTH_REQ_MITM_BOND	MITM から保護される。 Bonding する。
	max_key_size	最大暗号化キーサイズ	
	ikeys	イニシエータが配布を行うキーの種類（以下から論理和で選択） RBLE_KEY_DIST_NONE ：キーを配布しない RBLE_KEY_DIST_ENCKEY ：LTK を配布 RBLE_KEY_DIST_IDKEY ：IRK を配布 RBLE_KEY_DIST_SIGNKEY ：CSRK を配布	
	rkeys	レスポンダが配布を行うキーの種類（以下から論理和で選択） RBLE_KEY_DIST_NONE ：キーを配布しない RBLE_KEY_DIST_ENCKEY ：LTK を配布 RBLE_KEY_DIST_IDKEY ：IRK を配布 RBLE_KEY_DIST_SIGNKEY ：CSRK を配布	

Return:

RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

5.2.24 RBLE_GAP_Change_Connection_Param

RBLE_STATUS RBLE_GAP_Change_Connection_Param(uint16_t conhdl, uint16_t result, RBLE_CONN_PARAM *conn_param, uint8_t role)			
このファンクションは、確立したリンクの接続パラメータを変更するための関数です。以下の場合に使用し、用途に応じて通知されるイベントが異なります。 1. マスタが接続パラメータを変更します。結果は接続パラメータ変更完了イベント RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_COMP で通知されます。 2. スレーブがマスタに対し接続パラメータの変更を要求します。マスターが要求を受け入れたかどうか、結果は接続パラメータ変更要求応答通知イベント RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_RESPONSE で通知されます。 通信中の接続パラメータが変更された場合、結果は接続パラメータ変更完了イベント RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_COMP で通知されます。 3. マスタがスレーブからの接続パラメータ変更要求に応答します。結果は接続パラメータ変更完了イベント RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_COMP で通知されます。			
Parameters:			
	conhdl	コネクションハンドル	
	result	接続パラメータ変更要求に対する応答(0x0000：許諾、0x0001：拒否) ※本パラメータは上記 3.の場合のみ有効です。	
	**conn_param*	intv_min	最小コネクションインターバル N=0x0006～0x0C80 (Time=N×1.25msec(7.5msec～4.0sec))
		intv_max	最大コネクションインターバル N=0x0006～0x0C80 (Time=N×1.25msec(7.5msec～4.0sec))
		latency	コネクションスレーブレイテンシー(0x0000～0x01F3)
		time_out	スーパービジョンタイムアウト N=0x000A～0x0C80 (Time=N×10msec(100msec～32sec))
	role	自デバイスのロール(RBLE_MASTER：マスタ、RBLE_SLAVE：スレーブ)	
Return:			
	RBLE_OK	正常終了	
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可	

5.2.25 RBLE_GAP_Channel_Map_Req

RBLE_STATUS RBLE_GAP_Channel_Map_Req(bool update_map, uint16_t conhdl, RBLE_LE_CHNL_MAP *chmap)		
このファンクションは、データチャネルマップの設定または取得を行います。結果はチャネルマップ設定/取得完了イベント RBLE_GAP_EVENT_CHANNEL_MAP_REQ_COMP で通知されます。 ※チャネルマップの設定はマスタ時のみ可能です。		
Parameters:		
	update_map	アップデートマップフラグ(TRUE：マップを設定する、FALSE：取得する)
	conhdl	コネクションハンドル ※本パラメータはチャネルマップ取得時のみ有効です。
	**chmap*	各データチャネル(0ch～36ch)の bad 状態を示す 37bit 値(0：bad、1：unknown) ※本パラメータはチャネルマップ設定時のみ有効です。
Return:		
	RBLE_OK	正常終了
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

5.2.26 RBLE_GAP_Read_RSSI

RBLE_STATUS RBLE_GAP_Read_RSSI(uint16_t conhdl)		
このファンクションは、指定リモートデバイスの RSSI の取得を行います。 結果は RSSI 取得完了イベント RBLE_GAP_EVENT_READ_RSSI_COMP で通知されます。 ※接続時のみ RSSI を取得することが可能です。		
Parameters:		
	conhdl	コネクションハンドル
Return:		
	RBLE_OK	正常終了
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

5.2.27 RBLE_GAP_Authorized_Ind

RBLE_STATUS RBLE_GAP_Authorized_Ind (uint16_t conhdl)		
このファンクションは、指定リモートデバイスとの接続をユーザが承認したことを BLE スタックに通知します。リモートデバイスとの接続に承認が必要であれば、接続完了時にユーザへの確認を行い、承認された場合に本ファンクションを呼び出してください。		
Parameters:		
	conhdl	コネクションハンドル
Return:		
	RBLE_OK	正常終了
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

5.3 Events

rBLE の GAP 機能で定義されているイベントを表 5-2 に纏めます。次節より、そのイベントの詳細について説明します。

表 5-2 GAP 機能イベント一覧

RBLE_GAP_EVENT_RESET_RESULT	リセット完了イベント
RBLE_GAP_EVENT_SET_NAME_COMP	デバイスネーム設定完了イベント
RBLE_GAP_EVENT_OBSERVATION_ENABLE_COMP	オブザービングの有効設定イベント
RBLE_GAP_EVENT_OBSERVATION_DISABLE_COMP	オブザービングの無効設定イベント
RBLE_GAP_EVENT_BROADCAST_ENABLE_COMP	ブロードキャストの有効設定イベント
RBLE_GAP_EVENT_BROADCAST_DISABLE_COMP	ブロードキャストの無効設定イベント
RBLE_GAP_EVENT_SET_BONDING_MODE_COMP	ボンディングモードの設定イベント
RBLE_GAP_EVENT_SET_SECURITY_REQUEST_COMP	セキュリティモードの設定イベント
RBLE_GAP_EVENT_GET_DEVICE_INFO_COMP	デバイス情報の取得完了イベント
RBLE_GAP_EVENT_GET_WHITE_LIST_SIZE_COMP	ローカルデバイスの White list サイズ読み出し完了イベント
RBLE_GAP_EVENT_ADD_TO_WHITE_LIST_COMP	White List デバイス追加完了イベント
RBLE_GAP_EVENT_DEL_FROM_WHITE_LIST_COMP	White List デバイス削除完了イベント
RBLE_GAP_EVENT_GET_REMOTE_DEVICE_INFO_COMP	リモートデバイス情報取得完了イベント
RBLE_GAP_EVENT_GET_REMOTE_DEVICE_NAME_COMP	リモートデバイス名取得完了イベント
RBLE_GAP_EVENT_DEVICE_SEARCH_COMP	デバイスサーチコマンド完了イベント
RBLE_GAP_EVENT_DEVICE_SEARCH_RESULT_IND	デバイスサーチ結果通知イベント
RBLE_GAP_EVENT_RPA_RESOLVED	アドレス解決完了結果通知イベント
RBLE_GAP_EVENT_SET_RANDOM_ADDRESS_COMP	ランダムアドレス設定コマンド完了イベント
RBLE_GAP_EVENT_SET_PRIVACY_FEATURE_COMP	プライバシーフィーチャー設定完了イベント
RBLE_GAP_EVENT_CONNECTION_COMP	LE リンク確立イベント
RBLE_GAP_EVENT_CONNECTION_CANCEL_COMP	LE リンク確立キャンセル完了イベント
RBLE_GAP_EVENT_DISCONNECT_COMP	LE リンク切断完了イベント
RBLE_GAP_EVENT_ADVERTISING_REPORT_IND	アドバタイジングレポートおよびデータ通知イベント
RBLE_GAP_EVENT_BONDING_COMP	ボンディング完了イベント
RBLE_GAP_EVENT_BONDING_REQ_IND	ピアデバイスからのボンディング要求通知イベント
RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_REQ_IND	接続パラメータ変更要求通知イベント
RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_COMP	接続パラメータ変更完了イベント
RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_RESPONSE	接続パラメータ変更要求応答通知イベント
RBLE_GAP_EVENT_CHANNEL_MAP_REQ_COMP	チャネルマップ設定/取得完了イベント
RBLE_GAP_EVENT_READ_RSSI_COMP	RSSI 取得完了イベント
RBLE_GAP_EVENT_WR_CHAR_IND	GAP 特性値書き込み通知イベント
RBLE_GAP_EVENT_COMMAND_DISALLOWED_IND	GAP コマンド拒否通知イベント

5.3.1 RBLE_GAP_EVENT_RESET_RESULT

RBLE_GAP_EVENT_RESET_RESULT	
このイベントは、GAP リセットの実行(RBLE_GAP_Reset)結果を通知します。	
Parameters:	
status	GAP リセットの実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
rBLE_major_ver	rBLE メジャーバージョン
rBLE_minor_ver	rBLE マイナーバージョン

5.3.2 RBLE_GAP_EVENT_SET_NAME_COMP

RBLE_GAP_EVENT_SET_NAME_COMP	
このイベントは、ローカルデバイス名称の設定処理(RBLE_GAP_Set_Name)実行結果を通知します。	
Parameters:	
status	ローカルデバイス名称の設定処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

5.3.3 RBLE_GAP_EVENT_OBSERVATION_ENABLE_COMP

RBLE_GAP_EVENT_OBSERVATION_ENABLE_COMP	
このイベントは、オブザービングの有効設定処理(RBLE_GAP_Observation_Enable)実行結果を通知します。	
Parameters:	
status	オブザービングの有効設定処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

5.3.4 RBLE_GAP_EVENT_OBSERVATION_DISABLE_COMP

RBLE_GAP_EVENT_OBSERVATION_DISABLE_COMP	
このイベントは、オブザービングの無効設定処理(RBLE_GAP_Observation_Disable)実行結果を通知します。	
Parameters:	
status	オブザービングの無効設定処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

5.3.5 RBLE_GAP_EVENT_BROADCAST_ENABLE_COMP

RBLE_GAP_EVENT_BROADCAST_ENABLE_COMP	
このイベントは、ブロードキャストの有効設定処理(RBLE_GAP_Broadcast_Enable)実行結果を通知します。	
Parameters:	
status	ブロードキャストの有効設定処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

5.3.6 RBLE_GAP_EVENT_BROADCAST_DISABLE_COMP

RBLE_GAP_EVENT_BROADCAST_DISABLE_COMP		
このイベントは、ブロードキャスト無効設定処理(RBLE_GAP_Broadcast_Disable)実行結果を通知します。		
Parameters:		
	status	ブロードキャスト無効設定処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

5.3.7 RBLE_GAP_EVENT_SET_BONDING_MODE_COMP

RBLE_GAP_EVENT_SET_BONDING_MODE_COMP		
このイベントは、ボンディングモードの設定処理(RBLE_GAP_Set_Bonding_Mode)実行結果を通知します。		
Parameters:		
	status	ボンディングモードの設定処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

5.3.8 RBLE_GAP_EVENT_SET_SECURITY_REQUEST_COMP

RBLE_GAP_EVENT_SET_SECURITY_REQUEST_COMP		
このイベントは、セキュリティモードの設定処理(RBLE_GAP_Set_Security_Request)実行結果を通知します。		
Parameters:		
	status	セキュリティモードの設定処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
	sec	現在のセキュリティモード

5.3.9 RBLE_GAP_EVENT_GET_DEVICE_INFO_COMP

RBLE_GAP_EVENT_GET_DEVICE_INFO_COMP			
このイベントは、ローカルデバイス情報の取得完了結果を通知します。			
Parameters:			
	status	ローカルデバイス情報の取得完了結果 (3.2 rBLE ステータス列挙型宣言を参照ください)	
	addr	ローカルデバイスの BD Address	
	ver_info	*hci_ver*	HCI バージョン
		lmp_ver	LMP バージョン
		host_ver	Host バージョン
		hci_subver	HCI サブバージョン
		lmp_subver	LMP サブバージョン
		host_subver	Host サブバージョン
		company_id	カンパニーID https://www.bluetooth.com/specifications/assigned-numbers/company-identifiers を参照ください

5.3.10 RBLE_GAP_EVENT_GET_WHITE_LIST_SIZE_COMP

RBLE_GAP_EVENT_GET_WHITE_LIST_SIZE_COMP	
このイベントは、ローカルデバイスの White List サイズ読み出し処理(RBLE_GAP_Get_White_List_Size)実行結果を通知します。	
Parameters:	
status	ローカルデバイスの White List サイズ読み出し処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
wlist_size	ローカルデバイスの White List サイズ ※White List サイズ読み出し処理エラー時、本パラメータは無効です。

5.3.11 RBLE_GAP_EVENT_ADD_TO_WHITE_LIST_COMP

RBLE_GAP_EVENT_ADD_TO_WHITE_LIST_COMP	
このイベントは、White List への指定デバイス追加処理(RBLE_GAP_Add_To_White_List)実行結果を通知します。	
Parameters:	
status	White List への指定デバイス追加処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

5.3.12 RBLE_GAP_EVENT_DEL_FROM_WHITE_LIST_COMP

RBLE_GAP_EVENT_DEL_FROM_WHITE_LIST_COMP	
このイベントは、White List から指定デバイス削除処理(RBLE_GAP_Del_From_White_List)実行を通知します。	
Parameters:	
status	White List から指定デバイス削除処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

5.3.13 RBLE_GAP_EVENT_GET_REMOTE_DEVICE_NAME_COMP

RBLE_GAP_EVENT_GET_REMOTE_DEVICE_NAME_COMP	
このイベントは、リモートデバイス名称取得処理(RBLE_GAP_Get_Remote_Device_Name)実行結果を通知します。	
Parameters:	
status	リモートデバイス名称取得処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
bd_name	リモートデバイス名称 ※リモートデバイス名称取得エラー時、本パラメータは無効です。

5.3.14 RBLE_GAP_EVENT_GET_REMOTE_DEVICE_INFO_COMP

RBLE_GAP_EVENT_GET_REMOTE_DEVICE_INFO_COMP	
このイベントは、リモートデバイス情報の取得処理(RBLE_GAP_Get_Remote_Device_Info)実行結果を通知します。	
Parameters:	
status	リモートデバイス情報の取得処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
conhdl	コネクションハンドル
vers	LMP バージョン ※リモートデバイス情報取得エラー時、本パラメータは無効です。
compid	カンパニーID ※リモートデバイス情報取得エラー時、本パラメータは無効です。
subvers	LMP サブバージョン ※リモートデバイス情報取得エラー時、本パラメータは無効です。
feats_used	リモートデバイスのサポートする LE フィーチャー bit0：LE Encryption(1：サポート、0：未サポート) その他の bit は将来のために予約されています。 ※リモートデバイス情報取得エラー時、本パラメータは無効です。 ※マスタ時のみ、本パラメータが有効です。

5.3.15 RBLE_GAP_EVENT_DEVICE_SEARCH_COMP

RBLE_GAP_EVENT_DEVICE_SEARCH_COMP	
このイベントは、周辺デバイスの検索処理 (RBLE_GAP_Device_Search)の完了を通知します。	
Parameters:	
status	周辺デバイスの検索処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

5.3.16 RBLE_GAP_EVENT_DEVICE_SEARCH_RESULT_IND

RBLE_GAP_EVENT_DEVICE_SEARCH_RESULT_IND			
このイベントは、周辺デバイスの検索結果を通知します。			
Parameters:			
	adv_resp	*evt_type*	Advertising イベントタイプ 0x00：Connectable undirected advertising 0x01：Connectable directed advertising 0x02：Scannable undirected advertising 0x03：Non connectable undirected advertising 0x04：Scan Response
		adv_addr_type	Advertiser のアドレスタイプ パブリックアドレス：RBLE_ADDR_PUBLIC ランダムアドレス：RBLE_ADDR_RAND
		adv_addr	Advertiser の BD Address
		data_len	Advertising データ長
		data[RBLE_ADV_DATA_LEN]	Advertising または Scan レスポンスデータ 【注】Advertising データおよび Scan レスポンスデータフォーマットにつきましては Bluetooth Low Energy プロトコルスタック ・ユーザーズマニュアルを参照ください。
		rssi	Advertising データ受信時の RSSI

5.3.17 RBLE_GAP_EVENT_RPA_RESOLVED

RBLE_GAP_EVENT_RPA_RESOLVED		
このイベントは、アドレス解決完了結果を通知します。		
Parameters:		
	res_addr	解決済みの BD Address (アプリケーションから渡された IRK を使用して解決できたアドレス)
	res_addr_type	解決済みのアドレスタイプ • パブリックアドレス：RBLE_ADDR_PUBLIC • ランダムアドレス：RBLE_ADDR_RAND
	addr	以前の BD Address (ペアリング時に IRK とともに保存されたアドレス)
	addr_type	以前のアドレスタイプ • パブリックアドレス：RBLE_ADDR_PUBLIC • ランダムアドレス：RBLE_ADDR_RAND

5.3.18 RBLE_GAP_EVENT_SET_RANDOM_ADDRESS_COMP

RBLE_GAP_EVENT_SET_RANDOM_ADDRESS_COMP		
このイベントは、ランダムアドレス設定処理(RBLE_GAP_Set_Random_Address)実行結果を通知します。		
Parameters:		
	status	ランダムアドレス設定処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
	addr	設定したランダムアドレス

5.3.19 RBLE_GAP_EVENT_SET_PRIVACY_FEATURE_COMP

RBLE_GAP_EVENT_SET_PRIVACY_FEATURE_COMP		
このイベントは、自デバイスのプライバシーフィーチャー設定処理(RBLE_GAP_Set_Privacy_Feature)実行結果を通知します。		
Parameters:		
	status	自デバイスのプライバシーフィーチャー設定処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

5.3.20 RBLE_GAP_EVENT_CONNECTION_COMP

RBLE_GAP_EVENT_CONNECTION_COMP			
このイベントは、LE リンクの確立結果を通知します。			
Parameters:			
	connect_info	*status*	LE リンクの確立結果 (3.2 rBLE ステータス列挙型宣言を参照ください) ※接続エラー時、以下のパラメータは無効です。
		role	自デバイスのロール (RBLE_MASTER：マスタ、RBLE_SLAVE：スレーブ)
		conhdl	コネクションハンドル
		peer_addr_type	ピアデバイスのアドレスタイプ パブリックアドレス：RBLE_ADDR_PUBLIC ランダムアドレス：RBLE_ADDR_RAND
		peer_addr	ピアデバイスの BD Address
		idx	コネクションインデックス
		con_interval	コネクションインターバル
		con_latency	スレーブレイテンシー
		sup_to	スーパービジョンタイムアウト
		clk_accuracy	マスタクロック精度 (5.1 クロック精度列挙型宣言を参照ください)

5.3.21 RBLE_GAP_EVENT_CONNECTION_CANCEL_COMP

RBLE_GAP_EVENT_CONNECTION_CANCEL_COMP		
このイベントは、LE リンク確立キャンセル処理(RBLE_GAP_Connection_Cancel)実行結果を通知します。		
Parameters:		
	status	LE リンク確立キャンセル処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

5.3.22 RBLE_GAP_EVENT_DISCONNECT_COMP

RBLE_GAP_EVENT_DISCONNECT_COMP	
このイベントは、LE リンク切断完了を通知します。	
Parameters:	
reason	切断理由 (3.2 rBLE ステータス列挙型宣言を参照ください)
status	切断結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
conhdl	コネクションハンドル

5.3.23 RBLE_GAP_EVENT_ADVERTISING_REPORT_IND

RBLE_GAP_EVENT_ADVERTISING_REPORT_IND			
このイベントは、アドバタイジングレポートを通知します。			
Parameters:			
evt	*adv_rep*	*evt_type*	Advertising イベントタイプ 0x00：Connectable undirected advertising 0x01：Connectable directed advertising 0x02：Scannable undirected advertising 0x03：Non connectable undirected advertising 0x04：Scan Response
		adv_addr_type	Advertiser のアドレスタイプ パブリックアドレス：RBLE_ADDR_PUBLIC ランダムアドレス：RBLE_ADDR_RAND
		adv_addr	Advertiser の BD Address
		data_len	Advertising データ長
		data[RBLE_ADV_DATA_LEN]	Advertising または Scan レスポンスデータ 【注】Advertising データおよび Scan レスポンスデータフォーマットにつきましては Bluetooth Low Energy プロトコルスタック ・ユーザーズマニュアルを参照ください。
		rssi	Advertising データ受信時の RSSI

5.3.24 RBLE_GAP_EVENT_BONDING_COMP

RBLE_GAP_EVENT_BONDING_COMP	
このイベントは、ボンディング完了を通知します。	
Parameters:	
conhdl	コネクションハンドル
idx	コネクションインデックス
status	ボンディング結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
key_size	暗号化キーサイズ
sec_prop	キーのセキュリティプロパティ RBLE_SMP_KSEC_NONE：セキュリティなし RBLE_SMP_KSEC_UNAUTH_NO_MITM：MITM から保護されない RBLE_SMP_KSEC_AUTH_MITM：MITM から保護される

5.3.25 RBLE_GAP_EVENT_BONDING_REQ_IND

RBLE_GAP_EVENT_BONDING_REQ_IND			
このイベントは、リモートデバイスからのボンディング要求を通知します。 リモートデバイスからのボンディング要求に対する応答関数 RBLE_GAP_Bonding_Response にて応答してください。 • 自デバイスがマスタの場合、スレーブからの Security Request コマンドで通知される RBLE_SM_LTK_REQ_FOR_ENC_IND に RBLE_SM_Ltk_Req_Resp にてエラー応答した場合に通知されます。この時、addr, index, auth_req 以外のパラメータは無効です。 • 自デバイスがスレーブの場合、マスタからの Pairing Request コマンドを受けた場合に通知されます。			
Parameters:			
bonding_req	*addr*	ボンディングを要求するリモートデバイスの BD Adddress	
	index	コネクションインデックス	
	auth_req	RBLE_AUTH_REQ_NO_MITM_NO_BOND	MITM から保護されない。 Bonding しない。
		RBLE_AUTH_REQ_NO_MITM_BOND	MITM から保護されない。 Bonding する。
		RBLE_AUTH_REQ_MITM_NO_BOND	MITM から保護される。 Bonding しない。
		RBLE_AUTH_REQ_MITM_BOND	MITM から保護される。 Bonding する。
	io_cap	RBLE_IO_CAP_DISPLAY_ONLY	入力：なし 出力：ディスプレイ
		RBLE_IO_CAP_DISPLAY_YES_NO	入力：Yes,No 出力：ディスプレイ
		RBLE_IO_CAP_KB_ONLY	入力：キーボード 出力：なし
		RBLE_IO_CAP_NO_INPUT_NO_OUTPUT	入力：なし 出力：なし
		RBLE_IO_CAP_KB_DISPLAY	入力：キーボード 出力：ディスプレイ
	oob_data_flg	RBLE_OOB_AUTH_DATA_NOT_PRESENT	OOB データなし
		RBLE_OOB_AUTH_DATA_FROM_REMOTE_DEV_PRESENT	リモートデバイスの OOB データ有り
	max_enc_size	最大暗号化キーサイズ	
	ikey_dist	イニシエータが配布を行うキーの種類（以下から論理和で選択） RBLE_KEY_DIST_NONE ：キーを配布しない RBLE_KEY_DIST_ENCKEY ：LTK を配布 RBLE_KEY_DIST_IDKEY ：IRK を配布 RBLE_KEY_DIST_SIGNKEY ：CSRK を配布	
	rkey_dist	レスポンダが配布を行うキーの種類（以下から論理和で選択） RBLE_KEY_DIST_NONE ：キーを配布しない RBLE_KEY_DIST_ENCKEY ：LTK を配布 RBLE_KEY_DIST_IDKEY ：IRK を配布 RBLE_KEY_DIST_SIGNKEY ：CSRK を配布	

5.3.26 RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_REQ_IND

RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_REQ_IND			
このイベントは、リモートペリフェラルデバイスからの接続パラメータの変更要求を通知します。 接続パラメータ変更関数 RBLE_GAP_Change_Connection_Param にて応答してください。			
Parameters:			
	conhdl	コネクションハンドル	
	conn_param	*intv_min*	最小コネクションインターバル
		intv_max	最大コネクションインターバル
		latency	コネクションスレーブレイテンシー
		time_out	スーパービジョンタイムアウト

5.3.27 RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_COMP

RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_COMP		
このイベントは、接続パラメータの変更結果を通知します。		
Parameters:		
	status	接続パラメータの変更結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
	con_interval	コネクションインターバル
	con_latency	コネクションスレーブレイテンシー
	sup_to	スーパービジョンタイムアウト

5.3.28 RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_RESPONSE

RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_RESPONSE		
このイベントは、接続パラメータの変更要求に対するマスタからの応答を通知します。		
Parameters:		
	status	接続パラメータの変更要求処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
	result	接続パラメータの変更要求結果 0x0000：接続パラメータ変更を許諾 0x0001：接続パラメータ変更を拒否
	conhdl	コネクションハンドル

5.3.29 RBLE_GAP_EVENT_CHANNEL_MAP_REQ_COMP

RBLE_GAP_EVENT_CHANNEL_MAP_REQ_COMP		
このイベントは、データチャネルマップの設定または取得結果を通知します。		
Parameters:		
	conhdl	コネクションハンドル
	status	データチャネルマップの設定または取得処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
	chmap	各データチャネル(0ch～36ch)の使用状態を示す 37bit 値(0：unused、1：used) ※本パラメータはチャネルマップ取得時のみ有効です

5.3.30 RBLE_GAP_EVENT_READ_RSSI_COMP

RBLE_GAP_EVENT_READ_RSSI_COMP		
このイベントは、指定リモートデバイスの RSSI 取得結果を通知します。		
Parameters:		
	conhdl	コネクションハンドル
	status	RSSI 取得結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
	rssi	指定リモートデバイスの RSSI 値 ※RSSI 取得エラーの場合、本パラメータは無効です。

5.3.31 RBLE_GAP_EVENT_WR_CHAR_IND

RBLE_GAP_EVENT_WR_CHAR_IND		
このイベントは、リモートデバイスからの GAP 特性値書き込みを通知します。 (GAP 特性値の書き込みは、GATT API を使用することで実行可能です。)		
Parameters:		
conhdl	コネクションハンドル	
type	書き込みが行われた GAP 特性値種別 • RBLE_GAP_WR_CHAR_NAME 下記パラメータの *name* にアクセスしてください。 • RBLE_GAP_WR_CHAR_APPEARANCE 下記パラメータの *appearance* にアクセスしてください。	
param	*name*	デバイス名特性値
	appearance	アピアランス特性値

5.3.32 RBLE_GAP_EVENT_COMMAND_DISALLOWED_IND

RBLE_GAP_EVENT_COMMAND_DISALLOWED_IND		
このイベントは、GAP コマンドが拒否されたことを通知します。		
Parameters:		
	status	コマンド実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
	opcode	拒否されたコマンドのオペコード

6. Security Manager

このセクションは、ペアリング・暗号化・データ署名等のセキュリティに関する API について記載します。

6.1 Definitions

このセクションは、ペアリング・暗号化・データ署名等のセキュリティに関する API で使用される定義について記載します。

- SM イベントタイプ列挙型宣言

```
enum RBLE_SM_EVENT_TYPE_enum {
    RBLE_SM_EVENT_SET_CNF = 1,              キー設定完了イベント
                                            (Parameters : set_conf)
    RBLE_SM_ENC_START_IND,                  暗号化開始通知イベント
                                            (Parameters : sec_start)
    RBLE_SM_TK_REQ_IND,                     TK 要求通知イベント
                                            (Parameters : tk_req)
    RBLE_SM_LTK_REQ_IND,                    キー配布のための LTK 要求通知イベント
                                            (Parameters : ltk_req)
    RBLE_SM_LTK_REQ_FOR_ENC_IND,            暗号化のための LTK 要求通知イベント
                                            (Parameters : ltk_req_for_enc)
    RBLE_SM_IRK_REQ_IND,                    IRK 要求通知イベント
                                            (Parameters : irk_req)
    RBLE_SM_CSRK_REQ_IND,                   CSRK 要求通知イベント
                                            (Parameters : csrk_req)
    RBLE_SM_KEY_IND,                        キー通知イベント
                                            (Parameters : key_ind)
    RBLE_SM_CHK_BD_ADDR_REQ,                BD アドレスチェック要求イベント
                                            (Parameters : chk_bdaddr)
    RBLE_SM_TIMEOUT_EVT,                    SM 処理タイムアウト通知イベント
                                            (Parameters : timeout_evt)
    RBLE_SM_EVENT_COMMAND_DISALLOWED_IND    SM コマンド拒否通知イベント
                                            (Parameters : cmd_disallowed_ind)
};
```

- SM イベントタイプ型宣言

```
typedef uint8_t                         RBLE_SM_EVENT_TYPE;
```

- SM イベントコールバック関数型宣言

```
typedef void ( *RBLE_SM_EVENT_HANDLER )( RBLE_SM_EVENT *event );
```

- キー配布フラグ列挙型宣言

```
enum RBLE_SMP_KEY_DIST_FLAG_enum {
    RBLE_SMP_KDIST_NONE          = 0x00,        キー配布なし
    RBLE_SMP_KDIST_ENCKEY        = 0x01,        LTK を配布
    RBLE_SMP_KDIST_IDKEY         = 0x02,        IRK を配布
    RBLE_SMP_KDIST_SIGNKEY       = 0x04         CSRK を配布
};
```

- 配布されたキーのセキュリティプロパティ列挙型宣言

```
enum RBLE_SMP_KSEC_enum {
    RBLE_SMP_KSEC_NONE             = 0x00,      セキュリティなし
    RBLE_SMP_KSEC_UNAUTH_NO_MITM,               中間者攻撃から保護されない
    RBLE_SMP_KSEC_AUTH_MITM                     中間者攻撃から保護される
};
```

- BD アドレスチェック要求応答列挙型宣言

```
enum RBLE_SMP_CHK_BD_REQ_RSP_enum {
    RBLE_SMP_SEC_NONE              = 0x00,      セキュリティなし
    RBLE_SMP_UNAUTHENTICATED       = 0x01,      Unauthenticated ペアリング済み
    RBLE_SMP_AUTHENTICATED         = 0x02,      Authenticated ペアリング済み
    RBLE_SMP_AUTHORIZED            = 0x04,      Authorize 済み
    RBLE_SMP_BONDED                = 0x08       Bond 済み
};
```

- セキュリティキー構造体宣言

```
typedef struct RBLE_KEY_VALUE_t{
    uint8_t   key[RBLE_KEY_LEN];                キー
}RBLE_KEY_VALUE;
```

- 乱数構造体宣言

```
typedef struct RBLE_RAND_NB_t{
    uint8_t   nb[RBLE_RAND_NB_LEN];             乱数(Rand)
}RBLE_RAND_NB;
```

- SM イベントパラメータ構造体

```
typedef struct RBLE_SM_EVENT_t {
    RBLE_SM_EVENT_TYPE  type;                   SM イベントタイプ
    uint8_t             reserved;               予約
    union Event_Parameter_u {
        キー設定完了イベント
        struct RBLE_EVT_SM_Set_Cnf_t{
            RBLE_STATUS  status;                ステータス
            uint8_t      key_code;              キー種別
        }set_conf;
```

暗号化開始通知イベント

```
struct RBLE_EVT_SM_Sec_Start_t{
    uint8_t       idx;                  コネクションインデックス
    RBLE_STATUS   status;               ステータス
    uint8_t       key_size;             キーサイズ
    uint8_t       sec_prop;             セキュリティプロパティ
    uint8_t       bonded;               ボンディング状態フラグ
    uint8_t       reserved;
}sec_start;
```

TK 要求通知イベント

```
struct RBLE_EVT_SM_Tk_Req_t{
    uint8_t       idx;                  コネクションインデックス
    uint8_t       oob_en;               OOB 有効フラグ
    uint8_t       disp_en;              TK 表示フラグ
}tk_req;
```

キー配布のための LTK 要求通知イベント

```
struct RBLE_EVT_SM_Ltk_Req_t{
    uint8_t       idx;                  コネクションインデックス
    uint8_t       auth_req;             認証要件
}ltk_req;
```

暗号化のための LTK 要求通知イベント

```
struct RBLE_EVT_SM_Ltk_Req_For_Enc_t{
    uint8_t       idx;                  コネクションインデックス
    uint8_t       auth_req;             認証要件
    uint16_t          ediv;             EDIV
    RBLE_RAND_NB      nb;               Rand
}ltk_req_for_enc;
```

IRK 要求通知イベント

```
struct RBLE_EVT_SM_Irk_Req_t{
    uint8_t       idx;                  コネクションインデックス
}irk_req;
```

CSRK 要求通知イベント

```
struct RBLE_EVT_SM_Csrk_Req_t{
    uint8_t       idx;                  コネクションインデックス
    RBLE_BD_ADDR addr;                  デバイスアドレス
    uint8_t       reserved;             予約
    uint32_t      signcnt;              サインカウンタ値
}csrk_req;
```

キー通知イベント

```
struct RBLE_EVT_SM_Key_t{
    uint8_t          idx;                    コネクションインデックス
    uint8_t          key_code;               キー種別
    uint16_t         ediv;                   EDIV
    RBLE_RAND_NB     nb;                     Rand
    RBLE_KEY_VALUE   ltk;                    キーの値
}key_ind;
```

BD アドレスチェック要求イベント

```
struct RBLE_EVT_SM_Chk_Bd_Addr_Req_t{
    uint8_t      idx;                        コネクションインデックス
    uint8_t      type;                       アドレスタイプ
    RBLE_BD_ADDR addr;                       デバイスアドレス
}chk_bdaddr;
```

SM 処理タイムアウト通知イベント

```
struct RBLE_EVT_SM_Timeout_Evt_t{
   uint8_t      idx;                        コネクションインデックス
}timeout_evt;
```

SM コマンド拒否通知イベント

```
struct RBLE_EVT_SM_Command_Disallowed_Ind_t{
    RBLE_STATUS  status;                     ステータス
    uint8_t      reserved;                   予約
    uint16_t     opcode;                     オペコード
}cmd_disallowed_ind;
  } param;
} RBLE_SM_EVENT;
```

6.2 Functions

rBLE の SM 機能で定義されている API 関数を表 6-1 に纏めます。次節より、その API 関数の詳細について説明します。

表 6-1 SM 機能 API 関数一覧

RBLE_SM_Set_Key	キーの設定
RBLE_SM_Start_Enc	Encryption 暗号化開始
RBLE_SM_Tk_Req_Resp	TK 要求に対する応答
RBLE_SM_Ltk_Req_Resp	LTK 要求に対する応答
RBLE_SM_Irk_Req_Resp	IRK 要求に対する応答
RBLE_SM_Csrk_Req_Resp	CSRK 要求に対する応答
RBLE_SM_Chk_Bd_Addr_Req_Resp	BD Address チェック要求に対する応答

6.2.1 RBLE_SM_Set_Key

RBLE_STATUS RBLE_SM_Set_Key(uint8_t Key_code, RBLE_KEY_VALUE *Key_Value)

このファンクションは、アプリケーションの保有するキーを SM に対して設定します。結果はキー設定完了イベント RBLE_SM_EVENT_SET_CNF で通知されます。また、設定した各キーは Pairing の鍵交換フェーズでリモートデバイスに渡されます。

- RPA を使う(プライバシーを有効にする)場合、事前に本ファンクションをコールして IRK を設定する必要があります。
- データ署名を使用する場合、事前に本ファンクションをコールして CSRK を設定する必要があります。

Parameters:

Key_code	RBLE_SMP_KDIST_IDKEY	IRK(Identity Resolving Key)を設定
	RBLE_SMP_KDIST_SIGNKEY	CSRK(Connection Signature Resolving Key)を設定
**Key_Value*	設定するキー格納先へのポインタ	

Return:

RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

6.2.2 RBLE_SM_Start_Enc

RBLE_STATUS RBLE_SM_Start_Enc(uint8_t idx, uint8_t auth_req, uint16_t ediv, RBLE_RAND_NB *rand_nb, RBLE_KEY_VALUE *ltk)		
このファンクションは、指定パラメータにてリモートデバイスとのリンクの暗号化を開始します。結果は暗号化開始通知イベント RBLE_SM_ENC_START_IND で通知されます。 • 自デバイスがマスタの時、リモートデバイスに Start Encryption コマンドを送信します。 • 自デバイスがスレーブの時、リモートデバイスに Security Request コマンドを送信します。		
Parameters:		
idx	コネクションインデックス	
auth_req	RBLE_AUTH_REQ_NO_MITM_NO_BOND	MITM から保護されない。 Bonding しない。
	RBLE_AUTH_REQ_NO_MITM_BOND	MITM から保護されない。 Bonding する。
	RBLE_AUTH_REQ_MITM_NO_BOND	MITM から保護される。 Bonding しない。
	RBLE_AUTH_REQ_MITM_BOND	MITM から保護される。 Bonding する。
ediv	EDIV	
**rand_nb*	Rand 格納先へのポインタ	
**ltk*	LTK 格納先へのポインタ	
Return:		
RBLE_OK	正常終了	
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可	

6.2.3 RBLE_SM_Tk_Req_Resp

RBLE_STATUS RBLE_SM_Tk_Req_Resp(uint8_t idx, uint8_t status, RBLE_KEY_VALUE *tk)	
このファンクションは、TK の要求 RBLE_SM_TK_REQ_IND イベントに応答します。 OOB の場合は、OOB で取得した TK を*tk に設定します。 Passkey の場合は、6 ケタの Passkey を*tk に MSB first で設定します。 (例 : Passkey = 123456 (0x1E240)の場合、tk= {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xE2, 0x40}となります。Passkey と無関係の領域は 0 クリアしなければペアリングに失敗します。)	
Parameters:	
idx	コネクションインデックス
status	TK の要求に対する応答 RBLE_OK : TK あり 上記以外 : TK なし ※status が RBLE_OK 以外の場合、以下のパラメータは無効です。
**tk*	TK 格納先へのポインタ
Return:	
RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

6.2.4 RBLE_SM_Ltk_Req_Resp

RBLE_STATUS RBLE_SM_Ltk_Req_Resp(uint8_t idx, uint8_t status, uint8_t sec_prop, uint16_t ediv, RBLE_RAND_NB *nb, RBLE_KEY_VALUE *ltk)	
このファンクションは、LTK の要求 RBLE_SM_LTK_REQ_IND イベントまたは RBLE_SM_LTK_REQ_FOR_ENC_IND イベントに応答します。 RBLE_SM_LTK_REQ_IND： Pairing の鍵交換フェーズの通知に応答する場合、EDIV / Rand / LTK をパラメータに設定します。sec_prop は使用されません。 RBLE_SM_LTK_REQ_FOR_ENC_IND： 暗号化開始要求に応答する場合、EDIV / Rand / LTK をパラメータに設定します。sec_prop は LTK が「どのようなペアリングで交換された鍵か」を意味します。以前の Pairing 完了時に RBLE_GAP_EVENT_BONDING_COMP イベントで通知された sec_prop と同値を設定する必要があります。	
Parameters:	
idx	コネクションインデックス
status	LTK の要求に対する応答 RBLE_OK：LTK あり 上記以外：LTK なし ※status が RBLE_OK 以外の場合、以下のパラメータは無効です。
sec_prop	LTK のセキュリティプロパティ RBLE_SMP_KSEC_NONE：セキュリティなし RBLE_SMP_KSEC_UNAUTH_NO_MITM：中間者攻撃から保護されない RBLE_SMP_KSEC_AUTH_MITM：中間者攻撃から保護される
ediv	EDIV
**nb*	Rand 格納先へのポインタ
**ltk*	LTK 格納先へのポインタ
Return:	
RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

6.2.5 RBLE_SM_Irk_Req_Resp

RBLE_STATUS RBLE_SM_Irk_Req_Resp(uint8_t idx, uint8_t status, RBLE_BD_ADDR *orig_addr, RBLE_KEY_VALUE *irk, uint8_t lk_sec_status)	
このファンクションは、アドレス解決を行うための IRK の要求 RBLE_SM_IRK_REQ_IND イベントに応答します。 指定した IRK でアドレス解決ができなかった場合、再度 RBLE_SM_IRK_REQ_IND イベントが通知されます。保持する IRK が無くなった場合は status に RBLE_OK 以外を設定します。 lk_sec_status は、リモートデバイスと「以前どのようなセキュリティが確立されたか」を意味し、セキュリティ状態を論理和で設定します。	
Parameters:	
idx	コネクションインデックス
status	IRK の要求に対する応答 RBLE_OK：IRK あり 上記以外：IRK なし ※status が RBLE_OK 以外の場合、以下のパラメータは無効です。
**orig_addr*	リモートデバイスのオリジナルの BD アドレス (ペアリング時に IRK と共に保存された BD アドレス)
**irk*	リモートデバイスの IRK
lk_sec_status	リモートデバイスとのセキュリティ状態（以下から論理和で選択） RBLE_SMP_SEC_NONE：セキュリティなし RBLE_SMP_UNAUTHENTICATED：Unauthenticated ペアリング済み RBLE_SMP_AUTHENTICATED：Authenticated ペアリング済み RBLE_SMP_AUTHORIZED：Authorize 済み RBLE_SMP_BONDED：Bond 済み
Return:	
RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

6.2.6 RBLE_SM_Csrk_Req_Resp

RBLE_STATUS RBLE_SM_Csrk_Req_Resp(uint8_t idx, uint8_t status, RBLE_KEY_VALUE *csrk, uint8_t lk_sec_status)	
このファンクションは、CSRK の要求 RBLE_SM_CSRK_REQ_IND イベントに応答します。 RBLE_SM_CSRK_REQ_IND イベントで通知されたサインカウンタの値が前回通知された値＋1 の場合に status に RBLE_OK を設定します。lk_sec_status は、リモートデバイスと「以前どのようなセキュリティが確立されたか」を意味し、セキュリティ状態を論理和で設定します。 サインカウンタが＋1 でない場合は、悪意のあるデバイスからのセキュリティ攻撃の可能性が、リンクを切断することを推奨します。 【参考】 BLUETOOTH SPECIFICATION Version 4.2 | Vol 3, Part C 10.4.2 Authenticate Signed Data Procedure "Hence, it is recommended that the server disconnect the link in case the client is a malicious device attempting to mount a security attack."	
Parameters:	
idx	コネクションインデックス
status	CSRK の要求に対する応答 RBLE_OK：CSRK あり 上記以外：CSRK なし ※status が RBLE_OK 以外の場合、以下のパラメータは無効です。
**csrk*	CSRK 格納先へのポインタ
lk_sec_status	リモートデバイスとのセキュリティ状態（以下から論理和で選択） RBLE_SMP_SEC_NONE：セキュリティなし RBLE_SMP_UNAUTHENTICATED：Unauthenticated ペアリング済み RBLE_SMP_AUTHENTICATED：Authenticated ペアリング済み RBLE_SMP_AUTHORIZED：Authorize 済み RBLE_SMP_BONDED：Bond 済み
Return:	
RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

6.2.7 RBLE_SM_Chk_Bd_Addr_Req_Resp

RBLE_STATUS RBLE_SM_Chk_Bd_Addr_Req_Resp (uint8_t idx, uint8_t type, uint8_t found_flag, uint8_t lk_sec_status, RBLE_BD_ADDR *addr)	
このファンクションは、BD アドレスのチェック要求 RBLE_SM_CHK_BD_ADDR_REQ イベントに応答します。RBLE_SM_CHK_BD_ADDR_REQ イベントで通知されたリモートデバイスが既知のデバイスである場合に、found_flag に TRUE を設定します。lk_sec_status は、リモートデバイスと「以前どのようなセキュリティが確立されたか」を意味し、セキュリティ状態を論理和で設定します。	
Parameters:	
idx	コネクションインデックス
type	リモートデバイスのデバイスアドレスタイプ パブリックアドレス：RBLE_ADDR_PUBLIC ランダムアドレス：RBLE_ADDR_RAND
found_flag	該当する BD アドレスの情報保有フラグ(TRUE：情報有り、FALSE：情報なし)
lk_sec_status	リモートデバイスとのセキュリティ状態（以下から論理和で選択） RBLE_SMP_SEC_NONE：セキュリティなし RBLE_SMP_UNAUTHENTICATED：Unauthenticated ペアリング済み RBLE_SMP_AUTHENTICATED：Authenticated ペアリング済み RBLE_SMP_AUTHORIZED：Authorize 済み RBLE_SMP_BONDED：Bond 済み
**addr*	リモートデバイスの BD アドレス
Return:	
RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

6.3 Events

rBLE の SM 機能で定義されているイベントを表 6-2 に纏めます。次節より、そのイベントの詳細について説明します。

表 6-2 SM機能イベント一覧

RBLE_SM_EVENT_SET_CNF	キー設定完了イベント
RBLE_SM_ENC_START_IND	暗号化開始通知イベント
RBLE_SM_TK_REQ_IND	TK 要求通知イベント
RBLE_SM_LTK_REQ_IND	キー配布のための LTK 要求通知イベント
RBLE_SM_LTK_REQ_FOR_ENC_IND	暗号化のための LTK 要求通知イベント
RBLE_SM_IRK_REQ_IND	IRK 要求通知イベント
RBLE_SM_CSRK_REQ_IND	CSRK 要求通知イベント
RBLE_SM_KEY_IND	キー通知イベント
RBLE_SM_CHK_BD_ADDR_REQ	BD アドレスチェック要求イベント
RBLE_SM_TIMEOUT_EVT	SM 処理タイムアウト通知イベント
RBLE_SM_EVENT_COMMAND_DISALLOWED_IND	SM コマンド拒否通知イベント

6.3.1 RBLE_SM_EVENT_SET_CNF

RBLE_SM_EVENT_SET_CNF	
このイベントは、指定キーの設定処理(RBLE_SM_Set_Key)実行結果を通知します。	
Parameters:	
status	キーの設定処理実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
key_code	設定したキー RBLE_SMP_KDIST_IDKEY：IRK RBLE_SMP_KDIST_SIGNKEY：CSRK

6.3.2 RBLE_SM_ENC_START_IND

RBLE_SM_ENC_START_IND	
このイベントは、リンクの暗号化の開始を通知します。	
Parameters:	
idx	コネクションインデックス
status	リンクの暗号化の開始結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
key_size	暗号化キーサイズ
sec_prop	キーのセキュリティプロパティ RBLE_SMP_KSEC_NONE：セキュリティなし RBLE_SMP_KSEC_UNAUTH_NO_MITM：中間者攻撃から保護されない RBLE_SMP_KSEC_AUTH_MITM：中間者攻撃から保護される
bonded	ボンディング状態 0：未ボンディング 1：ボンディング済み

6.3.3 RBLE_SM_TK_REQ_IND

RBLE_SM_TK_REQ_IND	
このイベントは、TK の要求を通知します。 TK の要求応答関数 RBLE_SM_Tk_Req_Resp にて応答してください。 ※OOB または Passkey Entry によるペアリング時に通知されます。JustWorks では本イベントは発生しません。	
Parameters:	
idx	コネクションインデックス
oob_en	OOB によるペアリング実行フラグ (TRUE：OOB ペアリングを実行、FALSE：OOB 以外のペアリング実行)
disp_en	TK をディスプレイ表示するかどうかのフラグ (TRUE：表示する、FALSE：表示しない)

6.3.4 RBLE_SM_LTK_REQ_IND

RBLE_SM_LTK_REQ_IND		
このイベントは、ペアリング中のキー交換フェーズでリモートデバイスに配布する LTK の要求を通知します。また、BLE スタック V1.11 以前では、暗号化セットアップ時に必要となる LTK の要求を通知します。 LTK の要求応答関数 RBLE_SM_Ltk_Req_Resp にて応答してください。 ※auth_req は、BLE スタック V1.11 以前のみ有効です。BLE スタック V1.20 以降では無視してください ※BLE スタック V1.11 以前で、暗号化セットアップ時に本イベントが通知された場合は、RBLE_SM_LTK_REQ_FOR_ENC_IND に記載される動作をします。		
Parameters:		
idx	コネクションインデックス	
auth_req	RBLE_AUTH_REQ_NO_MITM_NO_BOND	MITM から保護されない。 Bonding しない。
	RBLE_AUTH_REQ_NO_MITM_BOND	MITM から保護されない。 Bonding する。
	RBLE_AUTH_REQ_MITM_NO_BOND	MITM から保護される。 Bonding しない。
	RBLE_AUTH_REQ_MITM_BOND	MITM から保護される。 Bonding する。

6.3.5 RBLE_SM_LTK_REQ_FOR_ENC_IND

RBLE_SM_LTK_REQ_FOR_ENC_IND		
このイベントは、暗号化セットアップ時に必要となる LTK の要求を通知します。 LTK の要求応答関数 RBLE_SM_Ltk_Req_Resp にて応答してください。 • 自デバイスがマスタの場合、スレーブからの SM Security Request コマンド受信時に本イベントが通知されます。LTK を保持していない場合(RBLE_SM_Ltk_Req_Resp にてエラー応答した場合)、RBLE_GAP_EVENT_BONDING_REQ_IND イベントが通知されます。 • 自デバイスがスレーブの場合、マスタからの LL_ENC_REQ コマンド受信時に本イベントが通知されます。		
Parameters:		
idx	コネクションインデックス	
auth_req	認証要件 ※本パラメータは自デバイスがマスタ時のみ有効です。	
	RBLE_AUTH_REQ_NO_MITM_NO_BOND	MITM から保護されない。 Bonding しない。
	RBLE_AUTH_REQ_NO_MITM_BOND	MITM から保護されない。 Bonding する。
	RBLE_AUTH_REQ_MITM_NO_BOND	MITM から保護される。 Bonding しない。
	RBLE_AUTH_REQ_MITM_BOND	MITM から保護される。 Bonding する。
ediv	EDIV ※本パラメータは自デバイスがスレーブ時のみ有効です。	
nb	Rand ※本パラメータは自デバイスがスレーブ時のみ有効です。	

6.3.6 RBLE_SM_IRK_REQ_IND

RBLE_SM_IRK_REQ_IND	
このイベントは、リモートデバイスの IRK の要求を通知します。IRK の要求応答関数 RBLE_SM_Irk_Req_Resp にて応答してください。 セキュリティが有効(RBLE_GAP_Set_Security_Request で RBLE_GAP_NO_SEC 以外を設定)または、プライバシーが有効の場合に、アドバタイジング受信時または接続完了時に当該アドレスが RPA のとき本イベントが通知されます。	
Parameters:	
idx	コネクションインデックス

6.3.7 RBLE_SM_CSRK_REQ_IND

RBLE_SM_CSRK_REQ_IND	
このイベントは、CSRK の要求を通知します。CSRK の要求応答関数 RBLE_SM_Csrk_Req_Resp にて応答してください。 リモート GATT クライアントからの Signed Write 時に本イベントが通知されます。通知された BD アドレスのデバイスが CSRK を保持しており、かつサインカウンタがアプリケーションで管理する値＋1 の場合に、RBLE_SM_Csrk_Req_Resp にて応答します。	
Parameters:	
idx	コネクションインデックス
addr	リモートデバイスの BD アドレス
signcnt	受信データの署名に含まれていたサインカウンタ

6.3.8 RBLE_SM_KEY_IND

RBLE_SM_KEY_IND	
このイベントは、キー配布によって配布されたキーを通知します。	
Parameters:	
idx	コネクションインデックス
key_code	配布されたキー RBLE_SMP_KDIST_ENCKEY：LTK RBLE_SMP_KDIST_IDKEY：IRK RBLE_SMP_KDIST_SIGNKEY：CSRK
ediv	EDIV ※本パラメータは key_code が RBLE_SMP_KDIST_ENCKEY の場合のみ有効です。
nb	Rand ※本パラメータは key_code が RBLE_SMP_KDIST_ENCKEY の場合のみ有効です。
ltk	key_code で示されたキーの値

6.3.9 RBLE_SM_CHK_BD_ADDR_REQ

RBLE_SM_CHK_BD_ADDR_REQ	
このイベントは、BD アドレスのチェック要求を通知します。 BD アドレスのチェック要求応答関数 RBLE_SM_Chk_Bd_Addr_Req_Resp にて応答してください。 セキュリティが有効(RBLE_GAP_Set_Security_Request で RBLE_GAP_NO_SEC 以外を設定)または、プライバシーが有効の場合に、アドバタイジング受信時または接続完了時に当該アドレスが RPA 以外のとき本イベントが通知されます。	
Parameters:	
idx	コネクションインデックス
type	アドレスタイプ
addr	チェック対象の BD アドレス

6.3.10 RBLE_SM_TIMEOUT_EVT

RBLE_SM_TIMEOUT_EVT	
このイベントは、SM の処理にてタイムアウト(30 秒)が発生したことを通知します。 再度ペアリング等を実行する場合は、RBLE_GAP_Disconnect にてリモートデバイスとのリンクを切断し、再接続処理を行ってください。	
Parameters:	
idx	コネクションインデックス

6.3.11 RBLE_SM_EVENT_COMMAND_ERROR_IND

RBLE_SM_EVENT_COMMAND_DISALLOWED_IND	
このイベントは、SM コマンドが拒否されたことを通知します。	
Parameters:	
status	コマンド実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
opcode	拒否されたコマンドのオペコード

7. Generic Attribute Profile

このセクションは、Generic Attribute Profile(以下、GATT)の API について記載します。ローカル GATT サーバのデータベース構造につきましては、「Bluetooth Low Energy プロトコルスタック ユーザーズマニュアル」を参照下さい。

7.1 Definitions

このセクションは、GATT の API で使用される定義について記載します。

- GATT 定数定義

```
#define RBLE_GATT_MAX_VALUE              0x18       最大特性値サイズ
#define RBLE_GATT_MAX_HDL_LIST           0x08       最大ハンドルリスト数
#define RBLE_GATT_MAX_LONG_VALUE         0x48       最大 Long 特性値サイズ
#define RBLE_GATT_MAX_NB_HDLS            0x04       最大ハンドルペア数
#define RBLE_GATT_16BIT_UUID_OCTET       0x02       16-bit UUID オクテット
#define RBLE_GATT_32BIT_UUID_OCTET       0x04       32-bit UUID オクテット
#define RBLE_GATT_128BIT_UUID_OCTET      0x10       128-bit UUID オクテット
#define RBLE_GATT_MAX_RELIABLE_WRITE_CONTENTS 0x10    最大信頼性データ書き込みサイズ
#define RBLE_GATT_MAX_RELIABLE_WRITE_NUM      0x04    最大信頼性データ書き込み個数
```

- GATT 複数読み出し時の期待するレスポンスデータサイズ

```
#define RBLE_GATT_LEN_UNDEF              0xFF       可変長
```

- GATT アトリビュートパーミッション宣言

```
#define  RBLE_GATT_PERM_NONE     0x0000      許可なし
#define  RBLE_GATT_PERM_RD       0x0001      読み出し可
#define  RBLE_GATT_PERM_RD_UNAUTH  0x0002    読み出しには Unauthenticated ペアリングが必要
#define  RBLE_GATT_PERM_RD_AUTH  0x0004      読み出しには Authenticated ペアリングが必要
#define  RBLE_GATT_PERM_RD_AUTZ  0x0008      読み出しには Authorization が必要
#define  RBLE_GATT_PERM_WR       0x0010      書き込み可
#define  RBLE_GATT_PERM_WR_UNAUTH 0x0020     書き込みには Unauthenticated ペアリングが必要
#define  RBLE_GATT_PERM_WR_AUTH  0x0040      書き込みには Authenticated ペアリングが必要
#define  RBLE_GATT_PERM_WR_AUTZ  0x0080      書き込みには Authorization が必要
#define  RBLE_GATT_PERM_NI       0x0100      サーバからの通知・表示可
#define  RBLE_GATT_PERM_NI_UNAUTH 0x0200     通知・表示には Unauthenticated ペアリングが必要
#define  RBLE_GATT_PERM_NI_AUTH  0x0400      通知・表示には Authenticated ペアリングが必要
#define  RBLE_GATT_PERM_NI_AUTZ  0x0800      通知・表示には Authorization が必要
#define  RBLE_GATT_PERM_EKS      0x1000      十分なキーサイズでの暗号化が必要
#define  RBLE_GATT_PERM_HIDE     0x2000      公開しない
```

```
#define  RBLE_GATT_PERM_ENC      0x4000        暗号化が必要
#define  RBLE_GATT_PERM_NOTIFY_COMP_EN  0x8000  Notification 送信完了イベント
                                                通知有効
```

- GATT イベントタイプ列挙型宣言

```
enum RBLE_GATT_EVENT_TYPE_enum {
    RBLE_GATT_EVENT_DISC_SVC_ALL_CMP = 1,     16bit UUID 全サービス検索完了イベント
                                               (Parameters : disc_svc_all_cmp)
    RBLE_GATT_EVENT_DISC_SVC_ALL_128_CMP,     128bit UUID 全サービス検索完了イベント
                                               (Parameters : disc_svc_all_128_cmp)
    RBLE_GATT_EVENT_DISC_SVC_BY_UUID_CMP,     UUID によるサービス検索完了イベント
                                               (Parameters : disc_svc_by_uuid_cmp)
    RBLE_GATT_EVENT_DISC_SVC_INCL_CMP,        インクルードサービス検索完了イベント
                                               (Parameters : disc_svc_incl_cmp)
    RBLE_GATT_EVENT_DISC_CHAR_ALL_CMP,        16bit UUID 全特性検索完了イベント
                                               (Parameters : disc_char_all_cmp)
    RBLE_GATT_EVENT_DISC_CHAR_ALL_128_CMP,    128bit UUID 全特性検索完了イベント
                                              (Parameters : disc_char_all_128_cmp)
    RBLE_GATT_EVENT_DISC_CHAR_BY_UUID_CMP,    16bit UUID による特性検索完了イベント
                                          (Parameters : disc_char_by_uuid_cmp)
    RBLE_GATT_EVENT_DISC_CHAR_BY_UUID_128_CMP, 128bit UUID による特性検索完了イベント
                                        (Parameters : disc_char_by_uuid_128_cmp)
    RBLE_GATT_EVENT_DISC_CHAR_DESC_CMP,      16bit 特性ディスクリプタ検索完了イベント
                                               (Parameters : disc_char_desc_cmp)
    RBLE_GATT_EVENT_DISC_CHAR_DESC_128_CMP, 128bit 特性ディスクリプタ検索完了イベント
                                          (Parameters : disc_char_desc_128_cmp)
    RBLE_GATT_EVENT_READ_CHAR_RESP,         特性・特性ディスクリプタ読み出し応答イベント
                                             (Parameters : read_char_resp)
    RBLE_GATT_EVENT_READ_CHAR_LONG_RESP,      Long 特性読み出し応答イベント
                                               (Parameters : read_char_long_resp)
    RBLE_GATT_EVENT_READ_CHAR_MULT_RESP,      複数特性読み出し応答イベント
                                               (Parameters : read_char_mult_resp)
    RBLE_GATT_EVENT_READ_CHAR_LONG_DESC_RESP,
                                          Long 特性ディスクリプタ読み出し応答イベント
                                        (Parameters : read_char_long_desc_resp)
    RBLE_GATT_EVENT_WRITE_CHAR_RESP,          特性書き込み応答イベント
                                               (Parameters : write_char_resp)
    RBLE_GATT_EVENT_WRITE_CHAR_RELIABLE_RESP, 信頼性特性書き込み応答イベント
                                               (Parameters : write_reliable_resp)
    RBLE_GATT_EVENT_CANCEL_WRITE_CHAR_RESP,   書き込みキャンセル応答イベント
                                               (Parameters : cancel_write_resp)
    RBLE_GATT_EVENT_HANDLE_VALUE_NOTIF,       特性値通知イベント
                                               (Parameters : handle_value_notif)
    RBLE_GATT_EVENT_HANDLE_VALUE_IND,         特性値表示イベント
                                               (Parameters : handle_value_ind)
    RBLE_GATT_EVENT_HANDLE_VALUE_CFM,         特性値表示確認イベント
```

```
                                                (Parameters : handle_value_cfm)
    RBLE_GATT_EVENT_DISCOVERY_CMP,              検索完了イベント
                                                (Parameters : discovery_cmp)
    RBLE_GATT_EVENT_COMPLETE,                   GATT 処理完了イベント
                                                (Parameters : complete)
    RBLE_GATT_EVENT_WRITE_CMD_IND,              書き込み通知イベント
                                                (Parameters : write_cmd_ind)
    RBLE_GATT_EVENT_RESP_TIMEOUT,               GATT 応答タイムアウトイベント
                                                (Parameters : none)
    RBLE_GATT_EVENT_SET_PERM_CMP,               パーミッション設定完了イベント
                                                (Parameters : set_perm_cmp)
    RBLE_GATT_EVENT_SET_DATA_CMP,               データ設定完了イベント
                                                (Parameters : set_data_cmp)
    RBLE_GATT_EVENT_NOTIFY_COMP,                Notification 送信完了イベント
                                                (Parameters : notify_cmp)
    RBLE_GATT_EVENT_COMMAND_DISALLOWED_IND      コマンド拒否通知イベント
                                               (Parameters : cmd_disallowed_ind)
};
```

- GATT イベントタイプ型宣言

```
typedef uint8_t                         RBLE_GATT_EVENT_TYPE;
```

- GATT イベントコールバック関数型宣言

```
typedef void ( *RBLE_GATT_EVENT_HANDLER )( RBLE_GATT_EVENT *event );
```

- GATT 要求タイプ列挙型宣言

```
enum RBLE_GATT_REQ_TYPE_enum {
    RBLE_GATT_DISC_ALL_SVC              = 0x00,     全サービス検索
    RBLE_GATT_DISC_BY_UUID_SVC,                     UUID によるサービス検索
    RBLE_GATT_DISC_INCLUDED_SVC,                    インクルードサービス検索
    RBLE_GATT_DISC_ALL_CHAR,                        全特性検索
    RBLE_GATT_DISC_BY_UUID_CHAR,                    UUID による特性検索
    RBLE_GATT_DISC_DESC_CHAR,                       特性記述子検索

    RBLE_GATT_READ_CHAR,                            特性値読み出し
    RBLE_GATT_READ_BY_UUID_CHAR,                    UUID による特性値読み出し
    RBLE_GATT_READ_LONG_CHAR,                       Long 特性値読み出し
    RBLE_GATT_READ_MULT_LONG_CHAR,                  複数の Long 特性値読み出し
    RBLE_GATT_READ_DESC,                            特性記述子読み出し
    RBLE_GATT_READ_LONG_DESC,                       Long 特性記述子読み出し

    RBLE_GATT_WRITE_NO_RESPONSE,                    レスポンス無しの特性値書き込み
    RBLE_GATT_WRITE_SIGNED,                         署名付き特性値書き込み
    RBLE_GATT_WRITE_CHAR,                           特性値書き込み
    RBLE_GATT_WRITE_LONG_CHAR,                      Long 特性値書き込み
    RBLE_GATT_WRITE_RELIABLE_CHAR,                  信頼性特性値書き込み
    RBLE_GATT_WRITE_DESC,                           特性記述子書き込み
    RBLE_GATT_WRITE_LONG_DESC,                      Long 特性記述子書き込み
    RBLE_GATT_WRITE_CANCEL_CHAR                     特性値書き込みのキャンセル
};
```

- GATT 特性プロパティ列挙型宣言

```
enum RBLE_GATT_CHAR_PROP_enum {
    RBLE_GATT_CHAR_PROP_BCAST           = 0x01,     サーバからブロードキャストする
    RBLE_GATT_CHAR_PROP_RD              = 0x02,     読み出し可能
    RBLE_GATT_CHAR_PROP_WR_NO_RESP      = 0x04,     書き込み可能(レスポンスなし)
    RBLE_GATT_CHAR_PROP_WR              = 0x08,     書き込み可能
    RBLE_GATT_CHAR_PROP_NTF             = 0x10,     サーバから通知する
    RBLE_GATT_CHAR_PROP_IND             = 0x20,     サーバから表示する
    RBLE_GATT_CHAR_PROP_AUTH            = 0x40,     署名付き書き込み可能
    RBLE_GATT_CHAR_PROP_EXT_PROP        = 0x80      拡張プロパティ
};
```

- 検索要求タイプ構造体宣言

```
typedef struct RBLE_GATT_DESIRED_TYPE_t {
    uint16_t  value_size;                           要求データサイズ
    uint8_t   value[RBLE_GATT_128BIT_UUID_OCTET];   要求データ
} RBLE_GATT_DESIRED_TYPE;
```

- UUID 検索タイプ構造体宣言

```
typedef struct RBLE_GATT_UUID_TYPE_t {
    uint8_t   value_size;                              UUID サイズ
    uint8_t   expect_resp_size;                        複数読み出し時の期待サイズ
    uint8_t   value[RBLE_GATT_128BIT_UUID_OCTET];      UUID
} RBLE_GATT_UUID_TYPE;
```

- 信頼性書き込みデータ構造体宣言

```
typedef struct RBLE_GATT_RELIABLE_WRITE_t {
    uint16_t  elmt_hdl;                                特性ハンドル
    uint16_t  size;                                    データサイズ
    uint8_t   value[RBLE_GATT_MAX_RELIABLE_WRITE_CONTENTS];     書き込みデータ
} RBLE_GATT_RELIABLE_WRITE;
```

- サービス検索要求構造体宣言

```
typedef struct RBLE_GATT_DISC_SVC_REQ_t {
    uint8_t   req_type;                                要求タイプ
    uint8_t   reserved;                                予約
    uint16_t  conhdl;                                  コネクションハンドル
    uint16_t  start_hdl;                               検索開始ハンドル
    uint16_t  end_hdl;                                 検索終了ハンドル
    RBLE_GATT_DESIRED_TYPE      desired_svc;           検索要求タイプ
} RBLE_GATT_DISC_SVC_REQ;
```

- 特性検索要求構造体宣言

```
typedef struct RBLE_GATT_DISC_CHAR_REQ_t {
    uint8_t   req_type;                                要求タイプ
    uint8_t   reserved;                                予約
    uint16_t  conhdl;                                  コネクションハンドル
    uint16_t  start_hdl;                               検索開始ハンドル
    uint16_t  end_hdl;                                 検索終了ハンドル
    RBLE_GATT_DESIRED_TYPE      desired_char;          検索要求タイプ
} RBLE_GATT_DISC_CHAR_REQ;
```

- ディスクリプタ検索要求構造体宣言

```
typedef struct RBLE_GATT_DISC_CHAR_DESC_REQ_t {
    uint16_t  conhdl;                                  コネクションハンドル
    uint16_t  start_hdl;                               検索開始ハンドル
    uint16_t  end_hdl;                                 検索終了ハンドル
} RBLE_GATT_DISC_CHAR_DESC_REQ;
```

- 特性読み出し要求構造体宣言

```
typedef struct RBLE_GATT_READ_CHAR_REQ_t {
    uint8_t   req_type;                                  要求タイプ
    uint8_t   reserved;                                  予約
    uint16_t  offset;                                    読み出しオフセット
    uint16_t  conhdl;                                    コネクションハンドル
    uint16_t  start_hdl;                                 開始ハンドル
    uint16_t  end_hdl;                                   終了ハンドル
    uint16_t  nb_uuid;                                   UUID 数
   RBLE_GATT_UUID_TYPE  uuid[RBLE_GATT_MAX_NB_HDLS];     UUID データ
} RBLE_GATT_READ_CHAR_REQ;
```

- 特性書き込み要求構造体宣言

```
typedef struct RBLE_GATT_WRITE_CHAR_REQ_t {
    uint16_t  conhdl;                                    コネクションハンドル
    uint16_t  charhdl;                                   特性ハンドル
    uint16_t  wr_offset;                                 書き込みオフセット
    uint16_t  val_len;                                   書き込み値サイズ
    uint8_t   req_type;                                  要求タイプ
    uint8_t   auto_execute;                              書き込み自動実行フラグ(Long 時)
    uint8_t   value[RBLE_GATT_MAX_LONG_VALUE];           書き込み値
} RBLE_GATT_WRITE_CHAR_REQ;
```

- 信頼性書き込み要求構造体宣言

```
typedef struct RBLE_GATT_WRITE_RELIABLE_REQ_t {
    uint8_t   nb_writes;                                 書き込みデータ数
    uint8_t   auto_execute;                              書き込み自動実行フラグ
    uint16_t  conhdl;                                    コネクションハンドル
    RBLE_GATT_RELIABLE_WRITE  value[RBLE_GATT_MAX_RELIABLE_WRITE_NUM];
                                                         書き込み値
} RBLE_GATT_WRITE_RELIABLE_REQ;
```

- 書き込み実行要求構造体宣言

```
typedef struct RBLE_GATT_EXE_WR_CHAR_REQ_t {
    uint8_t   exe_wr_ena;                                書き込み実行/キャンセルフラグ
    uint8_t   reserved;                                  予約
    uint16_t  conhdl;                                    コネクションハンドル
} RBLE_GATT_EXE_WR_CHAR_REQ;
```

- 通知要求構造体宣言

```
typedef struct RBLE_GATT_NOTIFY_REQ_t {
    uint16_t  conhdl;                                    コネクションハンドル
    uint16_t  charhdl;                                   特性ハンドル
} RBLE_GATT_NOTIFY_REQ;
```

- 表示要求構造体宣言

```
typedef struct RBLE_GATT_INDICATE_REQ_t {
    uint16_t  conhdl;                                   コネクションハンドル
    uint16_t  charhdl;                                  特性ハンドル
} RBLE_GATT_INDICATE_REQ;
```

- 書き込み応答構造体宣言

```
typedef struct RBLE_GATT_WRITE_RESP_t {
    uint16_t  conhdl;                                   コネクションハンドル
    uint16_t  att_hdl;                                  アトリビュートハンドル
    uint8_t   att_code;                                 応答コード
    uint8_t   reserved;                                 予約
} RBLE_GATT_WRITE_RESP;
```

- パーミッション設定構造体宣言

```
typedef struct RBLE_GATT_SET_PERM_t {
    uint16_t  start_hdl;                                開始ハンドル
    uint16_t  end_hdl;                                  終了ハンドル
    uint16_t  perm;                                     パーミッション
} RBLE_GATT_SET_PERM;
```

- データ設定構造体宣言

```
typedef struct RBLE_GATT_SET_DATA_t {
    uint16_t  val_hdl;                                  アトリビュートハンドル
    uint16_t  val_len;                                  設定データサイズ
    uint8_t   value[RBLE_GATT_MAX_LONG_VALUE];          設定データ
} RBLE_GATT_SET_DATA;
```

- 16bit UUID サービスリスト構造体宣言

```
typedef struct RBLE_GATT_SVC_LIST_t {
    uint16_t  start_hdl;                                開始ハンドル
    uint16_t  end_hdl;                                  終了ハンドル
    uint16_t  attr_hdl;                                 サービス UUID
} RBLE_GATT_SVC_LIST;
```

- 128bit UUID サービスリスト構造体宣言

```
typedef struct RBLE_GATT_SVC_128_LIST_t {
    uint16_t  start_hdl;                                開始ハンドル
    uint16_t  end_hdl;                                  終了ハンドル
    uint8_t   attr_hdl[RBLE_GATT_128BIT_UUID_OCTET];    サービス UUID
} RBLE_GATT_SVC_128_LIST;
```

- サービスレンジリスト構造体宣言

```
typedef struct RBLE_GATT_SVC_RANGE_LIST_t {
    uint16_t  start_hdl;                    開始ハンドル
    uint16_t  end_hdl;                      終了ハンドル
} RBLE_GATT_SVC_RANGE_LIST;
```

- 16bit インクルードサービスリスト構造体宣言

```
typedef struct RBLE_GATT_INCL_LIST_t {
    uint16_t  attr_hdl;                     アトリビュートハンドル
    uint16_t  start_hdl;                    開始ハンドル
    uint16_t  end_hdl;                      終了ハンドル
    uint16_t  uuid;                         インクルードサービス UUID
} RBLE_GATT_INCL_LIST;
```

- 128bit インクルードサービスリスト構造体宣言

```
typedef struct RBLE_GATT_INCL_128_LIST_t {
    uint16_t  attr_hdl;                     アトリビュートハンドル
    uint16_t  start_hdl;                    開始ハンドル
    uint16_t  end_hdl;                      終了ハンドル
    uint16_t  uuid[RBLE_GATT_128BIT_UUID_OCTET];    インクルードサービス UUID
} RBLE_GATT_INCL_128_LIST;
```

- 16bit 特性リスト構造体宣言

```
typedef struct RBLE_GATT_CHAR_LIST_t {
    uint16_t  attr_hdl;                     特性ハンドル
    uint8_t   prop;                         特性プロパティ
    uint8_t   reserved;                     予約
    uint16_t  pointer_hdl;                  特性値ハンドル
    uint16_t  uuid;                         特性 UUID
} RBLE_GATT_CHAR_LIST;
```

- 128bit 特性リスト構造体宣言

```
typedef struct RBLE_GATT_CHAR_128_LIST_t {
    uint16_t  attr_hdl;                     特性ハンドル
    uint8_t   prop;                         特性プロパティ
    uint8_t   reserved;                     予約
    uint16_t  pointer_hdl;                  特性値ハンドル
    uint8_t   uuid[RBLE_GATT_128BIT_UUID_OCTET];    特性 UUID
} RBLE_GATT_CHAR_128_LIST;
```

- 16bit 特性ディスクリプタリスト構造体宣言

```
typedef struct RBLE_GATT_CHAR_DESC_LIST_t {
    uint16_t  attr_hdl;                     特性ハンドル
    uint16_t  desc_hdl;                     ディスクリプタ UUID
} RBLE_GATT_CHAR_DESC_LIST;
```

- 128bit 特性ディスクリプタリスト構造体宣言

```
typedef struct RBLE_GATT_CHAR_DESC_128_LIST_t {
    uint16_t  attr_hdl;                          特性ハンドル
    uint8_t   uuid[RBLE_GATT_128BIT_UUID_OCTET];  ディスクリプタ UUID
} RBLE_GATT_CHAR_DESC_128_LIST;
```

- 読み出しデータ構造体宣言

```
typedef struct RBLE_GATT_INFO_DATA_t {
    uint8_t   each_len;                          各ハンドルと値ペアサイズ
    uint8_t   len;                               読み出しデータサイズ
    uint8_t   data[RBLE_GATT_MAX_VALUE];         読み出しデータ
} RBLE_GATT_INFO_DATA;
```

- 複数読み出しデータ構造体宣言

```
typedef struct RBLE_GATT_QUERY_RESULT_t {
    uint8_t   len;                               読み出しデータサイズ
    uint8_t   value[RBLE_GATT_MAX_VALUE];        読み出しデータ
} RBLE_GATT_QUERY_RESULT;
```

- GATT イベントパラメータ構造体

```
typedef struct RBLE_GATT_EVENT_t {
    RBLE_GATT_EVENT_TYPE    type;                GATT イベントタイプ
    uint8_t                 reserved;            予約
    union Event_Gatt_Parameter_u {
        16bit UUID 全サービス検索完了イベント
        struct RBLE_GATT_Disc_Svc_All_Comp_t {
            uint16_t     conhdl;                 コネクションハンドル
            uint8_t      att_code;               ステータス
            uint8_t      nb_resp;                取得リスト数
            RBLE_GATT_SVC_LIST list[RBLE_GATT_MAX_HDL_LIST];
                                                 取得サービスリスト
        } disc_svc_all_cmp;

        128bit UUID 全サービス検索完了イベント
        struct RBLE_GATT_Disc_Svc_All_128_Comp_t {
            uint16_t     conhdl;                 コネクションハンドル
            uint8_t      att_code;               ステータス
            uint8_t      nb_resp;                取得リスト数
            RBLE_GATT_SVC_128_LIST list;         取得サービスリスト
        } disc_svc_all_128_cmp;
```

UUID によるサービス検索完了イベント

```
struct RBLE_GATT_Disc_Svc_By_Uuid_Comp_t {
    uint16_t      conhdl;                       コネクションハンドル
    uint8_t       att_code;                     ステータス
    uint8_t       nb_resp;                      取得リスト数
    RBLE_GATT_SVC_RANGE_LIST list[RBLE_GATT_MAX_HDL_LIST];
                                                取得サービスレンジリスト
} disc_svc_by_uuid_cmp;
```

インクルードサービス検索完了イベント

```
struct RBLE_GATT_Disc_Svc_Incl_Comp_t {
    uint16_t      conhdl;                       コネクションハンドル
    uint8_t       nb_entry;                     取得サービス数
    uint8_t       entry_len;                    取得サービス UUID サイズ
    union incl_list_u {
        RBLE_GATT_INCL_128_LIST incl;           128bit インクルードサービス
        RBLE_GATT_INCL_LIST list[RBLE_GATT_MAX_HDL_LIST];
                                                16bitインクルードサービスリスト
    } incl_list;
} disc_svc_incl_cmp;
```

16bit UUID 全特性検索完了イベント

```
struct RBLE_GATT_Disc_Char_All_Comp_t {
    uint16_t      conhdl;                       コネクションハンドル
    uint8_t       att_code;                     ステータス
    uint8_t       nb_entry;                     取得リスト数
    RBLE_GATT_CHAR_LIST list[RBLE_GATT_MAX_HDL_LIST];
                                                取得特性リスト
} disc_char_all_cmp;
```

128bit UUID 全特性検索完了イベント

```
struct RBLE_GATT_Disc_Char_All_128_Comp_t {
    uint16_t      conhdl;                       コネクションハンドル
    uint8_t       att_code;                     ステータス
    uint8_t       nb_entry;                     取得リスト数
    RBLE_GATT_CHAR_128_LIST list;               取得特性リスト
} disc_char_all_128_cmp;
```

16bit UUID による特性検索完了イベント

```
struct RBLE_GATT_Disc_Char_By_Uuid_Comp_t {
    uint16_t      conhdl;                       コネクションハンドル
    uint8_t       att_code;                     ステータス
    uint8_t       nb_entry;                     取得リスト数
    RBLE_GATT_CHAR_LIST list[RBLE_GATT_MAX_HDL_LIST];
                                                取得特性リスト
} disc_char_by_uuid_cmp;
```

128bit UUID による特性検索完了イベント

```
struct RBLE_GATT_Disc_Char_By_Uuid_128_Comp_t {
    uint16_t      conhdl;                       コネクションハンドル
    uint8_t       att_code;                     ステータス
    uint8_t       nb_entry;                     取得リスト数
    RBLE_GATT_CHAR_128_LIST list;               取得特性リスト
} disc_char_by_uuid_128_cmp;
```

16bit 特性ディスクリプタ検索完了イベント

```
struct RBLE_GATT_Disc_Char_Desc_Comp_t {
    uint16_t      conhdl;                       コネクションハンドル
    uint8_t       nb_entry;                     取得リスト数
    uint8_t       reserved;                     予約
    RBLE_GATT_CHAR_DESC_LIST list[RBLE_GATT_MAX_HDL_LIST];
                                                取得特性ディスクリプタリスト
} disc_char_desc_cmp;
```

128bit 特性ディスクリプタ検索完了イベント

```
struct RBLE_GATT_Disc_Char_Desc_128_Comp_t {
    uint16_t      conhdl;                       コネクションハンドル
    uint8_t       nb_entry;                     取得リスト数
    uint8_t       reserved;                     予約
    RBLE_GATT_CHAR_DESC_128_LIST list_128;      取得特性ディスクリプタリスト
} disc_char_desc_128_cmp;
```

特性・特性ディスクリプタ読み出し応答イベント

```
struct RBLE_GATT_Read_Char_Resp_t {
    uint16_t      conhdl;                       コネクションハンドル
    uint8_t       att_code;                     ステータス
    RBLE_GATT_INFO_DATA data;                   読み出しデータ
} read_char_resp;
```

Long 特性読み出し応答イベント

```
struct RBLE_GATT_Read_Char_Long_Resp_t {
    uint16_t      conhdl;                       コネクションハンドル
    uint8_t       att_code;                     ステータス
    uint8_t       val_len;                      読み出しデータサイズ
    uint16_t      attr_hdl;                     特性ハンドル
    uint8_t       value[RBLE_GATT_MAX_VALUE];   読み出しデータ
} read_char_long_resp;
```

複数特性読み出し応答イベント

```
struct RBLE_GATT_Read_Char_Mult_Resp_t {
    uint16_t      conhdl;                          コネクションハンドル
    uint8_t       att_code;                        ステータス
    uint8_t       val_len;                         読み出しデータサイズ
    RBLE_GATT_QUERY_RESULT data[RBLE_GATT_MAX_NB_HDLS];
                                                   複数読み出しデータ
} read_char_mult_resp;
```

Long 特性ディスクリプタ読み出し応答イベント

```
struct RBLE_GATT_Read_Char_Long_Desc_Resp_t {
    uint16_t      conhdl;                          コネクションハンドル
    uint8_t       att_code;                        ステータス
    uint8_t       val_len;                         読み出しデータサイズ
    uint8_t       value[RBLE_GATT_MAX_VALUE];      読み出しデータ
    uint16_t      attr_hdl;                        特性ディスクリプタハンドル
} read_char_long_desc_resp;
```

特性書き込み応答イベント

```
struct RBLE_GATT_Write_Char_Resp_t {
    uint16_t      conhdl;                          コネクションハンドル
    uint8_t       att_code;                        ステータス
    uint8_t       reserved;                        予約
} write_char_resp;
```

信頼性特性書き込み応答イベント

```
struct RBLE_GATT_Write_Reliable_Resp_t {
    uint16_t      conhdl;                          コネクションハンドル
    uint8_t       att_code;                        ステータス
    uint8_t       reserved;                        予約
} write_reliable_resp;
```

書き込みキャンセル応答イベント

```
struct RBLE_GATT_Cancel_Write_Char_Resp_t {
    uint16_t      conhdl;                          コネクションハンドル
    uint8_t       att_code;                        ステータス
    uint8_t       reserved;                        予約
} cancel_write_resp;
```

特性値通知イベント

```
struct RBLE_GATT_Handle_Value_Notif_t {
    uint16_t     conhdl;                      コネクションハンドル
    uint16_t     charhdl;                     特性ハンドル
    uint8_t      size;                        通知データサイズ
    uint8_t      value[RBLE_GATT_MAX_VALUE];  通知データ
    uint8_t      reserved;                    予約
} handle_value_notif;
```

特性値表示イベント

```
struct RBLE_GATT_Handle_Value_Ind_t {
    uint16_t     conhdl;                      コネクションハンドル
    uint16_t     charhdl;                     特性ハンドル
    uint8_t      size;                        表示データサイズ
    uint8_t      value[RBLE_GATT_MAX_VALUE];  表示データ
    uint8_t      reserved;                    予約
} handle_value_ind;
```

特性値表示確認イベント

```
struct RBLE_GATT_Handle_Value_Cfm_t {
    RBLE_STATUS  status;                      特性値表示結果
} handle_value_cfm;
```

検索完了イベント

```
struct RBLE_GATT_Discovery_Comp_t {
    uint16_t     conhdl;                      コネクションハンドル
    uint8_t      att_code;                    ステータス
    uint8_t      reserved;                    予約
} discovery_cmp;
```

GATT 完了イベント

```
struct RBLE_GATT_Complete_t {
    uint16_t     conhdl;                      コネクションハンドル
    uint8_t      att_code;                    ステータス
    uint8_t      reserved;                    予約
} complete;
```

書き込み通知イベント

```
struct RBLE_GATT_Write_Cmd_Ind_t {
    uint16_t     conhdl;                      コネクションハンドル
    uint16_t     elmt;                        特性ハンドル
    uint16_t     size;                        書き込みデータサイズ
    uint8_t      offset;                      書き込みデータ位置
    bool         resp;                        応答(Confirmation)要否フラグ
    uint8_t      value[RBLE_GATT_MAX_VALUE];  書き込み要求データ
} write_cmd_ind;
```

パーミッション設定完了イベント

```
struct RBLE_GATT_Set_Perm_Complete_t {
    RBLE_STATUS  status;                    パーミッション設定結果
} set_perm_cmp;
```

データ設定完了イベント

```
struct RBLE_GATT_Set_Data_Complete_t {
    RBLE_STATUS  status;                    データ設定結果
} set_data_cmp;
```

Notification 送信完了イベント

```
struct RBLE_GATT_Notify_Comp_t {
    uint16_t     conhdl;                    コネクションハンドル
    uint16_t     charhdl;                   特性ハンドル
    RBLE_STATUS  status;                    送信結果
    uint8_t      reserved;                  予約
} notify_cmp;
```

GATT コマンド拒否通知イベント

```
struct RBLE_EVT_GATT_Command_Disallowed_Ind_t{
    RBLE_STATUS  status;                    ステータス
    uint8_t      reserved;                  予約
    uint16_t     opcode;                    オペコード
} cmd_disallowed_ind;
  } param;
} RBLE_GATT_EVENT;
```

7.2 Functions

rBLE の GATT 機能で定義されている API 関数を表 7-1 に纏めます。次節より、その API 関数の詳細について説明します。

表 7-1 GATT 機能 API 関数一覧

RBLE_GATT_Enable	GATT 機能を有効にする
RBLE_GATT_Discovery_Service_Request	サービスを検索する
RBLE_GATT_Discovery_Char_Request	特性値を検索する
RBLE_GATT_Discovery_Char_Descriptor_Request	特性ディスクリプタを検索する
RBLE_GATT_Read_Char_Request	特性値を読み出す
RBLE_GATT_Write_Char_Request	特性値を書き込む
RBLE_GATT_Write_Reliable_Request	信頼性特性値を書き込む
RBLE_GATT_Execute_Write_Char_Request	特性書き込みの実行を要求する
RBLE_GATT_Notify_Request	特性値の通知を行う
RBLE_GATT_Indicate_Request	特性値の表示を行う
RBLE_GATT_Write_Response	特性値の書き込み要求に応答する
RBLE_GATT_Set_Permission	ローカルデータベースのパーミッションを設定する
RBLE_GATT_Set_Data	ローカルデータベースのデータを設定する

7.2.1 RBLE_GATT_Enable

RBLE_STATUS RBLE_GATT_Enable(RBLE_GATT_EVENT_HANDLER callback)		
このファンクションは、GATT 機能を有効にします。GATT の rBLE API をご使用になる場合は、必ず本ファンクションを呼び出す必要があります。		
Parameters:		
	callback	GATT のイベントを通知するコールバックファンクションを指定
Return:		
	RBLE_OK	正常終了
	RBLE_PARAM_ERR	パラメータ異常
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

7.2.2 RBLE_GATT_Discovery_Service_Request

RBLE_STATUS RBLE_GATT_Discovery_Service_Request(RBLE_GATT_DISC_SVC_REQ *disc_svc)

このファンクションは、リモート GATT サーバのサービス検索を行います。要求タイプにより、全プライマリサービス検索、UUID 指定によるプライマリサービス検索、インクルードサービスの検索のいずれかが実行可能です。

全サービス検索を行った場合、サービスを発見すると、発見したサービスの UUID に応じて、16bit UUID 全サービス検索完了イベント RBLE_GATT_EVENT_DISC_SVC_ALL_CMP または、128bit UUID 全サービス検索完了イベント RBLE_GATT_EVENT_DISC_SVC_ALL_128_CMP が通知されます。
検索の完了は検索完了イベント RBLE_GATT_EVENT_DISCOVERY_CMP で通知されます。

UUID 指定によるサービス検索を行った場合、該当サービスを発見すると UUID によるサービス検索完了イベント RBLE_GATT_EVENT_DISC_SVC_BY_UUID_CMP が通知されます。
検索の完了は GATT 完了イベント RBLE_GATT_EVENT_COMPLETE で通知されます。

インクルードサービス検索を行った場合、該当サービスを発見するとインクルードサービス検索完了イベント RBLE_GATT_EVENT_DISC_SVC_INCL_CMP が通知されます。
検索の完了は検索完了イベント RBLE_GATT_EVENT_DISCOVERY_CMP で通知されます。

Parameters:

*disc_svc	*req_type*	RBLE_GATT_DISC_ALL_SVC	全プライマリサービス検索を実行
		RBLE_GATT_DISC_BY_UUID_SVC	UUID 指定によるプライマリサービス検索を実行
		RBLE_GATT_DISC_INCLUDED_SVC	インクルードサービスの検索を実行
	conhdl	コネクションハンドル	
	start_hdl	検索開始ハンドル(インクルードサービスの検索時のみ有効)	
	end_hdl	検索終了ハンドル(インクルードサービスの検索時のみ有効)	
	desired_svc	全サービス検索時： *value_size*: RBLE_GATT_16BIT_UUID_OCTET のみ指定可 *value[RBLE_GATT_128BIT_UUID_OCTET]*: サービス検索を中断するサービス 16bit UUID(下位バイトより前詰め) UUID 指定によるサービス検索時： *value_size*: 検索対象のサービス UUID オクテットサイズ *value[RBLE_GATT_128BIT_UUID_OCTET]*: 検索対象のサービス UUID(下位バイトより前詰め)	

Return:

RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

7.2.3 RBLE_GATT_Discovery_Char_Request

RBLE_STATUS RBLE_GATT_Discovery_Char_Request(RBLE_GATT_DISC_CHAR_REQ *disc_char)

このファンクションは、リモート GATT サーバの特性検索を行います。要求タイプにより、全特性検索、UUID 指定による特性検索のいずれかが実行可能です。

全特性検索を行った場合、特性を発見すると、発見した特性の UUID に応じて、16bit UUID 全特性検索完了イベント RBLE_GATT_EVENT_DISC_CHAR_ALL_CMP または、128bit UUID 全特性検索完了イベント RBLE_GATT_EVENT_DISC_CHAR_ALL_128_CMP が通知されます。

UUID 指定による特性検索を行った場合、該当特性を発見すると、発見した特性の UUID に応じて、16bit UUID による特性検索完了イベント RBLE_GATT_EVENT_DISC_CHAR_BY_UUID_CMP または、128bit UUID による特性検索完了イベント RBLE_GATT_EVENT_DISC_CHAR_BY_UUID_128_CMP が通知されます。

それぞれの検索の完了は GATT 完了イベント RBLE_GATT_EVENT_COMPLETE で通知されます。

Parameters:

*disc_char	*req_type*	RBLE_GATT_DISC_ALL_CHAR	全特性検索を実行
		RBLE_GATT_DISC_BY_UUID_CHAR	UUID 指定による特性検索を実行
	conhdl	コネクションハンドル	
	start_hdl	検索開始ハンドル	
	end_hdl	検索終了ハンドル	
	desired_char	*value_size*	検索対象の特性 UUID オクテットサイズ(UUID 指定による特性検索時のみ有効)
		value[RBLE_GATT_128BIT_UUID_OCTET]	検索対象の特性 UUID(UUID 指定による特性検索時のみ有効) (下位バイトより前詰め)

Return:

RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

7.2.4 RBLE_GATT_Discovery_Char_Descriptor_Request

RBLE_STATUS RBLE_GATT_Discovery_Char_Descriptor_Request(RBLE_GATT_DISC_CHAR_DESC_REQ *disc_char_desc)		
このファンクションは、リモート GATT サーバの特性ディスクリプタ検索を行います。 指定ハンドル範囲内の特性ディスクリプタを発見すると、発見した特性ディスクリプタの UUID に応じて、16bit 特性ディスクリプタ検索完了イベント RBLE_GATT_EVENT_DISC_CHAR_DESC_CMP または、128bit 特性ディスクリプタ検索完了イベント RBLE_GATT_EVENT_DISC_CHAR_DESC_128_CMP が通知されます。 検索の完了は GATT 完了イベント RBLE_GATT_EVENT_COMPLETE で通知されます。		
Parameters:		
*disc_char_desc	*conhdl*	コネクションハンドル
	start_hdl	検索開始ハンドル
	end_hdl	検索終了ハンドル

Return:	
RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

7.2.5 RBLE_GATT_Read_Char_Request

RBLE_STATUS RBLE_GATT_Read_Char_Request(RBLE_GATT_READ_CHAR_REQ *rd_char)

このファンクションは、リモート GATT サーバの特性値または特性ディスクリプタの読み出しを行います。要求タイプにより、以下のいずれかの読み出しが実行可能です。

- 指定ハンドルの特性値の読み出し
- UUID 指定による特性値の読み出し
- 指定ハンドルの Long 特性値の読み出し
- 複数特性値の読み出し
- 指定ハンドルの特性ディスクリプタの読み出し
- 指定ハンドルの Long 特性ディスクリプタの読み出し

指定ハンドルの特性値または、特性ディスクリプタの読み出しを行った場合、読み出し完了時に、特性・特性ディスクリプタ読み出し応答イベント RBLE_GATT_EVENT_READ_CHAR_RESP が通知されます。

UUID 指定による特性値の読み出しを行った場合、該当特性値読み出し完了時に、特性・特性ディスクリプタ読み出し応答イベント RBLE_GATT_EVENT_READ_CHAR_RESP が通知されます。

指定ハンドルの Long 特性値の読み出しを行った場合、読み出し完了時に、Long 特性読み出し応答イベント RBLE_GATT_EVENT_READ_CHAR_LONG_RESP が通知されます。

複数特性値の読み出しを行った場合、読み出し完了時に、複数特性読み出し応答イベント RBLE_GATT_EVENT_READ_CHAR_MULT_RESP が通知されます。

指定ハンドルの Long 特性ディスクリプタの読み出しを行った場合、読み出し完了時に、Long 特性ディスクリプタ読み出し応答イベント RBLE_GATT_EVENT_READ_CHAR_LONG_DESC_RESP が通知されます。

Parameters:

*rd_char	*req_type*	RBLE_GATT_READ_CHAR	指定ハンドルの特性値の読み出しを実行
		RBLE_GATT_READ_BY_UUID_CHAR	UUID 指定による特性値の読み出しを実行
		RBLE_GATT_READ_LONG_CHAR	指定ハンドルの Long 特性値の読み出しを実行
		RBLE_GATT_READ_MULT_LONG_CHAR	複数特性値の読み出しを実行
		RBLE_GATT_READ_DESC	指定ハンドルの特性ディスクリプタの読み出しを実行
		RBLE_GATT_READ_LONG_DESC	指定ハンドルの Long 特性ディスクリプタの読み出しを実行
	offset	読み出しオフセット(Long 特性値、Long 特性ディスクリプタ読み出し時のみ有効)	
	conhdl	コネクションハンドル	
	start_hdl	読み出し開始ハンドル(UUID 指定による特性値の読み出し時のみ有効。それ以外は 0 を設定してください)	
	end_hdl	読み出し終了ハンドル(UUID 指定による特性値の読み出し時のみ有効。それ以外は 0 を設定してください)	
	nb_uuid	読み出しを行うハンドル数(本パラメータは複数特性値の読み出し時のみ有効)	

<table>
<tr><td colspan="5">RBLE_STATUS RBLE_GATT_Read_Char_Request(RBLE_GATT_READ_CHAR_REQ *rd_char)</td></tr>
<tr><td></td><td></td><td rowspan="3">uuid[RBLE_GATT_MAX_NB_HDLS]</td><td>value_size</td><td>uuid[0].value[ ] に設定した UUID のサイズ
• 16bit UUID の時
RBLE_GATT_16BIT_UUID_OCTET
• 128bit UUID の時
RBLE_GATT_128BIT_UUID_OCTET
(UUID 指定による特性値の読み出し時のみ有効)</td></tr>
<tr><td></td><td></td><td>expect_resp_size</td><td>期待する読み出しデータサイズ（オクテット）。
可変長データが含まれる場合、nb_uuid 番目の要素のみに RBLE_GATT_LEN_UNDEF を設定可能です。 (本パラメータは複数特性値の読み出し時のみ有効)
※各期待する読み出しサイズの合計が 22 オクテット(ATT_MTU – 1)を超えないように設定してください。</td></tr>
<tr><td></td><td></td><td>value[RBLE_GATT_128BIT_UUID_OCTET]</td><td>• UUID 指定による特性値の読み出しの場合
読み出しを行う特性値 UUID
• 上記以外
読み出しを行う既知の特性値ハンドル
UUID およびハンドルは下位バイトより前詰めで設定してください</td></tr>
<tr><td colspan="5">Return:<table>
<tr><td>RBLE_OK</td><td>正常終了</td></tr>
<tr><td>RBLE_STATUS_ERROR</td><td>rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可</td></tr>
</table></td></tr>
</table>

7.2.6 RBLE_GATT_Write_Char_Request

RBLE_STATUS RBLE_GATT_Write_Char_Request(RBLE_GATT_WRITE_CHAR_REQ *wr_char)

このファンクションは、リモート GATT サーバの特性値または特性ディスクリプタへの書き込みを行います。要求タイプにより、以下のいずれかの書き込みが実行可能です。

- レスポンス無しの特性値または特性ディスクリプタ書き込み
- データ署名つきの特性値書き込み
- レスポンス有りの特性値または特性ディスクリプタ書き込み
- Long 特性値または、Long 特性ディスクリプタ書き込み

レスポンス無しの特性値または、特性ディスクリプタ書き込みを行った場合、書き込み完了のイベント通知はされません。

データ署名つきの特性値書き込みを行う場合には、事前に RBLE_SM_Set_Key にて自デバイスの CSRK を設定し、ペアリングにてその CSRK をリモートデバイスに配布しておく必要があります。また、書き込み完了のイベント通知はされません。

レスポンス有りの特性値または、特性ディスクリプタ書き込みを行った場合、書き込み完了時に、特性書き込み応答イベント RBLE_GATT_EVENT_WRITE_CHAR_RESP が通知されます。

Long 特性値または Long 特性ディスクリプタの自動書き込みを行った場合、書き込み完了時に、GATT 完了イベント RBLE_GATT_EVENT_COMPLETE が通知されます。

Parameters:

*wr_char	*conhdl*	コネクションハンドル	
	charhdl	特性値ハンドル	
	wr_offset	書き込みオフセット(Long 特性値または、Long 特性ディスクリプタ書き込み時のみ有効)	
	val_len	書き込みデータサイズ	
	req_type	RBLE_GATT_WRITE_NO_RESPONSE	レスポンス無しの特性値または特性ディスクリプタ書き込みを実行
		RBLE_GATT_WRITE_SIGNED	データ署名つきの特性値書き込みを実行
		RBLE_GATT_WRITE_CHAR	レスポンス有りの特性値書き込みを実行
		RBLE_GATT_WRITE_LONG_CHAR	Long 特性値書き込みを実行
		RBLE_GATT_WRITE_DESC	レスポンス有りの特性ディスクリプタ書き込みを実行
		RBLE_GATT_WRITE_LONG_DESC	Long 特性ディスクリプタ書き込みを実行
	auto_execute	自動書き込みフラグ(TRUE : 自動書き込み実行、FALSE : ユーザが書き込みを実行) (Long 特性値または、Long 特性ディスクリプタ書き込み時のみ有効)	
	value[RBLE_GATT_MAX_LONG_VALUE]	書き込みデータ	

Return:

RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

7.2.7 RBLE_GATT_Write_Reliable_Request

RBLE_STATUS RBLE_GATT_Write_Reliable_Request(RBLE_GATT_WRITE_RELIABLE_REQ *rel_write)				
このファンクションは、リモート GATT サーバ特性値の信頼性書き込みを行います。 特性値の信頼性書き込みの自動実行を行った場合、全データの書き込み完了時に、信頼性特性書き込み応答イベント RBLE_GATT_EVENT_WRITE_CHAR_RELIABLE_RESP が通知されます。				
Parameters:				
	rel_write	*nb_writes*	書き込みデータ数	
		auto_execute	自動書き込みフラグ(TRUE：自動書き込み実行、FALSE：ユーザが書き込みを実行)	
		conhdl	コネクションハンドル	
		value[RBLE_GATT_MAX_RELIABLE_WRITE_NUM]	*elmt_hdl*	特性値ハンドル
			size	書き込みデータサイズ
			value[RBLE_GATT_MAX_RELIABLE_WRITE_CONTENTS]	書き込みデータ
Return:				
	RBLE_OK	正常終了		
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可		

7.2.8 RBLE_GATT_Execute_Write_Char_Request

RBLE_STATUS RBLE_GATT_Execute_Write_Char_Request(RBLE_GATT_EXE_WR_CHAR_REQ *exe_wr_char)			
このファンクションは、リモート GATT サーバのキューイングされた特性値の書き込みを実行またはキャンセルします。 RBLE_GATT_Write_Char_Request(要求タイプ：RBLE_GATT_WRITE_LONG_CHAR または RBLE_GATT_WRITE_LONG_DESC)または、RBLE_GATT_Execute_Write_Char_Request にて自動書き込みを行わなかった場合に、本ファンクションは有効です。 Long 特性値または Long 特性値ディスクリプタの書き込みを実行した場合、全データの書き込み完了時に、GATT 完了イベント RBLE_GATT_EVENT_COMPLETE が通知されます。 信頼性特性値の書き込みを実行した場合、全データの書き込み完了時に、信頼性特性書き込み応答イベント RBLE_GATT_EVENT_WRITE_CHAR_RELIABLE_RESP が通知されます。 書き込みをキャンセルした場合、キャンセル処理完了時に、書き込みキャンセル応答イベント RBLE_GATT_EVENT_CANCEL_WRITE_CHAR_RESP が通知されます。			
Parameters:			
	**exe_wr_char*	*exe_wr_ena*	書き込み実行フラグ (TRUE：書き込み実行、FALSE：書き込みキャンセル)
		conhdl	コネクションハンドル
Return:			
	RBLE_OK	正常終了	
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可	

7.2.9 RBLE_GATT_Notify_Request

RBLE_STATUS RBLE_GATT_Notify_Request(RBLE_GATT_NOTIFY_REQ *notify)			
このファンクションは、ローカル GATT サーバの特性値をリモート GATT クライアントに通知します。 指定されたハンドルの特性値を、ローカル GATT データベースより取得して通知を行います。このため本ファンクションを呼び出す前に、ローカル GATT データベースのデータを更新してください。			
Parameters:			
	**notify*	*conhdl*	コネクションハンドル
		charhdl	特性値ハンドル
Return:			
	RBLE_OK		正常終了
	RBLE_STATUS_ERROR		rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

7.2.10 RBLE_GATT_Indicate_Request

RBLE_STATUS RBLE_GATT_Indicate_Request(RBLE_GATT_INDICATE_REQ *indicate)			
このファンクションは、ローカル GATT サーバの特性値をリモート GATT クライアントに表示します。 指定されたハンドルの特性値を、ローカル GATT データベースより取得して表示を行います。このため本ファンクションを呼び出す前に、ローカル GATT データベースのデータを更新してください。 結果は、特性値表示確認イベント RBLE_GATT_EVENT_HANDLE_VALUE_CFM で通知されます。			
Parameters:			
	**indicate*	*conhdl*	コネクションハンドル
		charhdl	特性値ハンドル
Return:			
	RBLE_OK		正常終了
	RBLE_STATUS_ERROR		rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

7.2.11 RBLE_GATT_Write_Response

RBLE_STATUS RBLE_GATT_Write_Response(RBLE_GATT_WRITE_RESP *wr_resp)			
このファンクションは、リモート GATT クライアントからの特性値書き込み要求に応答します。 ※BLE ソフトウェアではローカル GATT データベースの更新は行いません。リモートからの書き込み要求に応じる場合は、ローカル GATT データベースのデータを更新してください。			
Parameters:			
	**wr_resp*	*conhdl*	コネクションハンドル
		att_hdl	特性値ハンドル
		att_code	書き込み要求応答 (3.2 ATT エラーコード列挙型宣言を参照ください)
Return:			
	RBLE_OK		正常終了
	RBLE_STATUS_ERROR		rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

7.2.12 RBLE_GATT_Set_Permission

RBLE_STATUS RBLE_GATT_Set_Permission(RBLE_GATT_SET_PERM *set_perm)		
このファンクションは、指定ハンドル範囲のローカル GATT データベースのパーミッションを設定します。結果はパーミッション設定完了イベント RBLE_GATT_EVENT_SET_PERM_CMP で通知されます。		
Parameters:		
*set_perm	*start_hdl*	パーミッション設定開始ハンドル
	end_hdl	パーミッション設定終了ハンドル
	perm	パーミッション (GATT アトリビュートパーミッション列挙型宣言より、論理和で設定してください)
Return:		
RBLE_OK		正常終了
RBLE_STATUS_ERROR		rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

7.2.13 RBLE_GATT_Set_Data

RBLE_STATUS RBLE_GATT_Set_Data(RBLE_GATT_SET_DATA *set_data)		
このファンクションは、ローカル GATT データベースの指定されたハンドルのデータを更新します。結果はデータ設定完了イベント RBLE_GATT_EVENT_SET_DATA_CMP で通知されます。		
Parameters:		
*set_data	*val_hdl*	アトリビュートハンドル
	val_len	設定データサイズ
	value[RBLE_GATT_MAX_LONG_VALUE]	設定データ
Return:		
RBLE_OK		正常終了
RBLE_STATUS_ERROR		rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

7.3 Events

rBLE の GATT 機能で定義されているイベントを表 7-2 に纏めます。次節より、そのイベントの詳細について説明します。

表 7-2 GATT 機能イベント一覧

RBLE_GATT_EVENT_DISC_SVC_ALL_CMP	16bit UUID 全サービス検索完了イベント
RBLE_GATT_EVENT_DISC_SVC_ALL_128_CMP	128bit UUID 全サービス検索完了イベント
RBLE_GATT_EVENT_DISC_SVC_BY_UUID_CMP	UUID によるサービス検索完了イベント
RBLE_GATT_EVENT_DISC_SVC_INCL_CMP	インクルードサービス検索完了イベント
RBLE_GATT_EVENT_DISC_CHAR_ALL_CMP	16bit UUID 全特性検索完了イベント
RBLE_GATT_EVENT_DISC_CHAR_ALL_128_CMP	128bit UUID 全特性検索完了イベント
RBLE_GATT_EVENT_DISC_CHAR_BY_UUID_CMP	16bit UUID による特性検索完了イベント
RBLE_GATT_EVENT_DISC_CHAR_BY_UUID_128_CMP	128bit UUID による特性検索完了イベント
RBLE_GATT_EVENT_DISC_CHAR_DESC_CMP	16bit 特性ディスクリプタ検索完了イベント
RBLE_GATT_EVENT_DISC_CHAR_DESC_128_CMP	128bit 特性ディスクリプタ検索完了イベント
RBLE_GATT_EVENT_READ_CHAR_RESP	特性・特性ディスクリプタ読み出し応答イベント
RBLE_GATT_EVENT_READ_CHAR_LONG_RESP	Long 特性読み出し応答イベント
RBLE_GATT_EVENT_READ_CHAR_MULT_RESP	複数特性読み出し応答イベント
RBLE_GATT_EVENT_READ_CHAR_LONG_DESC_RESP	Long 特性ディスクリプタ読み出し応答イベント
RBLE_GATT_EVENT_WRITE_CHAR_RESP	特性書き込み応答イベント
RBLE_GATT_EVENT_WRITE_CHAR_RELIABLE_RESP	信頼性特性書き込み応答イベント
RBLE_GATT_EVENT_CANCEL_WRITE_CHAR_RESP	書き込みキャンセル応答イベント
RBLE_GATT_EVENT_HANDLE_VALUE_NOTIF	特性値通知イベント
RBLE_GATT_EVENT_HANDLE_VALUE_IND	特性値表示イベント
RBLE_GATT_EVENT_HANDLE_VALUE_CFM	特性値表示確認イベント
RBLE_GATT_EVENT_DISCOVERY_CMP	検索完了イベント
RBLE_GATT_EVENT_COMPLETE	GATT 処理完了イベント
RBLE_GATT_EVENT_WRITE_CMD_IND	書き込み通知イベント
RBLE_GATT_EVENT_RESP_TIMEOUT	GATT 応答タイムアウトイベント
RBLE_GATT_EVENT_SET_PERM_CMP	パーミッション設定完了イベント
RBLE_GATT_EVENT_SET_DATA_CMP	データ設定完了イベント
RBLE_GATT_EVENT_NOTIFY_COMP	Notification 送信完了イベント
RBLE_GATT_EVENT_COMMAND_DISALLOWED_IND	GATT コマンド拒否通知イベント

7.3.1 RBLE_GATT_EVENT_DISC_SVC_ALL_CMP

RBLE_GATT_EVENT_DISC_SVC_ALL_CMP				
このイベントは、リモート GATT サーバの 16bit UUID プライマリサービス検索結果を通知します。 MTU の制限などにより一度に通知できない場合は、本イベントは複数回通知されます。				
Parameters:				
disc_svc_all_cmp	*conhdl*	コネクションハンドル		
	att_code	サービス検索結果 ※0x00 以外の場合、以下のパラメータは無効です。 (3.2 ATT エラーコード列挙型宣言を参照ください)		
	nb_resp	検索結果個数 ※以下のパラメータは、本個数分有効です。		
	list[RBLE_GATT_MAX_HDL_LIST]	*start_hdl*	サービス開始ハンドル	
		end_hdl	サービス終了ハンドル	
		attr_hdl	16bit サービス UUID	

7.3.2 RBLE_GATT_EVENT_DISC_SVC_ALL_128_CMP

RBLE_GATT_EVENT_DISC_SVC_ALL_128_CMP			
このイベントは、リモート GATT サーバの 128bit UUID プライマリサービス検索結果を通知します。 128bit UUID のプライマリサービスを発見するたびに、本イベントは通知されます。			
Parameters:			
disc_svc_all_128_cmp	*conhdl*	コネクションハンドル	
	att_code	サービス検索結果 ※0x00 以外の場合、以下のパラメータは無効です。 (3.2 ATT エラーコード列挙型宣言を参照ください)	
	nb_resp	検索結果個数	
	list	*start_hdl*	サービス開始ハンドル
		end_hdl	サービス終了ハンドル
		attr_hdl[RBLE_GATT_128BIT_UUID_OCTET]	128bit サービス UUID

7.3.3 RBLE_GATT_EVENT_DISC_SVC_BY_UUID_CMP

RBLE_GATT_EVENT_DISC_SVC_BY_UUID_CMP			
このイベントは、UUID によるリモート GATT サーバのプライマリサービス検索結果を通知します。 MTU の制限などにより一度に通知できない場合は、本イベントは複数回通知されます。			
Parameters:			
disc_svc_by_uuid_cmp	*conhdl*	コネクションハンドル	
	att_code	サービス検索結果 ※0x00 以外の場合、以下のパラメータは無効です。 (3.2 ATT エラーコード列挙型宣言を参照ください)	
	nb_resp	検索結果個数	
	list[RBLE_GATT_MAX_HDL_LIST]	*start_hdl*	サービス開始ハンドル
		end_hdl	サービス終了ハンドル

7.3.4 RBLE_GATT_EVENT_DISC_SVC_INCL_CMP

RBLE_GATT_EVENT_DISC_SVC_INCL_CMP					
このイベントは、リモート GATT サーバのインクルードサービス検索結果を通知します。 MTU の制限などにより一度に通知できない場合は、本イベントは複数回通知されます。					
Parameters:					
disc_svc_incl_cmp	*conhdl*	コネクションハンドル			
	nb_entry	検索結果個数			
	entry_len	UUID サイズ RBLE_GATT_128BIT_UUID_OCTET : 下記パラメータの *incl* にアクセスしてください RBLE_GATT_16BIT_UUID_OCTET : 下記パラメータの *list[ ]*にアクセスしてください。*nb_entry* 分の要素が有効です。			
	incl_list	*incl*	*attr_hdl*	アトリビュートハンドル	
			start_hdl	サービス開始ハンドル	
			end_hdl	サービス終了ハンドル	
			uuid[RBLE_GATT_128BIT_UUID_OCTET]	サービス UUID	
		list[RBLE_GATT_MAX_HDL_LIST]	*attr_hdl*	アトリビュートハンドル	
			start_hdl	サービス開始ハンドル	
			end_hdl	サービス終了ハンドル	
			uuid	サービス UUID	

7.3.5 RBLE_GATT_EVENT_DISC_CHAR_ALL_CMP

RBLE_GATT_EVENT_DISC_CHAR_ALL_CMP				
このイベントは、リモート GATT サーバの 16bit UUID 特性検索結果を通知します。 MTU の制限などにより一度に通知できない場合は、本イベントは複数回通知されます。				
Parameters:				
disc_char_all_cmp	*conhdl*	コネクションハンドル		
	att_code	特性検索結果 ※0x00 以外の場合、以下のパラメータは無効です。 (3.2 ATT エラーコード列挙型宣言を参照ください)		
	nb_entry	検索結果個数 ※以下のパラメータは、本個数分有効です。		
	list[RBLE_GATT_MAX_HDL_LIST]	*attr_hdl*	特性ハンドル	
		prop	特性値プロパティ	
		pointer_hdl	特性値ハンドル	
		uuid	特性値 UUID	

7.3.6 RBLE_GATT_EVENT_DISC_CHAR_ALL_128_CMP

RBLE_GATT_EVENT_DISC_CHAR_ALL_128_CMP				
このイベントは、リモート GATT サーバの 128bit UUID 特性検索結果を通知します。 128bit UUID の特性を発見するたびに、本イベントは通知されます。				
Parameters:				
disc_char_all_128_cmp	*conhdl*	コネクションハンドル		
	att_code	特性検索結果 ※0x00 以外の場合、以下のパラメータは無効です。 (3.2 ATT エラーコード列挙型宣言を参照ください)		
	nb_entry	検索結果個数		
	list	*attr_hdl*	特性ハンドル	
		prop	特性値プロパティ	
		pointer_hdl	特性値ハンドル	
		uuid[RBLE_GATT_128BIT_UUID_OCTET]	特性値 UUID	

7.3.7 RBLE_GATT_EVENT_DISC_CHAR_BY_UUID_CMP

RBLE_GATT_EVENT_DISC_CHAR_BY_UUID_CMP			
このイベントは、指定 16bit UUID によるリモート GATT サーバの特性検索結果を通知します。 MTU の制限などにより一度に通知できない場合は、本イベントは複数回通知されます。			
Parameters:			
disc_char_by_uuid_cmp	*conhdl*	コネクションハンドル	
	att_code	特性検索結果 ※0x00 以外の場合、以下のパラメータは無効です。 (3.2 ATT エラーコード列挙型宣言を参照ください)	
	nb_entry	検索結果個数 ※以下のパラメータは、本個数分有効です。	
	list[RBLE_GATT_MAX_HDL_LIST]	*attr_hdl*	指定 UUID の特性ハンドル
		prop	特性値プロパティ
		pointer_hdl	特性値ハンドル
		uuid	特性値 UUID

7.3.8 RBLE_GATT_EVENT_DISC_CHAR_BY_UUID_128_CMP

RBLE_GATT_EVENT_DISC_CHAR_BY_UUID_128_CMP				
このイベントは、指定 128bit UUID によるリモート GATT サーバの 128bit UUID 特性検索結果を通知します。 指定 128bit UUID の特性を発見するたびに、本イベントは通知されます。				
Parameters:				
disc_char_by_uuid_128_cmp	*conhdl*	コネクションハンドル		
	att_code	特性検索結果 ※0x00 以外の場合、以下のパラメータは無効です。 (3.2 ATT エラーコード列挙型宣言を参照ください)		
	nb_entry	検索結果個数		
	list	*attr_hdl*	特性ハンドル	
		prop	特性値プロパティ	
		pointer_hdl	特性値ハンドル	
		uuid[RBLE_GATT_128BIT_UUID_OCTET]	特性値 UUID	

7.3.9 RBLE_GATT_EVENT_DISC_CHAR_DESC_CMP

RBLE_GATT_EVENT_DISC_CHAR_DESC_CMP			
このイベントは、リモート GATT サーバの 16bit UUID 特性ディスクリプタ検索結果を通知します。 MTU の制限などにより一度に通知できない場合は、本イベントは複数回通知されます。			
Parameters:			
disc_char_desc_cmp	*conhdl*	コネクションハンドル	
	nb_entry	検索結果個数 ※以下のパラメータは、本個数分有効です。	
	list[RBLE_GATT_MAX_HDL_LIST]	*attr_hdl*	特性ディスクリプタハンドル
		desc_hdl	特性ディスクリプタ UUID

7.3.10 RBLE_GATT_EVENT_DISC_CHAR_DESC_128_CMP

RBLE_GATT_EVENT_DISC_CHAR_DESC_128_CMP			
このイベントは、リモート GATT サーバの 128bit UUID 特性ディスクリプタ検索結果を通知します。 128bit UUID の特性ディスクリプタを発見するたびに、本イベントは通知されます。			
Parameters:			
disc_char_desc_128_cmp	*conhdl*	コネクションハンドル	
	nb_entry	検索結果個数	
	list_128	*attr_hdl*	特性ハンドル
		uuid[RBLE_GATT_128BIT_UUID_OCTET]	特性値 UUID

7.3.11 RBLE_GATT_EVENT_READ_CHAR_RESP

RBLE_GATT_EVENT_READ_CHAR_RESP			
このイベントは、リモート GATT サーバからの特性・特性ディスクリプタ読み出し応答を通知します。			
Parameters:			
read_char_resp	*conhdl*	コネクションハンドル	
	att_code	読み出し結果 ※0x00 以外の場合、以下のパラメータは無効です。 (3.2 ATT エラーコード列挙型宣言を参照ください)	
	data	*each_len*	各ハンドルと値ペアサイズ ※UUID 指定による特性値の読み出しを行った場合にのみ有効
		len	読み出しデータサイズ
		data[RBLE_GATT_MAX_VALUE]	読み出しデータ

7.3.12 RBLE_GATT_EVENT_READ_CHAR_LONG_RESP

RBLE_GATT_EVENT_READ_CHAR_LONG_RESP		
このイベントは、リモート GATT サーバからの Long 特性読み出し応答を通知します。 特性値すべての読み出しが完了するまで、本イベントは複数回通知されます。		
Parameters:		
read_char_long_resp	*conhdl*	コネクションハンドル
	att_code	読み出し結果 ※0x00 以外の場合、以下のパラメータは無効です。 (3.2 ATT エラーコード列挙型宣言を参照ください)
	val_len	読み出しデータサイズ サイズが 22 オクテット(ATT_MTU – 1)未満の場合、特性値すべての読み出しが完了したことを意味します。
	attr_hdl	特性値ハンドル
	value[RBLE_GATT_MAX_VALUE]	読み出しデータ

7.3.13 RBLE_GATT_EVENT_READ_CHAR_MULT_RESP

RBLE_GATT_EVENT_READ_CHAR_MULT_RESP				
このイベントは、リモート GATT サーバからの複数特性読み出し応答を通知します。				
Parameters:				
read_char_mult_resp	*conhdl*	コネクションハンドル		
	att_code	読み出し結果 ※0x00 以外の場合、以下のパラメータは無効です。 (3.2 ATT エラーコード列挙型宣言を参照ください)		
	val_len	総読み出しデータサイズ		
	data[RBLE_GATT_MAX_NB_HDLS]	*len*	読み出しデータサイズ	
		value[RBLE_GATT_MAX_VALUE]	読み出しデータ	

7.3.14 RBLE_GATT_EVENT_READ_CHAR_LONG_DESC_RESP

RBLE_GATT_EVENT_READ_CHAR_LONG_DESC_RESP		
このイベントは、リモート GATT サーバからの Long 特性ディスクリプタ読み出し応答を通知します。 特性ディスクリプタすべての読み出しが完了するまで、本イベントは複数回通知されます。		
Parameters:		
read_char_long_desc_resp	*conhdl*	コネクションハンドル
	att_code	読み出し結果 ※0x00 以外の場合、以下のパラメータは無効です。 (3.2 ATT エラーコード列挙型宣言を参照ください)
	val_len	読み出しデータサイズ サイズが 22 オクテット(ATT_MTU – 1)未満の場合、特性ディスクリプタすべての読み出しが完了したことを意味します。
	value[RBLE_GATT_MAX_VALUE]	読み出しデータ
	attr_hdl	特性ディスクリプタハンドル

7.3.15 RBLE_GATT_EVENT_WRITE_CHAR_RESP

RBLE_GATT_EVENT_WRITE_CHAR_RESP		
このイベントは、特性書き込み要求に対するリモート GATT サーバからの応答を通知します。		
Parameters:		
write_char_resp	*conhdl*	コネクションハンドル
	att_code	特性書き込み結果 (3.2 ATT エラーコード列挙型宣言を参照ください)

7.3.16 RBLE_GATT_EVENT_WRITE_CHAR_RELIABLE_RESP

RBLE_GATT_EVENT_WRITE_CHAR_RELIABLE_RESP		
このイベントは、信頼性特性書き込み要求に対するリモート GATT サーバからの応答を通知します。		
Parameters:		
write_reliable_resp	*conhdl*	コネクションハンドル
	att_code	信頼性特性書き込み結果 (3.2 ATT エラーコード列挙型宣言を参照ください)

7.3.17 RBLE_GATT_EVENT_CANCEL_WRITE_CHAR_RESP

RBLE_GATT_EVENT_CANCEL_WRITE_CHAR_RESP		
このイベントは、特性書き込みキャンセル要求に対する、リモート GATT サーバからの応答を通知します。		
Parameters:		
cancel_write_resp	*conhdl*	コネクションハンドル
	att_code	特性書き込みキャンセル結果 (3.2 ATT エラーコード列挙型宣言を参照ください)

7.3.18 RBLE_GATT_EVENT_HANDLE_VALUE_NOTIF

RBLE_GATT_EVENT_HANDLE_VALUE_NOTIF		
このイベントは、リモート GATT サーバからの特性値通知(Notification)を通知します。		
Parameters:		
handle_value_notif	*conhdl*	コネクションハンドル
	charhdl	特性値ハンドル
	size	通知データサイズ
	value[RBLE_GATT_MAX_VALUE]	通知データ

7.3.19 RBLE_GATT_EVENT_HANDLE_VALUE_IND

RBLE_GATT_EVENT_HANDLE_VALUE_IND		
このイベントは、リモート GATT サーバからの特性値表示(Indication)を通知します。 ※リモート GATT サーバに対する表示確認応答(Confirmation)は BLE ソフトウェア内部で自動的に行います。		
Parameters:		
handle_value_ind	*conhdl*	コネクションハンドル
	charhdl	特性値ハンドル
	size	表示データサイズ
	value[RBLE_GATT_MAX_VALUE]	表示データ

7.3.20 RBLE_GATT_EVENT_HANDLE_VALUE_CFM

RBLE_GATT_EVENT_HANDLE_VALUE_CFM		
このイベントは、リモート GATT クライアントからの特性値表示確認応答(Confirmation)を通知します。		
Parameters:		
handle_value_cfm	*status*	特性値表示結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

7.3.21 RBLE_GATT_EVENT_DISCOVERY_CMP

RBLE_GATT_EVENT_DISCOVERY_CMP		
このイベントは、サービス検索(全検索、インクルードサービス検索)の完了を通知します。		
Parameters:		
discovery_cmp	*conhdl*	コネクションハンドル
	att_code	サービス検索結果 (3.2 ATT エラーコード列挙型宣言を参照ください)

7.3.22 RBLE_GATT_EVENT_COMPLETE

RBLE_GATT_EVENT_COMPLETE		
このイベントは、GATT の処理が完了したことを通知します。		
Parameters:		
complete	*conhdl*	コネクションハンドル
	att_code	GATT の処理結果 (3.2 ATT エラーコード列挙型宣言を参照ください)

7.3.23 RBLE_GATT_EVENT_WRITE_CMD_IND

RBLE_GATT_EVENT_WRITE_CMD_IND		
このイベントは、リモート GATT クライアントからの特性値書き込み要求を通知します。 書き込み要求に対しレスポンスが必要な場合は、RBLE_GATT_Write_Response にて応答してください。レスポンスの要否は、パラメータ *resp* にて判定してください。 データの正当性を確認し、正常なデータであれば RBLE_GATT_Set_Data にてローカル GATT データベースの該当データを更新してください。		
Parameters:		
write_cmd_ind	*conhdl*	コネクションハンドル
	elmt	特性値ハンドル
	size	書き込み要求データサイズ
	offset	書き込み要求データオフセット
	resp	書き込み要求に対するレスポンス要否 (TRUE：レスポンス要、FALSE：レスポンス不要)
	value[RBLE_GATT_MAX_VALUE]	書き込み要求データ

7.3.24 RBLE_GATT_EVENT_RESP_TIMEOUT

RBLE_GATT_EVENT_RESP_TIMEOUT
このイベントは、GATT の処理中にリモートデバイスからの応答タイムアウトが発生したことを通知します。 ※タイムアウト時間は 30 秒です。
Parameters:
none

7.3.25 RBLE_GATT_EVENT_SET_PERM_CMP

RBLE_GATT_EVENT_SET_PERM_CMP		
このイベントは、ローカル GATT データベースのパーミッション設定結果を通知します。		
Parameters:		
set_perm_cmp	*status*	パーミッション設定結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

7.3.26 RBLE_GATT_EVENT_SET_DATA_CMP

RBLE_GATT_EVENT_SET_DATA_CMP		
このイベントは、ローカル GATT データベースのデータ設定結果を通知します。		
Parameters:		
set_data_cmp	*status*	データ設定結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

7.3.27 RBLE_GATT_EVENT_NOTIFY_COMP

RBLE_GATT_EVENT_NOTIFY_COMP		
このイベントは、リモート GATT クライアントに対し特性値通知(Notification)の送信が完了したことを通知します。このイベントを有効にするには、該当の特性値が、GATT データベースにおいて RBLE_GATT_PERM_NOTIFY_COMP_EN パーミッションを設定する必要があります。 ※このイベントは送信を保証するものではありません		
Parameters:		
notify_cmp	*conhdl*	コネクションハンドル
	charhdl	特性値ハンドル
	status	送信結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

7.3.28 RBLE_GATT_EVENT_COMMAND_DISALLOWED_IND

RBLE_GATT_EVENT_COMMAND_DISALLOWED_IND		
このイベントは、GATT コマンドが拒否されたことを通知します。		
Parameters:		
cmd_disallowed_ind	*status*	コマンド実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
	opcode	拒否されたコマンドのオペコード

8. Vendor Specific

このセクションは、Vendor Specific(以下、VS)の API について記載します。VS を使用することで、Direct Test Mode やルネサス独自の Direct Test Mode 拡張機能等が利用可能です。

8.1 Definitions

このセクションは、VS の API で使用される定義について記載します。

- GPIO ビット定義

```
#define RBLE_VS_GPIO_BIT_0              0x01        Bit0
#define RBLE_VS_GPIO_BIT_1              0x02        Bit1
#define RBLE_VS_GPIO_BIT_2              0x04        Bit2
#define RBLE_VS_GPIO_BIT_3              0x08        Bit3
```

- GPIO 入出力方向定義

```
#define RBLE_VS_GPIO_INPUT              0           入力
#define RBLE_VS_GPIO_OUTPUT             1           出力
```

- GPIO 入出力値定義

```
#define RBLE_VS_GPIO_LOW                0           Low
#define RBLE_VS_GPIO_HIGH               1           High
```

- GPIO 入出力方向設定マクロ定義

```
#define RBLE_VS_GPIO_DIR_SETTING(val, bit, dir)  ¥
                        val = (uint8_t)(((dir)==RBLE_VS_GPIO_INPUT)¥
                        ?((uint8_t)(val)&~(bit))¥
                        :((uint8_t)(val)|(bit)))
```

- GPIO 出力設定マクロ定義

```
#define RBLE_VS_GPIO_OUTPUT_SETTING(val, bit, set)     ¥
                        val = (uint8_t)(((set)==RBLE_VS_GPIO_LOW)¥
                        ?((uint8_t)(val)&~(bit))¥
                        :((uint8_t)(val)|(bit)))
```

- VS イベントタイプ列挙型宣言

```
enum RBLE_VS_EVENT_TYPE_enum {
    RBLE_VS_EVENT_TEST_RX_START_COMP = 0x01,   受信テスト開始完了イベント
                                               (Parameters : status)
    RBLE_VS_EVENT_TEST_TX_START_COMP,      送信テスト開始完了イベント
                                           (Parameters : status)
    RBLE_VS_EVENT_TEST_END_COMP,           テスト終了イベント
                                           (Parameters : test_end_cmp)
    RBLE_VS_EVENT_WR_BD_ADDR_COMP,         BD アドレス書き込み完了イベント
                                           (Parameters : status)
```

```
    RBLE_VS_EVENT_SET_TEST_PARAM_COMP,     Direct Test Mode 時の
                                           拡張パラメータ設定完了イベント
                                           (Parameters : status)
    RBLE_VS_EVENT_READ_TEST_RSSI_COMP,     Direct Test Mode 時の RSSI 取得完了イベント
                                           (Parameters : test_rssi_cmp)
    RBLE_VS_EVENT_GPIO_DIR_COMP,           GPIO 入出力方向設定完了イベント
                                           (Parameters : gpio_dir_cmp)
    RBLE_VS_EVENT_GPIO_ACCESS_COMP,        GPIO アクセス完了イベント*/
                                           (Parameters : gpio_access_cmp)
    RBLE_VS_EVENT_FLASH_MANAGEMENT_COMP,   Data Flash アクセス管理コマンド完了イベント
                                           (Parameters : management_comp)
    RBLE_VS_EVENT_FLASH_ACCESS_COMP,     Data Flash データアクセスコマンド完了イベント
                                           (Parameters : access_comp)
    RBLE_VS_EVENT_FLASH_OPERATION_COMP,    Data Flash ブロック操作完了イベント
                                           (Parameters : operation_comp)
    RBLE_VS_EVENT_FLASH_GET_SPACE_COMP,    Data Flash 空きサイズ取得完了イベント
                                           (Parameters : get_space)
    RBLE_VS_EVENT_FLASH_GET_EEL_VER_COMP, Data Flash EEL バージョン取得完了イベント
                                           (Parameters : get_eel_ver)
    RBLE_VS_EVENT_ADAPT_ENABLE_COMP,       アダプタブル機能有効完了イベント
                                           (Parameters : adapt_enable_cmp)
    RBLE_VS_EVENT_ADAPT_STATE_IND,         アダプタブル機能状態変化通知イベント
                                           (Parameters : adapt_state_ind)
    RBLE_VS_EVENT_COMMAND_DISALLOWED_IND, VS コマンド拒否通知イベント
                                           (Parameters : cmd_disallowed_ind)
    RBLE_VS_EVENT_SET_TX_POWER_COMP,       送信パワー設定完了イベント
                                           (Parameters : status)
    RBLE_VS_EVENT_SET_PARAMS_COMP,         パラメータ設定完了イベント
                                           (Parameters : status)
    RBLE_VS_EVENT_RF_CONTROL_COMP          RF 電源制御完了イベント
                                           (Parameters : rf_control_cmp)
};
```

- VS イベントタイプ型宣言

```
typedef uint8_t                         RBLE_VS_EVENT_TYPE;
```

- VS イベントコールバック関数型宣言

```
typedef void ( *RBLE_VS_EVENT_HANDLER )( RBLE_VS_EVENT *event );
```

- 送信データパターン列挙型宣言

```
enum RBLE_TEST_DATA_PATTERN_enum {
    RBLE_TEST_DATA_PATTERN_PN9          = 0x00,     擬似ランダムビットシーケンス 9
    RBLE_TEST_DATA_PATTERN_11110000     = 0x01,     ビットパターン'11110000'
    RBLE_TEST_DATA_PATTERN_10101010     = 0x02,     ビットパターン'10101010'
    RBLE_TEST_DATA_PATTERN_PN15         = 0x03,     擬似ランダムビットシーケンス 15
    RBLE_TEST_DATA_PATTERN_ALL1         = 0x04,     全 bit が 1
    RBLE_TEST_DATA_PATTERN_ALL0         = 0x05,     全 bit が 0
    RBLE_TEST_DATA_PATTERN_00001111     = 0x06,     ビットパターン'00001111'
    RBLE_TEST_DATA_PATTERN_01010101     = 0x07      ビットパターン'01010101'
};
```

- 送信パワーレベル列挙型宣言

```
enum RBLE_VS_TXPW_SET_LEVEL_enum {
    RBLE_VS_TXPW_LV1        = 0x01,             -15dbm
    RBLE_VS_TXPW_LV2        = 0x02,             -10dbm
    RBLE_VS_TXPW_LV3        = 0x03,             -7dbm
    RBLE_VS_TXPW_LV4        = 0x04,             -2dbm
    RBLE_VS_TXPW_LV5        = 0x05,             予約
    RBLE_VS_TXPW_LV6        = 0x06,             予約
    RBLE_VS_TXPW_LV7        = 0x07,             -1dbm
    RBLE_VS_TXPW_LV8        = 0x08,             予約
    RBLE_VS_TXPW_LV9        = 0x09              0dbm
};
```

- 送信パワー設定モード列挙型宣言

```
enum RBLE_VS_TXPW_MODE_enum {
    RBLE_VS_TXPW_MODE_NORMAL,                   アダプタブル機能無効時
    RBLE_VS_TXPW_MODE_ADAPT_NEAR,               RF ロー・パワーモード
    RBLE_VS_TXPW_MODE_ADAPT_MIDDLE,             RF ノーマルモード
    RBLE_VS_TXPW_MODE_ADAPT_FAR                 RF ハイ・パフォーマンスモード
};
```

- GPIO アクセスモード列挙型宣言

```
enum RBLE_VS_GPIO_MD_enum {
    RBLE_VS_GPIO_INPUT_MD,                      入力モード
    RBLE_VS_GPIO_OUTPUT_MD,                     出力モード
};
```

- アダプタブル状態列挙型宣言

```
enum RBLE_VS_ADAPT_STATE_enum {
    RBLE_VS_ADAPT_MODE_NEAR,                    RF ロー・パワーモード状態
    RBLE_VS_ADAPT_MODE_MIDDLE,                  RF ノーマルモード状態
    RBLE_VS_ADAPT_MODE_FAR                      RF ハイ・パフォーマンスモード状態
};
```

- アダプタブル機能有効・無効コマンド列挙型宣言

```
enum RBLE_VS_ADAPT_CMD_enum {
    RBLE_VS_ADAPT_CMD_DISABLE        = 0x00,   アダプタブル機能動作禁止
    RBLE_VS_ADAPT_CMD_ENABLE         = 0x01    アダプタブル機能動作許可、状態通知許可
    RBLE_VS_ADAPT_CMD_ENABLE_WO_IND = 0x81    アダプタブル機能動作許可、状態通知禁止
};
```

- Flash 制御コマンド列挙型宣言

```
enum RBLE_VS_FLASH_CMD_enum {
    RBLE_VS_FLASH_CMD_START,                  アクセス開始
    RBLE_VS_FLASH_CMD_STOP,                   アクセス停止
    RBLE_VS_FLASH_CMD_WRITE,                  データ書き込み
    RBLE_VS_FLASH_CMD_READ,                   データ読み込み
    RBLE_VS_FLASH_CMD_CLEANUP,                データ配置整理
    RBLE_VS_FLASH_CMD_FORMAT                  データ一括消去
};
```

- RF 電源制御コマンド列挙型宣言

```
enum RBLE_VS_RFCNTL_CMD_enum {
    RBLE_VS_RFCNTL_CMD_POWDOWN,               RF 電源 OFF
    RBLE_VS_RFCNTL_CMD_POWUP_DDCON,           RF 電源 ON(DC-DC コンバータ有効)
    RBLE_VS_RFCNTL_CMD_POWUP_DDCOFF           RF 電源 ON(DC-DC コンバータ無効)
};
```

- パラメータ設定タイプ列挙型宣言

```
enum RBLE_VS_SET_PARAM_enum {
    RBLE_VS_PARAM_DISC_SCAN_TIME = 0x00,      gap_discovery_scan_time
    RBLE_VS_PARAM_DISC_SCAN_INTV,             gap_dev_search_scan_intv
    RBLE_VS_PARAM_DISC_SCAN_WIND,             gap_dev_search_scan_window
    RBLE_VS_PARAM_LIM_ADV_TO,                 gap_lim_adv_timeout
    RBLE_VS_PARAM_SCAN_FAST_INTV,             gap_scan_fast_intv
    RBLE_VS_PARAM_SCAN_FAST_WIND,             gap_scan_fast_window
    RBLE_VS_PARAM_CONN_INTV_MIN,              gap_init_conn_min_intv
    RBLE_VS_PARAM_CONN_INTV_MAX,              gap_init_conn_max_intv
    RBLE_VS_PARAM_CONN_CE_MIN,                gap_conn_min_ce_length
    RBLE_VS_PARAM_CONN_CE_MAX,                gap_conn_max_ce_length
    RBLE_VS_PARAM_CONN_SLAVE_LATENCY,         gap_conn_slave_latency
    RBLE_VS_PARAM_CONN_SVTO,                  gap_dev_supervision_timeout
    RBLE_VS_PARAM_RPA_INTV,                   gap_resolvable_private_addr_intv
    RBLE_VS_PARAM_USER_DEFINED_TOP  = 0x80    ユーザ定義パラメータ先頭
};
```

- Flash アクセスパラメータ構造体

```
typedef struct RBLE_VS_FLASH_ACCESS_PARAM_t {
    uint8_t cmd;                                実行コマンド
    uint8_t id;                                 データ ID
    uint8_t size;                               データサイズ
    uint8_t reserved;
    uint8_t *addr;                              データバッファアドレス
} RBLE_VS_FLASH_ACCESS_PARAM;
```

- VS イベントパラメータ構造体

```
typedef struct RBLE_VS_EVENT_t {
    RBLE_VS_EVENT_TYPE      type;               VS イベントタイプ
    uint8_t                 reserved;           予約
    union Event_Parameter_u {
        Generic イベント
        RBLE_STATUS         status;             ステータス

        テスト終了イベント
        struct RBLE_VS_Test_End_Comp_t{
            RBLE_STATUS     status;             ステータス
            uint8_t         reserved;           予約
            uint16_t        nb_packet_received; 受信パケット数
        }test_end_cmp;

        Direct Test Mode 時の RSSI 取得完了イベント
        struct RBLE_VS_Read_Test_RSSI_Comp_t {
          RBLE_STATUS       status;             ステータス
          uint8_t           rssi;               RSSI 値
        } test_rssi_cmp;

        GPIO 入出力方向設定完了イベント
        struct RBLE_VS_GPIO_Dir_Comp_t {
          RBLE_STATUS       status;             ステータス
          uint8_t           mask;               GPIO マスク
        } gpio_dir_cmp;

        GPIO アクセス完了イベント
        struct RBLE_VS_GPIO_Access_Comp_t {
          RBLE_STATUS       status;             ステータス
          uint8_t           value;              GPIO 入力値
        } gpio_access_cmp;
```

Data Flash アクセス管理コマンド完了イベント

```
struct RBLE_VS_Flash_Management_Comp_t {
  RBLE_STATUS       status;                 ステータス
  uint8_t           cmd;                    実行コマンド
}management_comp;
```

Data Flash データアクセス完了イベント

```
struct RBLE_VS_Flash_Access_Comp_t {
  RBLE_STATUS       status;                 ステータス
  uint8_t           cmd;                    実行コマンド
  uint8_t           id;                     データ ID
  uint8_t           size;                   データサイズ
  uint8_t           *addr;                  データバッファアドレス
}access_comp;
```

Data Flash ブロック操作完了イベント

```
struct RBLE_VS_Flash_Operation_Comp_t {
  RBLE_STATUS       status;                 ステータス
  uint8_t           cmd;                    実行コマンド
}operation_comp;
```

Data Flash 空きサイズ取得完了イベント

```
struct RBLE_VS_Flash_Get_Space_Comp_t {
  RBLE_STATUS       status;                 ステータス
  uint8_t           reserved;               予約
  uint16_t          size;                   空きサイズ
}get_space;
```

Data Flash EEL バージョン情報完了イベント

```
struct RBLE_VS_Flash_Get_EEL_Ver_Comp_t {
  RBLE_STATUS       status;                 ステータス
  uint8_t           version[24];            バージョン情報
}get_eel_ver;
```

アダプタブル機能有効完了イベント

```
struct RBLE_VS_Adapt_Enable_Comp_t {
  RBLE_STATUS       status;                 ステータス
  uint8_t           cmd;                    アダプタブル機能
                                            有効・無効コマンド
}adapt_enable_cmp;
```

アダプタブル状態通知イベント

```
struct RBLE_VS_Adapt_State_Ind_t {
  uint8_t           state;                  アダプタブル状態
}adapt_state_ind;
```

RF 電源制御効完了イベント

```
struct RBLE_VS_RF_Control_Comp_t {
  RBLE_STATUS       status;                 ステータス
}rf_control_cmp;
```

VS コマンド拒否通知イベント

```
struct RBLE_VS_Command_Disallowed_Ind_t{
    RBLE_STATUS     status;                 ステータス
    uint8_t         reserved;               予約
    uint16_t        opcode;                 オペコード
}cmd_disallowed_ind;
  } param;
} RBLE_SM_EVENT;
```

8.2 Functions

rBLE の VS 機能で定義されている API 関数を以下に纏めます。次節より、その API 関数の詳細について説明します。

表 8-1 VS 機能 API 関数一覧

RBLE_VS_Enable	VS 機能を有効にする
RBLE_VS_Test_Rx_Start	受信テストを開始する
RBLE_VS_Test_Tx_Start	送信テストを開始する
RBLE_VS_Test_End	テストを終了する(送受信共通)
RBLE_VS_Set_Test_Parameter	Direct Test Mode 時の拡張パラメータを設定する
RBLE_VS_Read_Test_RSSI	Direct Test Mode 時の RSSI を取得する
RBLE_VS_Write_Bd_Address	BD アドレスを書き込む
RBLE_VS_Set_Tx_Power	送信パワーを設定する
RBLE_VS_GPIO_Dir	GPIO の入出力方向を設定する
RBLE_VS_GPIO_Access	GPIO にアクセスする
RBLE_VS_Flash_Management	Data Flash アクセス管理コマンドを実行する
RBLE_VS_Flash_Access	Data Flash にアクセスする
RBLE_VS_Flash_Operation	Data Flash ブロック操作を行う
RBLE_VS_Flash_Get_Space	Data Flash 空き容量を取得する
RBLE_VS_Flash_Get_EEL_Ver	Data Flash アクセスに使用する EEL バージョンを取得する
RBLE_VS_Adapt_Enable	アダプタブル機能を有効/無効にする
RBLE_VS_RF_Control	RF チップの電源を制御する
RBLE_VS_Set_Params	パラメータを設定する

8.2.1 RBLE_VS_Enable

RBLE_STATUS RBLE_VS_Enable(RBLE_VS_EVENT_HANDLER callback)		
このファンクションは、VS 機能を有効にします。		
Parameters:		
	callback	VS のイベントを通知するコールバックファンクションを指定
Return:		
	RBLE_OK	正常終了
	RBLE_PARAM_ERR	パラメータ異常
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.2.2 RBLE_VS_Test_Rx_Start

RBLE_STATUS RBLE_VS_Test_Rx_Start(uint8_t rx_freq)		
このファンクションは、受信テストを開始します。結果は受信テスト開始完了イベント RBLE_VS_EVENT_TEST_RX_START_COMP で通知されます。		
Parameters:		
	rx_freg	受信周波数 N = (F – 2402) / 2 (範囲：0x00～0x27、F：2402MHz～2480MHz)
Return:		
	RBLE_OK	正常終了
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.2.3 RBLE_VS_Test_Tx_Start

RBLE_STATUS RBLE_VS_Test_Tx_Start(uint8_t tx_freq, uint8_t test_data_len, uint8_t pk_payload_type)			
このファンクションは、送信テストを開始します。結果は送信テスト開始完了イベント RBLE_VS_EVENT_TEST_TX_START_COMP で通知されます。			
Parameters:			
	tx_freg	送信周波数 N = (F – 2402) / 2 (N：0x00～0x27、F：2402MHz～2480MHz)	
	test_data_len	送信パケットペイロード長(0x00～0x25)	
	pk_payload_type	RBLE_TEST_DATA_PATTERN_PN9	擬似ランダムビットシーケンス 9
		RBLE_TEST_DATA_PATTERN_11110000	ビットパターン'11110000'
		RBLE_TEST_DATA_PATTERN_10101010	ビットパターン'10101010'
		RBLE_TEST_DATA_PATTERN_PN15	擬似ランダムビットシーケンス 15
		RBLE_TEST_DATA_PATTERN_ALL1	全 bit が 1
		RBLE_TEST_DATA_PATTERN_ALL0	全 bit が 0
		RBLE_TEST_DATA_PATTERN_00001111	ビットパターン'00001111'
		RBLE_TEST_DATA_PATTERN_01010101	ビットパターン'01010101'
Return:			
	RBLE_OK	正常終了	
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可	

8.2.4 RBLE_VS_Test_End

RBLE_STATUS RBLE_VS_Test_End(void)		
このファンクションは、実行中の受信または送信テストを終了します。結果はテスト終了イベント RBLE_VS_EVENT_TEST_END_COMP で通知されます。		
Parameters:		
	none	
Return:		
	RBLE_OK	正常終了
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.2.5 RBLE_VS_Set_Test_Parameter

RBLE_STATUS RBLE_VS_Set_Test_Parameter(uint16_t rx_nb_packet, uint16_t tx_nb_packet, uint8_t infinite_setting)	
このファンクションは、Direct Test Mode の拡張機能用のパラメータを設定します。 拡張機能には以下のものがあります。 • 受信テスト時に、指定パケット数受信でテスト終了 (指定パケット数受信またはテスト終了関数(RBLE_VS_Test_End)呼び出しにてテスト終了) • 送信テスト時に、指定パケット数送信でテスト終了 (指定パケット数送信またはテスト終了関数(RBLE_VS_Test_End)呼び出しにてテスト終了) • 送信または受信テスト時にバースト転送を行う (テスト終了関数(RBLE_VS_Test_End)呼び出しにてテスト終了) • 送信テスト時に連続搬送波(CW)出力を行う (テスト終了関数(RBLE_VS_Test_End)呼び出しにてテスト終了) 結果は拡張機能用のパラメータ設定完了イベント RBLE_VS_EVENT_SET_TEST_PARAM_COMP で通知されます。 ※本機能を利用するには Direct Test Mode 実行前に本ファンクションを呼び出す必要があります。	
Parameters:	
rx_nb_packet	受信テスト時、テストを終了する受信パケット数 (0 を指定した場合は自動でテストを終了しない)
tx_nb_packet	送信テスト時、テストを終了する送信パケット数 (0 を指定した場合は自動でテストを終了しない)
infinite_setting	0：バースト転送無効、1：バースト転送有効、2：連続搬送波(CW)出力有効
Return:	
RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.2.6 RBLE_VS_Read_Test_RSSI

RBLE_STATUS RBLE_VS_Read_Test_RSSI(void)	
このファンクションは、受信 Direct Test Mode 実行時の RSSI 値を取得します。 結果は Direct Test Mode 時の RSSI 取得完了イベント RBLE_VS_EVENT_READ_TEST_RSSI_COMP で通知されます。 ※RSSI 値は、受信 DirectTestMode の実行開始から、DirectTestMode 完了後の通常パケット受信直前まで取得可能です。	
Parameters:	
none	
Return:	
RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.2.7 RBLE_VS_Write_Bd_Address

RBLE_STATUS RBLE_VS_Write_Bd_Address(RBLE_BD_ADDR *address)		
このファンクションは、指定パブリックアドレスを Data Flash へ書き込みます。 結果は BD アドレス書き込み完了イベント RBLE_VS_EVENT_WR_BD_ADDR_COMP で通知されます。 ※書き込みを行った BD アドレスは次回起動後の GAP リセット処理(RBLE_GAP_Reset)完了で有効となります。 ※このファンクションを実行する前に、RBLE_VS_Flash_Management にて Data Flash へのアクセスを開始してください。また、BD アドレスの書き込みが完了するまで、パラメータで指定したバッファは保持しておく必要があります。		
Parameters:		
	address	DataFlash へ格納するパブリックアドレス
Return:		
	RBLE_OK	正常終了
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.2.8 RBLE_VS_Set_Tx_Power

RBLE_STATUS RBLE_VS_Set_Tx_Power(uint16_t conhdl, uint8_t power_lvl , uint8_t state)		
このファンクションは、指定コネクションハンドルの送信パワーを設定します。 結果は送信パワー設定完了イベント RBLE_VS_EVENT_SET_TX_POWER_COMP で通知されます。 ※以下に該当する場合、接続中の送信パワー変更は意図しない動作を引き起こす場合がありますのでご注意ください。 ・Proximity プロファイルで Tx Power レベルを公開する場合 ・Advertising データに Tx Power レベル AD タイプを含める場合		
Parameters:		
	conhdl	コネクションハンドル 0x10 を指定することで Advertising/Scaning/Initiating 時の送信パワーを設定可能です。
	power_lvl	送信パワーレベル RBLE_VS_TXPW_LV1：-15dBm RBLE_VS_TXPW_LV2：-10dBm RBLE_VS_TXPW_LV3：-7dBm RBLE_VS_TXPW_LV4：-2dBm RBLE_VS_TXPW_LV5：予約 RBLE_VS_TXPW_LV6：予約 RBLE_VS_TXPW_LV7：-1dBm RBLE_VS_TXPW_LV8：予約 RBLE_VS_TXPW_LV9：0dBm
	state	送信パワーを設定する動作状態 RBLE_VS_TXPW_MODE_NORMAL：アダプタブル機能無効時 RBLE_VS_TXPW_MODE_ADAPT_NEAR：RF ロー・パワーモード RBLE_VS_TXPW_MODE_ADAPT_MIDDLE：RF ノーマルモード RBLE_VS_TXPW_MODE_ADAPT_FAR：RF ハイ・パフォーマンスモード
Return:		
	RBLE_OK	正常終了
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.2.9 RBLE_VS_GPIO_Dir

RBLE_STATUS RBLE_VS_GPIO_Dir (uint8_t dir)		
このファンクションは、RF チップ GPIO[3:0]端子の入出力方向を設定します。 兼用機能の動作設定時は、以下の兼用機能を優先します。 GPIO[0]：外部パワーアンプ使用時は、TXSELH_RF 機能を優先 GPIO[1]：外部パワーアンプ使用時は、TXSELL_RF 機能を優先 GPIO[2]：高速クロック出力時は、CLKOUT_RF を優先 GPIO[3]：32kHz スリープクロックの入出力時は、CLK32KIN/EXSLK_RF を優先 出力に設定した GPIO 端子の初期出力値は Low(0)となります。 結果は GPIO 入出力方向設定完了イベント RBLE_VS_EVENT_GPIO_DIR_COMP で通知されます。 ※兼用機能の動作設定は、Bluetooth Low Energy プロトコルスタック・ユーザーズマニュアルを参照ください。		
Parameters:		
	dir	各 GPIO 端子の入出力方向 (RBLE_VS_GPIO_INPUT：入力、RBLE_VS_GPIO_OUTPUT：出力) bit3：GPIO3 入出力設定ビット bit2：GPIO2 入出力設定ビット bit1：GPIO1 入出力設定ビット bit0：GPIO0 入出力設定ビット
Return:		
	RBLE_OK	正常終了
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.2.10 RBLE_VS_GPIO_Access

RBLE_STATUS RBLE_VS_GPIO_Access (uint8_t mode, uint8_t value)		
このファンクションは、RF チップ GPIO[3:0]端子の入力値を取得または、出力値を設定します。このファンクションを呼び出す前に、RBLE_VS_GPIO_Dir にて各 GPIO 端子の入出力方向を設定してください。 結果は GPIO アクセス完了イベント RBLE_VS_EVENT_GPIO_ACCESS_COMP で通知されます。 ※RF チップが Deep Sleep に遷移した場合、GPIO[2:0]端子は GPIO 機能の入力設定、GPIO[3]端子は兼用機能の出力設定にリセットされるため、出力値は維持されません。Deep Sleep からの起床時に、本ファンクションで設定した出力値に復帰します。		
Parameters:		
	mode	入力値の取得・出力値の設定 RBLE_VS_GPIO_INPUT_MD：入力値の取得 RBLE_VS_GPIO_OUTPUT_MD：出力値の設定
	value	各 GPIO 端子の出力値(出力設定端子のみ有効) (RBLE_VS_GPIO_LOW：0、RBLE_VS_GPIO_HIGH：1) bit3：GPIO3 出力値ビット bit2：GPIO2 出力値ビット bit1：GPIO1 出力値ビット bit0：GPIO0 出力値ビット
Return:		
	RBLE_OK	正常終了
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.2.11 RBLE_VS_Flash_Management

RBLE_STATUS RBLE_VS_Flash_Management (uint8_t cmd)		
このファンクションは、コマンドで指定した Data Flash のアクセス管理機能を実行します。 結果は Data Flash アクセス管理コマンド完了イベント RBLE_VS_EVENT_FLASH_MANAGEMENT_COMP で通知されます。 ※Data Flash のアクセス開始から停止までの期間中、SLEEP 機能は無効となります。		
Parameters:		
	cmd	Data Flash アクセス管理コマンド RBLE_VS_FLASH_CMD_START：アクセス開始 RBLE_VS_FLASH_CMD_STOP：アクセス停止
Return:		
	RBLE_OK	正常終了
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.2.12 RBLE_VS_Flash_Access

RBLE_STATUS RBLE_VS_Flash_Access (RBLE_VS_FLASH_ACCESS_PARAM *param)		
このファンクションは、Data Flash へデータの書き込みまたは、データの読み出しを行います。 結果は Data Flash データアクセスコマンド完了イベント RBLE_VS_EVENT_FLASH_ACCESS_COMP で通知されます。 ※このファンクションを実行する前に、RBLE_VS_Flash_Management にて Data Flash へのアクセスを開始してください。また、データの書き込みまたは読み出しが完了するまで、パラメータで指定したバッファは保持しておく必要があります。		
Parameters:		
	cmd	Data Flash アクセスコマンド RBLE_VS_FLASH_CMD_WRITE：データ書き込み RBLE_VS_FLASH_CMD_READ：データ読み出し
	id	データ ID(0x01 – 0xFF)
	size	データサイズ(1 ～ 255 バイト)
	*addr	書き込み・読み出しバッファへのポインタ
Return:		
	RBLE_OK	正常終了
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.2.13 RBLE_VS_Flash_Operation

RBLE_STATUS RBLE_VS_Flash_Operation (uint8_t cmd)		
このファンクションは、Data Flash のブロック操作を行います。 結果は Data Flash ブロック操作完了イベント RBLE_VS_EVENT_FLASH_OPERATION_COMP で通知されます。 ※このファンクションを実行する前に、RBLE_VS_Flash_Management にて Data Flash へのアクセスを開始してください。データ配置整理実行時、Data Flash に格納されている BD アドレスは退避され、データ配置整理完了後に再度 Data Flash に書き込まれます。		
Parameters:		
	cmd	Data Flash ブロック操作コマンド RBLE_VS_FLASH_CMD_CLEANUP：データ配置整理 RBLE_VS_FLASH_CMD_FORMAT：データー括消去
Return:		
	RBLE_OK	正常終了
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.2.14 RBLE_VS_Flash_Get_Space

RBLE_STATUS RBLE_VS_Flash_Get_Space (void)		
このファンクションは、Data Flash の現在の有効ブロックおよび準備ブロックの合計空き容量を取得します。 結果は Data Flash 空きサイズ取得完了イベント RBLE_VS_EVENT_FLASH_GET_SPACE_COMP で通知されます。		
Parameters:		
	none	
Return:		
	RBLE_OK	正常終了
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.2.15 RBLE_VS_Flash_Get_EEL_Ver

RBLE_STATUS RBLE_VS_Flash_Get_EEL_Ver (void)		
このファンクションは、Data Flash アクセスに使用する EEPROM Emulation Library(EEL)のバージョン情報を取得します。 結果は Data Flash EEL バージョン取得完了イベント RBLE_VS_EVENT_FLASH_GET_EEL_VER_COMP で通知されます。		
Parameters:		
	none	
Return:		
	RBLE_OK	正常終了
	RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.2.16 RBLE_VS_Adapt_Enable

RBLE_STATUS RBLE_VS_Adapt_Enable (uint8_t cmd)	
このファンクションは、アダプタブル機能を有効・無効に設定します。 結果はアダプタブル機能有効完了イベント RBLE_VS_EVENT_ADAPT_ENABLE_COMP で通知されます。 ※以下に該当する場合はアダプタブル機能を有効にしないでください。 ・Proximity プロファイルで Tx Power レベルを公開する場合 ・Advertising データに Tx Power レベル AD タイプを含める場合	
Parameters:	
cmd	アダプタブル機能有効・無効コマンド RBLE_VS_ADAPT_CMD_DISABLE：アダプタブル動作禁止 RBLE_VS_ADAPT_CMD_ENABLE：アダプタブル動作許可、状態通知許可 RBLE_VS_ADAPT_CMD_ENABLE_WO_IND：アダプタブル動作許可、状態通知禁止
Return:	
RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.2.17 RBLE_VS_RF_Control

RBLE_STATUS RBLE_VS_RF_Control (uint8_t cmd)	
このファンクションは、RF チップの電源を制御します。 結果は RF 電源制御完了イベント RBLE_VS_EVENT_RF_CONTROL_COMP で通知されます。 ※RF 電源 OFF 中は、以下の機能が使用できません。 • RF 部高速クロックの外部出力 • RF 部 GPIO[3:0]端子制御 • ke_timer 機能 • RSCIP 再送機能(Modem 構成) ※RF 電源を ON に設定後は、RBLE_GAP_Reset 関数を呼び出してください。	
Parameters:	
cmd	RF 電源制御コマンド RBLE_VS_RFCNTL_CMD_POWDOWN：RF 電源 OFF RBLE_VS_RFCNTL_CMD_POWUP_DDCON：RF 電源 ON(DC-DC コンバータ有効) RBLE_VS_RFCNTL_CMD_POWUP_DDCOFF：RF 電源 ON (DC-DC コンバータ無効)
Return:	
RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.2.18 RBLE_VS_Set_Params

RBLE_STATUS RBLE_VS_Set_Params (uint8_t param_id, uint8_t param_len, uint8_t *param_data)

このファンクションは、BLE MCU 内のパラメータ値を設定します。
結果はパラメータ設定完了イベント RBLE_VS_EVENT_SET_PARAMS_COMP で通知されます。
※param_id は 0x80 以降をユーザが自由に使用可能です。param_id に 0x80 以降を設定した場合、arch_main.c の RBLE_User_Set_Params 関数が呼び出されますので、任意の処理を実装し、処理結果を戻り値に設定してください。

Parameters:

param_id	設定パラメータ ID ※RBLE_VS_PARAM_USER_DEFINED_TOP(0x80)以降は、ユーザが自由に使用可能です。
param_len	パラメータ長
**param_data*	パラメータデータ格納先へのポインタ(データは下位バイトより前詰め)

設定パラメータ ID	変数名
RBLE_VS_PARAM_DISC_SCAN_TIME	gap_discovery_scan_time
RBLE_VS_PARAM_DISC_SCAN_INTV	gap_dev_search_scan_intv
RBLE_VS_PARAM_DISC_SCAN_WIND	gap_dev_search_scan_window
RBLE_VS_PARAM_LIM_ADV_TO	gap_lim_adv_timeout
RBLE_VS_PARAM_SCAN_FAST_INTV	gap_scan_fast_intv
RBLE_VS_PARAM_SCAN_FAST_WIND	gap_scan_fast_window
RBLE_VS_PARAM_CONN_INTV_MIN	gap_init_conn_min_intv
RBLE_VS_PARAM_CONN_INTV_MAX	gap_init_conn_max_intv
RBLE_VS_PARAM_CONN_CE_MIN	gap_conn_min_ce_length
RBLE_VS_PARAM_CONN_CE_MAX	gap_conn_max_ce_length
RBLE_VS_PARAM_CONN_SLAVE_LATENCY	gap_conn_slave_latency
RBLE_VS_PARAM_CONN_SVTO	gap_dev_supervision_timeout
RBLE_VS_PARAM_RPA_INTV	gap_resolvable_private_addr_intv

Return:

RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.3 Events

rBLE の VS 機能で定義されているイベントを以下に纏めます。次節より、そのイベントの詳細について説明します。

表 8-2 VS 機能イベント一覧

RBLE_VS_EVENT_TEST_RX_START_COMP	受信テスト開始完了イベント
RBLE_VS_EVENT_TEST_TX_START_COMP	送信テスト開始完了イベント
RBLE_VS_EVENT_TEST_END_COMP	テスト終了イベント
RBLE_VS_EVENT_WR_BD_ADDR_COMP	BD アドレス書き込み完了イベント
RBLE_VS_EVENT_SET_TEST_PARAM_COMP	Direct Test Mode 時の拡張パラメータ設定完了イベント
RBLE_VS_EVENT_READ_TEST_RSSI_COMP	Direct Test Mode 時の RSSI 取得完了イベント
RBLE_VS_EVENT_GPIO_DIR_COMP	GPIO 入出力方向設定完了イベント
RBLE_VS_EVENT_GPIO_ACCESS_COMP	GPIO アクセス完了イベント
RBLE_VS_EVENT_FLASH_MANAGEMENT_COMP	Data Flash アクセス管理コマンド完了イベント
RBLE_VS_EVENT_FLASH_ACCESS_COMP	Data Flash データアクセスコマンド完了イベント
RBLE_VS_EVENT_FLASH_OPERATION_COMP	Data Flash ブロック操作完了イベント
RBLE_VS_EVENT_FLASH_GET_SPACE_COMP	Data Flash 空きサイズ取得完了イベント
RBLE_VS_EVENT_FLASH_GET_EEL_VER_COMP	Data Flash EEL バージョン取得完了イベント
RBLE_VS_EVENT_ADAPT_ENABLE_COMP	アダプタブル機能有効完了イベント
RBLE_VS_EVENT_ADAPT_STATE_IND	アダプタブル状態通知イベント
RBLE_VS_EVENT_COMMAND_DISALLOWED_IND	VS コマンド拒否通知イベント
RBLE_VS_EVENT_SET_TX_POWER_COMP	送信パワー設定完了イベント
RBLE_VS_EVENT_SET_PARAMS_COMP	パラメータ設定完了イベント
RBLE_VS_EVENT_RF_CONTROL_COMP	RF 電源制御完了イベント

8.3.1 RBLE_VS_EVENT_TEST_RX_START_COMP

RBLE_VS_EVENT_TEST_RX_START_COMP	
このイベントは、受信テスト開始完了を通知します。	
Parameters:	
status	受信テスト開始結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

8.3.2 RBLE_VS_EVENT_TEST_TX_START_COMP

RBLE_VS_EVENT_TEST_TX_START_COMP	
このイベントは、送信テスト開始完了を通知します。	
Parameters:	
status	送信テスト開始結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

8.3.3 RBLE_VS_EVENT_TEST_END_COMP

RBLE_VS_EVENT_TEST_END_COMP	
このイベントは、実行中の受信または送信テストの終了を通知します。	
Parameters:	
status	テスト終了結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
nb_packet_received	受信テスト中に受信したパケット数 ※送信テスト終了時、本パラメータは無効です。

8.3.4 RBLE_VS_EVENT_WR_BD_ADDR_COMP

RBLE_VS_EVENT_WR_BD_ADDR_COMP	
このイベントは、BD アドレスの書き込み完了を通知します。	
Parameters:	
status	BD アドレスの書き込み完了結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

8.3.5 RBLE_VS_EVENT_SET_TEST_PARAM_COMP

RBLE_VS_EVENT_SET_TEST_PARAM_COMP	
このイベントは、Direct Test Mode 時の拡張パラメータ設定完了を通知します。	
Parameters:	
status	Direct Test Mode 時の拡張パラメータ設定完了結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

8.3.6 RBLE_VS_EVENT_READ_TEST_RSSI_COMP

RBLE_VS_EVENT_READ_TEST_RSSI_COMP	
このイベントは、受信 Direct Test Mode 時の RSSI 取得完了を通知します。	
Parameters:	
status	受信 Direct Test Mode 時の RSSI 取得完了結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
rssi	RSSI 値(単位：dBm) ※*status* が RBLE_OK 以外の場合、本パラメータは無効です。

8.3.7 RBLE_VS_EVENT_GPIO_DIR_COMP

RBLE_VS_EVENT_GPIO_DIR_COMP	
このイベントは、RF チップ GPIO[3:0]端子の入出力方向設定完了を通知します。	
Parameters:	
status	GPIO 入出力方向設定完了結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
mask	GPIO マスク bit3:GPIO3 マスクビット(1:GPIO,0:兼用機能で使用) bit2:GPIO2 マスクビット(1:GPIO,0:兼用機能で使用) bit1:GPIO1 マスクビット(1:GPIO,0:兼用機能で使用) bit0:GPIO0 マスクビット(1:GPIO,0:兼用機能で使用)

8.3.8 RBLE_VS_EVENT_GPIO_ACCESS_COMP

RBLE_VS_EVENT_GPIO_ACCESS_COMP	
このイベントは、RF チップ GPIO[3:0]端子の入力値取得または、出力値設定の完了を通知します。	
Parameters:	
status	GPIO 入力値取得・出力値設定完了結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
value	GPIO 入力取得値(入力設定端子のみ有効) (RBLE_VS_GPIO_LOW：0、RBLE_VS_GPIO_HIGH：1) bit3:GPIO3 入力値ビット bit2:GPIO2 入力値ビット bit1:GPIO1 入力値ビット bit0:GPIO0 入力値ビット

8.3.9 RBLE_VS_EVENT_FLASH_MANAGEMENT_COMP

RBLE_VS_EVENT_FLASH_MANAGEMENT_COMP	
このイベントは、Data Flash アクセス管理コマンド実行結果を通知します。	
Parameters:	
status	Data Flash アクセス管理コマンド実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
cmd	実行コマンド

8.3.10 RBLE_VS_EVENT_FLASH_ACCESS_COMP

RBLE_VS_EVENT_FLASH_ACCESS_COMP	
このイベントは、Data Flash データアクセスコマンド実行結果を通知します。	
Parameters:	
status	Data Flash アクセスコマンド実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
cmd	実行コマンド
id	データ ID
size	データサイズ
**addr*	データ格納先へのポインタ

8.3.11 RBLE_VS_EVENT_FLASH_OPERATION_COMP

RBLE_VS_EVENT_FLASH_OPERATION_COMP	
このイベントは、Data Flash ブロック操作コマンド実行結果を通知します。	
Parameters:	
status	Data Flash ブロック操作コマンド実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
cmd	実行コマンド

8.3.12 RBLE_VS_EVENT_FLASH_GET_SPACE_COMP

RBLE_VS_EVENT_FLASH_GET_SPACE_COMP	
このイベントは、Data Flash 空きサイズ取得完了結果を通知します。	
Parameters:	
status	Data Flash 空きサイズ取得完了結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
wsize	空きワード数(1 ワードは 4byte)

8.3.13 RBLE_VS_EVENT_FLASH_GET_EEL_VER_COMP

RBLE_VS_EVENT_FLASH_GET_EEL_VER_COMP	
このイベントは、Data Flash アクセスに使用する EEL バージョン情報取得完了結果を通知します。	
Parameters:	
status	EEL バージョン情報取得完了結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
version[24]	バージョン情報

8.3.14 RBLE_VS_EVENT_ADAPT_ENABLE_COMP

RBLE_VS_EVENT_ADAPT_ENABLE_COMP	
このイベントは、アダプタブル機能の有効・無効設定完了結果を通知します。	
Parameters:	
status	アダプタブル機能の有効・無効設定結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
cmd	アダプタブル機能有効・無効コマンド

8.3.15 RBLE_VS_EVENT_ADAPT_STATE_IND

RBLE_VS_EVENT_ADAPT_STATE_IND	
このイベントは、アダプタブル機能有効時、アダプタブル機能の状態変化を通知します。	
Parameters:	
state	アダプタブル機能状態

8.3.16 RBLE_VS_EVENT_COMMAND_DISALLOWED_IND

RBLE_VS_EVENT_COMMAND_DISALLOWED_IND		
このイベントは、VS コマンドが拒否されたことを通知します。		
Parameters:		
	status	コマンド実行結果 (3.2 rBLE ステータス列挙型宣言を参照ください)
	opcode	拒否されたコマンドのオペコード

8.3.17 RBLE_VS_EVENT_SET_TX_POWER_COMP

RBLE_VS_EVENT_SET_TX_POWER_COMP		
このイベントは、送信パワー設定完了を通知します。		
Parameters:		
	status	送信パワー設定結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

8.3.18 RBLE_VS_EVENT_SET_PARAMS_COMP

RBLE_VS_EVENT_SET_PARAMS_COMP		
このイベントは、任意のパラメータ設定完了を通知します。		
Parameters:		
	status	任意のパラメータ設定結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

8.3.19 RBLE_VS_EVENT_RF_CONTROL_COMP

RBLE_VS_EVENT_RF_CONTROL_COMP		
このイベントは、RF チップの電源制御完了を通知します。		
Parameters:		
	status	RF 電源制御結果 (3.2 rBLE ステータス列挙型宣言を参照ください)

9. RWKE

このセクションでは、RWKE(Renesas Wireless Kernel Extension)の API について記載します。

RWKE は、BLE プロトコルスタックを動作させるために設計された基本ソフトウェアで、疑似マルチタスク方式(ノンプリエンプティブルマルチタスク方式)の簡易オペレーティングシステムです。

9.1 型宣言

```
typedef  uint32_t  evt_field_t ;                          カーネルイベントフィールド
typedef  void  ( * evt_ptr_t ) ( void ) ;                 カーネルイベントハンドラ
typedef  uint16_t  ke_state_t ;                           タスク状態
typedef  uint16_t  ke_task_id_t ;                         タスク識別子
typedef  uint16_t  ke_msg_id_t ;                          メッセージ識別子
typedef  int  ( * ke_msg_func_t ) ( const                 メッセージハンドラ
ke_msg_id_t msg,
const void *param, const ke_task_id_t dst, const
ke_task_id_t src );
typedef  uint16_t  ke_time_t ;                            相対時刻（10ms 単位）
```

9.2 カーネルイベント管理機能

RWKE は、割り込みの遅延処理などを行なう手段として、カーネルイベント管理機能を提供しています。

RWKE は、常に実行されているループ（カーネルイベントループ）を持ち、ループするたびにカーネルイベントの発生を確認します。カーネルイベントが発生すると、RWKE は、対応するカーネルイベントハンドラを呼び出して処理してから、カーネルイベントループに戻ります。複数のカーネルイベントが同時に発生した場合、優先度の高いカーネルイベントが先に処理されます。

カーネルイベントは、0 から 31 までのカーネルイベント番号によってシステム内で一意に識別されます。カーネルイベント番号 0 の優先度が最も高く、31 の優先度が最も低くなります。カーネルイベント管理機能の API では、カーネルイベント番号ではなく、evt_field_t 型のカーネルイベントフィールドを使います。カーネルイベント番号 evt とカーネルイベントフィールド evt_field の間には、次の対応関係が成り立ちます。

evt_field ＝ (uint32_t) 1 << (31 — evt)

複数のカーネルイベントフィールドを指定する場合には、個々のカーネルイベントフィールドのビットごとの論理和を取ります。

表 9-1 カーネルイベント管理機能

RWKE API 名称	機能概要
ke_evt_get	カーネルイベントの設定状態を取得する
ke_evt_set	カーネルイベントをセットする
ke_evt_clear	カーネルイベントをクリアする

カーネルイベントハンドラは、evt_ptr_t 型の関数です。カーネルイベントハンドラが呼ばれたら、カーネルイベントに対する処理を行い、ke_evt_clear を呼んで、カーネルイベントをクリアします。クリアしない場合、クリアするまでカーネルイベントハンドラが呼ばれ続けますので、ご注意ください。

また、カーネルイベントは、計数タイプではありません。言い換えると、カーネルイベントがセットされ、対応するカーネルイベントハンドラが呼び出される前に、再度、同じカーネルイベントがセットされても、対

応するカーネルイベントハンドラは、一度しか呼ばれません。

9.2.1 ke_evt_get

evt_field_t ke_evt_get (void)	
カーネルイベントの設定状態を取得します。	
Parameters:	
none	
Return:	
カーネルイベントの設定状態が返ります。 evt_field_t 型の返り値の MSB から 1 ビットずつ順に、カーネルイベント番号 0、1、2、…、31 に対応し、ビットが 1 の場合には、カーネルイベントがセットされていることを、0 の場合には、カーネルイベントがクリアされていることを表します。	

9.2.2 ke_evt_set

void ke_evt_set (evt_field_t evt)	
evt で指定するカーネルイベントをセットします。	
Parameters:	
evt_field_t evt	セットするカーネルイベント 複数のイベントを指定する場合には、ビットごとの論理和を指定します。
Return:	
無し	

9.2.3 ke_evt_clear

void ke_evt_clear (evt_field_t evt)	
evt で指定するカーネルイベントをクリアします。	
Parameters:	
evt_field_t evt	クリアするカーネルイベント 複数のイベントを指定する場合には、ビットごとの論理和を指定します。
Return:	
無し	

9.3 メッセージ通信管理機能

RWKE は、タスク間やカーネルイベントハンドラとタスクの間の同期や通信の手段として、メッセージ通信管理機能を提供しています。

タスクがメッセージを送信すると、そのメッセージは、いったん、RWKE のカーネルメッセージキューにつながれます。その後、カーネルイベントハンドラの 1 つであるメッセージスケジューラによって、カーネルメッセージキューから取り出され、受信者タスクのメッセージハンドラに渡されます（メッセージへのポインタを引数として、受信者タスクのメッセージハンドラが呼び出されます）。

メッセージは、送信者のタスク識別子、受信者のタスク識別子、メッセージタイプ、メッセージ長などを保持するメッセージヘッダとメッセージ本体とで構成します。メッセージタイプは、送信者タスクと受信者タスクの間で一意に定めた、メッセージの種類です。

メッセージは、表 9-2 のような構造体として管理されます。②の部分がメッセージヘッダ、③の部分がメッセージ本体で、メッセージ本体は、実際には param_len バイトのサイズがあります。①の部分は、RWKE の管

理用の領域です。

表 9-2 メッセージ構造体

```
struct ke_msg
{
    struct co_list_hdr hdr;     ///< List header for chaining                    ①
#if (BLE_SPLIT || BLE_FULLEMB)
    uint8_t  hci_type;          ///< Type of HCI data(used by the HCI only)      ①
    int8_t  hci_off;            ///< Offset of the HCI data in the message (used ①
                                by the HCI only)
    uint16_t  hci_len;          ///< Length of the HCI traffic (used by the HCI only) ①
#endif
    ke_msg_id_t  id;            ///< Message id.                                 ②
    ke_task_id_t  dest_id;      ///< Destination kernel identifier.              ②
    ke_task_id_t  src_id;       ///< Source kernel identifier.                   ②
    uint16_t  param_len;        ///< Parameter embedded struct length.           ②
    uint32_t  param[1];         ///< Parameter embedded struct. Must be word-aligned. ③
};
```

表 9-3 メッセージ通信管理機能

RWKE API 名称	機能概要
ke_msg_alloc	メッセージ用のメモリブロックを確保します
ke_msg_free	メッセージ用のメモリブロックを解放します
ke_msg_send	メッセージを送信します
ke_msg_send_basic	空メッセージ（メッセージヘッダのみのメッセージ）を送信します
ke_msg_forward	メッセージを転送します
ke_msg2param	メッセージヘッダの先頭アドレスからメッセージ本体のアドレスを取得する
ke_param2msg	メッセージ本体の先頭アドレスからメッセージヘッダのアドレスを取得する

メッセージハンドラは、ke_msg_func_t 型の関数です。メッセージハンドラが呼ばれたら、渡されたメッセージに対する処理を行い、次のいずれかの値を返します。

KE_MSG_CONSUMED	渡されたメッセージを処理した。RWKEは、メッセージを削除(解放)します
KE_MSG_NO_FREE	渡されたメッセージを処理した。RWKEは、メッセージ削除(解放)しません
KE_MSG_SAVED	渡されたメッセージを処理しなかった。RWKEは、タスクの状態が変化した場合に、再度、メッセージをメッセージハンドラに渡します

RWKE では、タスクは、タスクディスクリプタと複数のメッセージハンドラで構成します。タスクディスクリプタは、タスクの状態、メッセージタイプとメッセージハンドラとを対応付ける情報で、メッセージスケジューラは、その時点の受信者タスクの状態とメッセージ中のメッセージタイプとをキーにして受信者タスクのタスクディスクリプタを検索し、適切なメッセージハンドラを選択して、メッセージを渡します。

9.3.1 ke_msg_alloc

void * ke_msg_alloc (ke_msg_id_t id, ke_task_id_t dest_id, ke_task_id_t src_id, uint16_t param_len)	
メッセージ用のメモリブロックを確保します	
Parameters:	
ke_msg_id_t id	送信するメッセージタイプ
ke_task_id_t dest_id	受信者のタスク識別子
ke_task_id_t src_id	送信者のタスク識別子
uint16_t param_len	メッセージ本体用に確保する領域のサイズ ※ke_malloc にて確保可能なサイズの範囲で設定してください。
Return:	
確保したメッセージ用メモリブロックの、メッセージ本体部分の先頭アドレス	

9.3.2 ke_msg_free

void ke_msg_free (const struct ke_msg *msg)	
ke_msg_alloc で確保したメッセージ用のメモリブロックを開放する	
Parameters:	
*const struct ke_msg *msg*	開放するメッセージ用メモリブロックの先頭アドレス
Return:	
無し	

9.3.3 ke_msg_send

void ke_msg_send (const void *param_ptr)	
param_ptr で指定するメッセージ本体を含むメッセージを送信する	
Parameters:	
*const void *param_ptr*	送信するメッセージ本体の先頭アドレス
Return:	
無し	

9.3.4 ke_msg_send_basic

void ke_msg_send_basic (ke_msg_id_t id, ke_task_id_t dest_id, ke_task_id_t src_id)	
空メッセージ（メッセージヘッダのみのメッセージ）を送信します	
Parameters:	
ke_msg_id_t id	送信するメッセージタイプ
ke_task_id_t dest_id	受信者のタスク識別子
ke_task_id_t src_id	送信者のタスク識別子
Return:	
無し	

9.3.5 ke_msg_forward

void ke_msg_forward (const void *param_ptr, ke_task_id_t dest_id, ke_task_id src_id)	
param_ptr で指定するメッセージ本体を含むメッセージを転送します	
Parameters:	
*const void *param_ptr*	転送するメッセージのメッセージ本体の先頭アドレス
ke_task_id_t dest_id	転送先のタスク識別子
ke_task_id_t src_id	転送元のタスク識別子
Return:	
無し	

9.3.6 ke_msg2param

void * ke_msg2param (const struct ke_msg *msg)	
param_ptr で指定するメッセージの先頭アドレスからメッセージ本体の先頭アドレスを計算する	
Parameters:	
*const struct ke_msg *msg*	メッセージの先頭アドレス
Return:	
メッセージ本体の先頭アドレス	

9.3.7 ke_param2msg

struct ke_msg * ke_param2msg (const void *param_ptr)	
param_ptr で指定するメッセージ本体の先頭アドレスからメッセージの先頭アドレスを計算する	
Parameters:	
*const void *param_ptr*	メッセージ本体の先頭アドレス
Return:	
メッセージの先頭アドレス	

9.4 タスク状態管理機能

タスクは、0～63 までのタスクタイプによってシステム内で一意に識別されます。また、各タスクは、0～63 までのタスクインデクスを持つことができ、マルチインスタンス化することができます。タスクのそれぞれのインスタンスは、ke_task_id_t 型を持つタスク識別子によってシステム内で一意に識別されます。

タスクタイプ type、タスクインデクス idx とタスク識別子 task_id との間には、次の関係があります。

task_id = (idx << 8) | type

通常は、タスクインデクスを 0 にして使用してください。

タスクのそれぞれのインスタンスは、ke_state_t 型を持つ状態と呼ぶ変数を 1 つ、管理します。状態の値は、タスクのインスタンスごとに意味付けが異なります。また、システムの初期化直後は、通常、0 です。

RWKE は、タスクが状態（ステート）を管理する手段として、タスク状態管理機能を提供しています。

表 9-4 タスク状態管理機能

RWKE API 名称	機能概要
ke_state_get	タスク状態を参照する
ke_state_set	タスク状態を設定（変更）する

9.4.1 ke_state_get

ke_state_t ke_state_get (const ke_task_id_t task)		
task で指定したタスクの状態を取得する		
Parameters:		
	const ke_task_id_t task	状態を取得するタスクのタスク識別子
Return:		
	タスクの状態が返ります	

9.4.2 ke_state_set

void ke_state_set (const ke_task_id_t task, const ke_state_t state)		
task で指定したタスクの状態を state に設定する ※指定可能なタスクはユーザタスクに限られます。ユーザタスク以外を指定した場合の動作は保証しません。		
Parameters:		
	const ke_task_id_t task	状態を設定するタスクのタスク識別子
	const ke_state_t state	設定する状態の値
Return:		
	無し	

9.5 タイマ管理機能

RWKE は、時間に依存した処理を行なうための手段として、タイマ管理機能を提供しています。

RWKE が提供するタイマ管理機能は、指定した時刻に指定したタスク宛てに空メッセージを送る機能で、実際の処理は、空メッセージを受信したタスクのメッセージハンドラで行ないます。

タスクがタイマを設定すると、タイマ要求ブロックが作られ、RWKE のカーネルタイマキューにつながれます。その後、指定した時刻になると、カーネルイベントハンドラの 1 つであるタイマスケジューラによって、タイマ要求ブロックがカーネルタイマキューから取り出されて、指定されたタスクに空メッセージが送られます。

表 9-5 タイマ管理機能

RWKE API 名称	機能概要
ke_time	現在のタイマ値を取得する
ke_timer_set	タイマを設定する
ke_timer_clear	設定したタイマを取り消す

9.5.1 ke_time

ke_time_t ke_time (void)		
現在のタイマ値を取得する		
Parameters:		
	無し	
Return:		
	現在のタイマ値（10ms 単位）	

9.5.2 ke_timer_set

void ke_timer_set (ke_msg_id_t timerid, ke_task_id_t task, ke_time_t delay)		
タイマを設定する。delay で指定した時間経過後に、task で指定したタスクにメッセージタイプ timerid の空メッセージが送られます		
Parameters:		
	ke_msg_id_t timerid	指定時間経過後に送るメッセージのメッセージタイプ
	ke_task_id task	指定時間経過後に送るメッセージの受信者タスク
	ke_time_t delay	時間(10ms 単位) ※1～29999 の範囲で設定が可能です。
Return:		
	無し	

9.5.3 ke_timer_clear

void ke_timer_clear (ke_msg_id_t timerid, ke_task_id_t task)		
設定したタイマを取り消します。		
Parameters:		
	ke_msg_id_t timerid	設定したタイマのメッセージタイプ
	ke_task_id_t task	設定したタイマの受信者タスク
Return:		
	無し	

9.6 メモリ管理機能

RWKE は、動的なメモリ管理を行なう手段として、メモリ管理機能を提供しています。

ヒープ領域は、RAM 上に確保された単一の連続領域で、ヒープ領域の先頭アドレスは、ke_mem_heap、最終アドレス＋1 は、ke_mem_heap_end で、それぞれ示されます。

```
extern uint8_t  ke_mem_heap[];
extern uint8_t  ke_mem_heap_end[];
```

ヒープから確保されるメモリブロックの先頭は、2 バイト境界で整列されています。

表 9-6 メモリ管理機能

RWKE API 名称	機能概要
ke_malloc	メモリブロックを確保する
ke_free	メモリブロックを解放する

9.6.1 ke_malloc

void * ke_malloc (size_t size)
ヒープ領域から size で指定したサイズのメモリブロックを確保する

void * ke_malloc (size_t size)		
Parameters:		
	size_t size	確保するメモリブロックのサイズ ※BLE_HEAP_SIZE にユーザアプリ用として確保したサイズ分を上限としてください。
Return:		
	確保したメモリブロックの先頭アドレス	

9.6.2 ke_free

void ke_free (void *mem_ptr)		
ke_malloc で確保したメモリブロックを開放する		
Parameters:		
	*void *mem_ptr*	開放するメモリブロックの先頭アドレス
Return:		
	戻り値はありません	

9.7 排他制御機能

RWKE は、メイン処理（メッセージハンドラやイベントハンドラなど）と割り込み処理との間の排他制御の手段として、割り込み禁止を使用した排他制御機能を提供しています。

割り込み禁止は、RL78/G1D の PSW の IE ビットを制御して行なっており、PSW の ISP0、ISP1 ビットは変更しません。また、割り込み処理の実行には、RWKE は介在しません。

表 9-7 排他制御機能

RWKE API 名称	機能概要
GLOBAL_INT_START	割り込み許可にします
GLOBAL_INT_STOP	割り込み禁止にします
GLOBAL_INT_DISABLE	割り込み禁止状態（許可/禁止）を保存し、割り込み禁止にします
GLOBAL_INT_RESTORE	割り込み禁止状態を復元します

注 1） GLOBAL_INT_START、GLOBAL_INT_STOP は、マクロです。
これらは、割り込み禁止状態（許可/禁止）が明確に解っている場合にのみ、ご使用ください。

注 2） GLOBAL_INT_DISABLE、GLOBAL_INT_RESTORE は、マクロです。
これらは、必ず、同一関数内で対にして使用します。ネストすることができます。

9.8 初期化とイベントループの実行

RWKE の機能を使用する前に、ke_init 関数を呼んで RWKE の初期化を行ないます。

また、応用システムの初期化が終了したら、その後、ループ処理を行ない、ke_evt_schedule 関数を繰り返し呼び続けます。

RWKE を実行する場合の簡略化した main 関数は、次のようになります。

```
void main(void)
{
    ke_init();                    // RWKE の初期化
    GLOBAL_INT_START();
```

```
        for ( ; ; ) {                   // RWKE のイベントループの実行
              ke_evt_schedule();        //
        }                               //
    }
```

表 9-8 初期化とイベントループの実行

関数名称	機能概要
ke_init	RWKE の初期化を行ないます
ke_evt_schedule	RWKE のカーネルイベントループの処理を 1 回、実行します

9.9 割り込み処理から利用可能な RWKE の API

割り込み処理（マスカブル割り込みに限る）から利用可能な RWKE の API は、次の通りです。
割り込み処理からこれら以外の RWKE API を呼び出した場合、その動作は保証しません。

ke_evt_set	ke_evt_clear	
ke_msg_alloc	ke_msg_send	ke_msg_send_basic

10. 注意事項

付録A　Message Sequence Chart

A.1　Initialization of BLE S/W

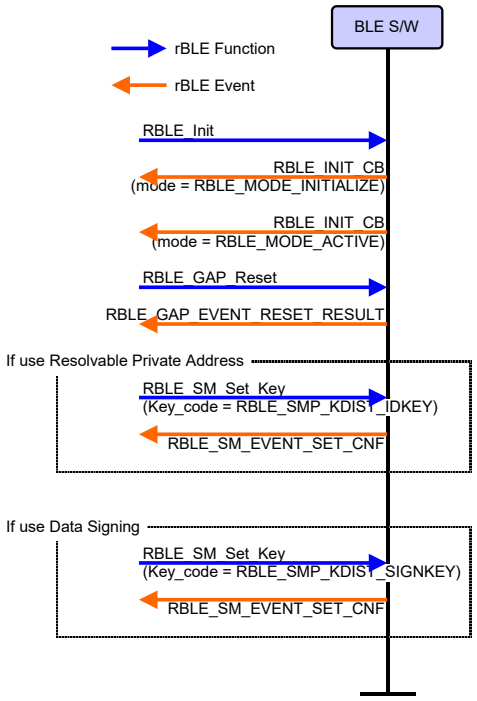


図 A-1　Initialization of BLE S/W

A.2　Broadcast Mode & Observation Procedure

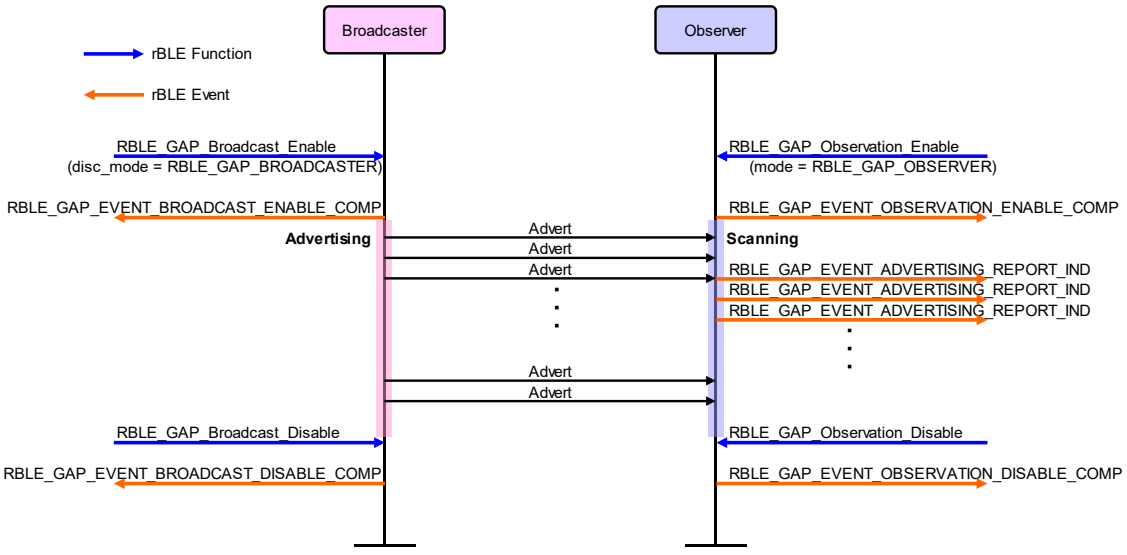


図 A-2　Broadcast Mode & Observation Procedure

A. 3　General Discoverable Mode

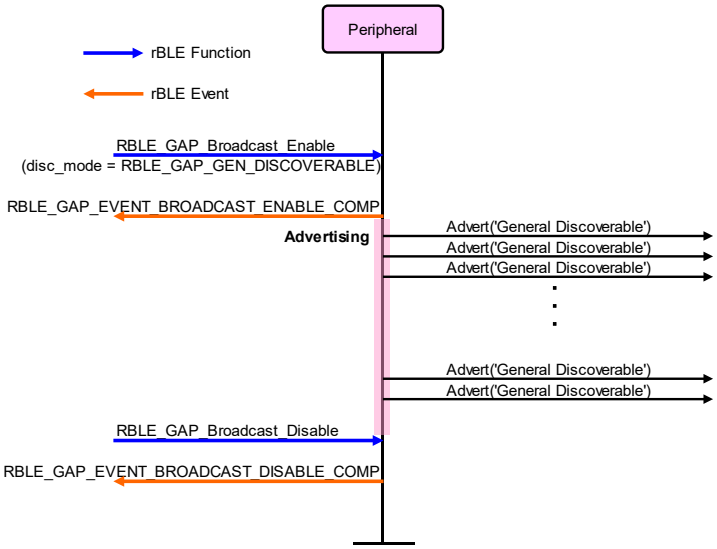


図 A-3　General Discoverable Mode

A. 4　General Discovery Procedure

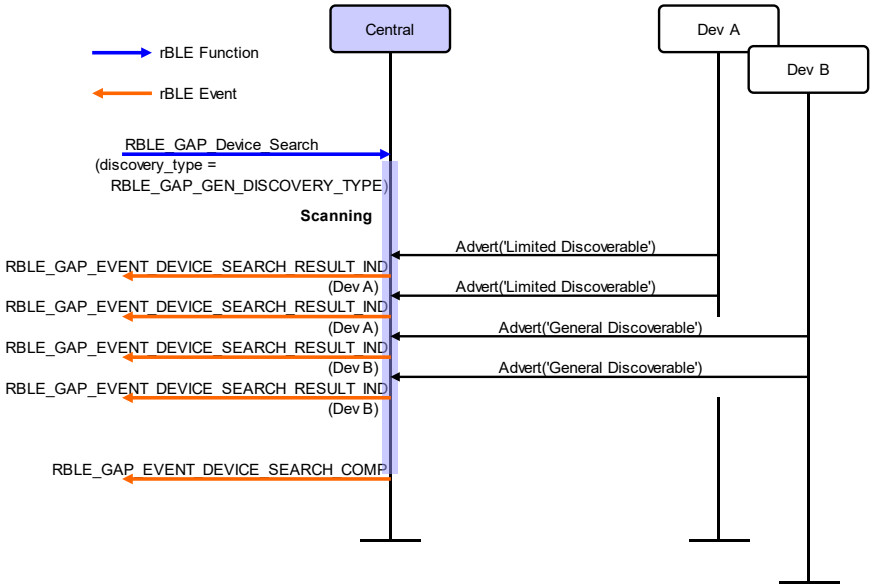


図 A-4　General Discovery Procedure

A. 5　Limited Discovery Procedure

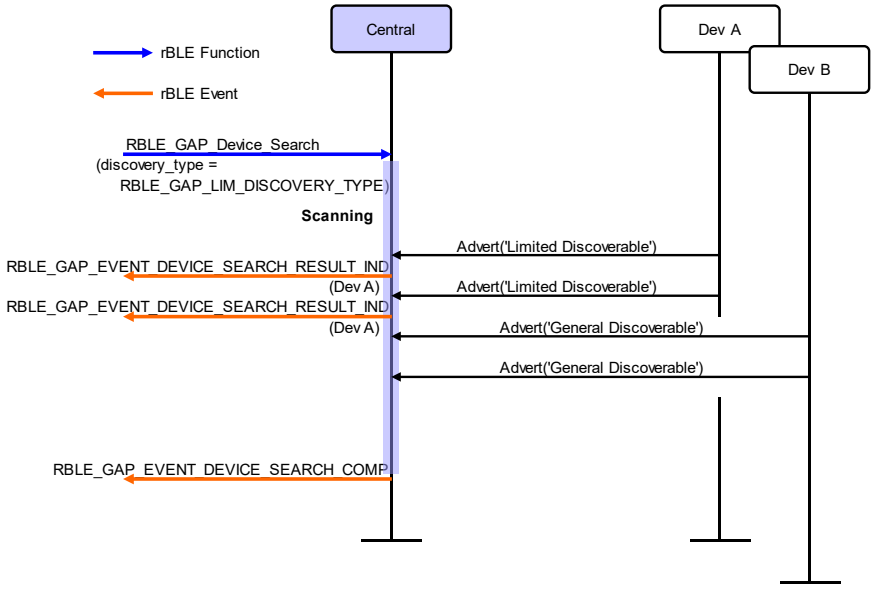


図 A-5　Limited Discovery Procedure

A. 6　Name Discovery Procedure (Non-connected state)

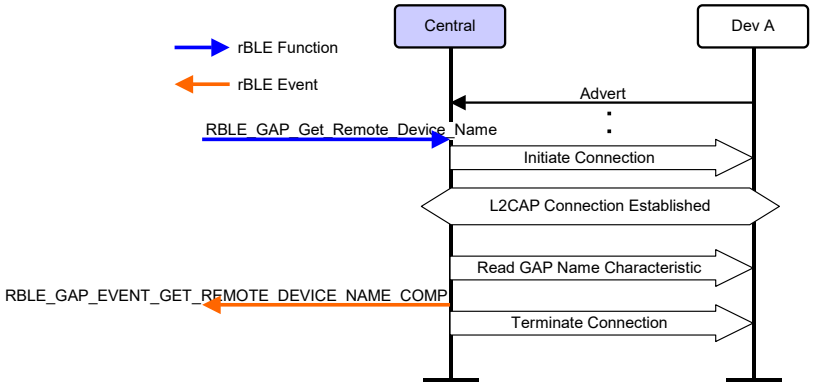


図 A-6　Name Discovery Procedure (Non-connected state)

A. 7　Name Discovery Procedure (Connected state)

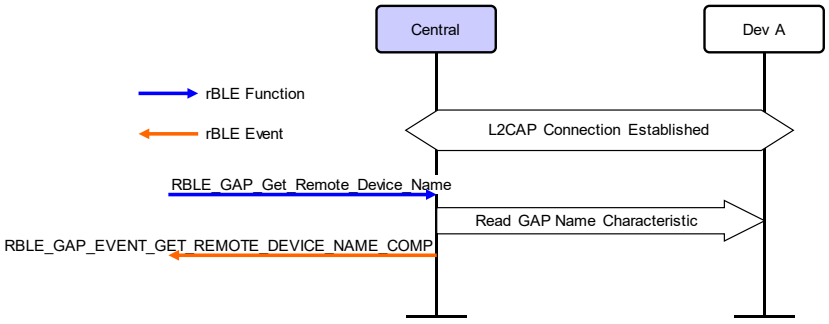


図 A-7　Name Discovery Procedure (Connected state)

A. 8　General Connection Establishment Procedure

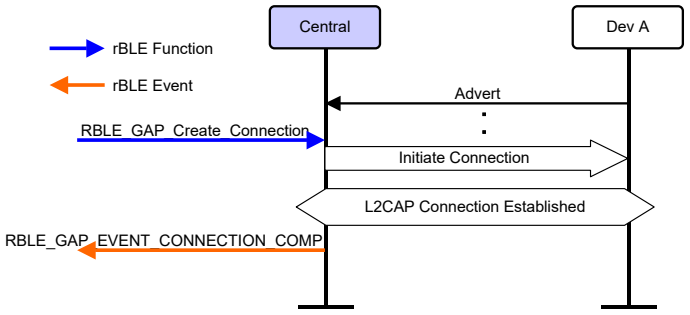


図 A-8　General Connection Establishment Procedure

A. 9　Terminate Connection Procedure

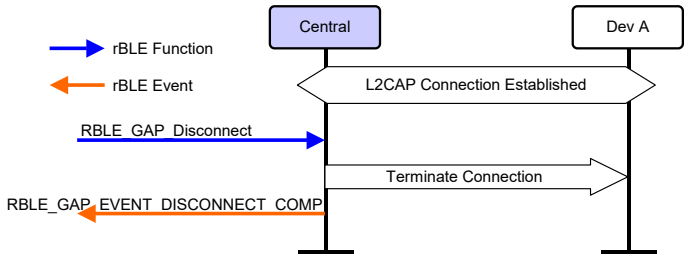


図 A-9　Terminate Connection Procedure

A. 10　Auto Connection Establishment Procedure

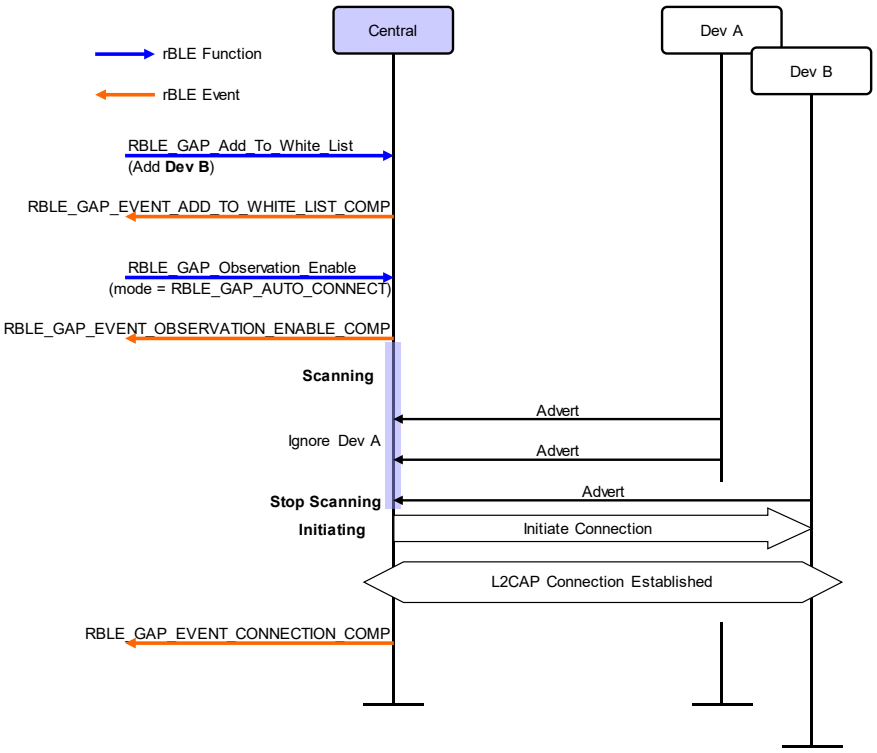


図 A-10　Auto Connection Establishment Procedure

A. 11　Connection Parameter Update Procedure - Central initiate

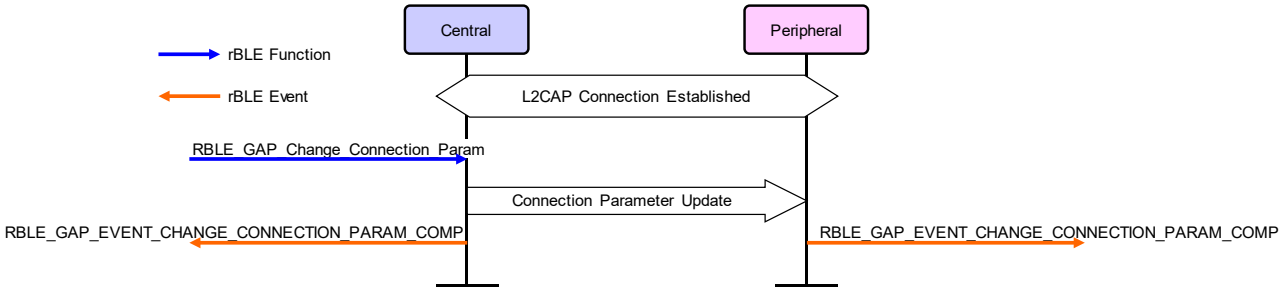
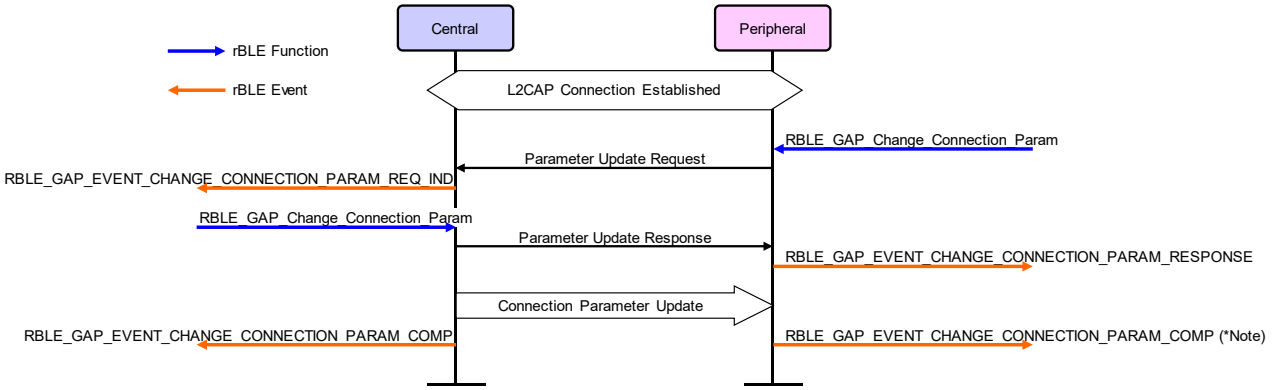


図 A-11　Connection Parameter Update Procedure - Central initiate

A. 12　Connection Parameter Update Procedure - Peripheral request



*Note: 接続パラメータが変更された場合に発生します。RBLE_GAP_Change_Connection_Param で通信中のパラメータと同じ値を設定した場合は発生しません。

図 A-12　Connection Parameter Update Procedure - Peripheral request

A. 13　Bonding Procedure - Central Initiate

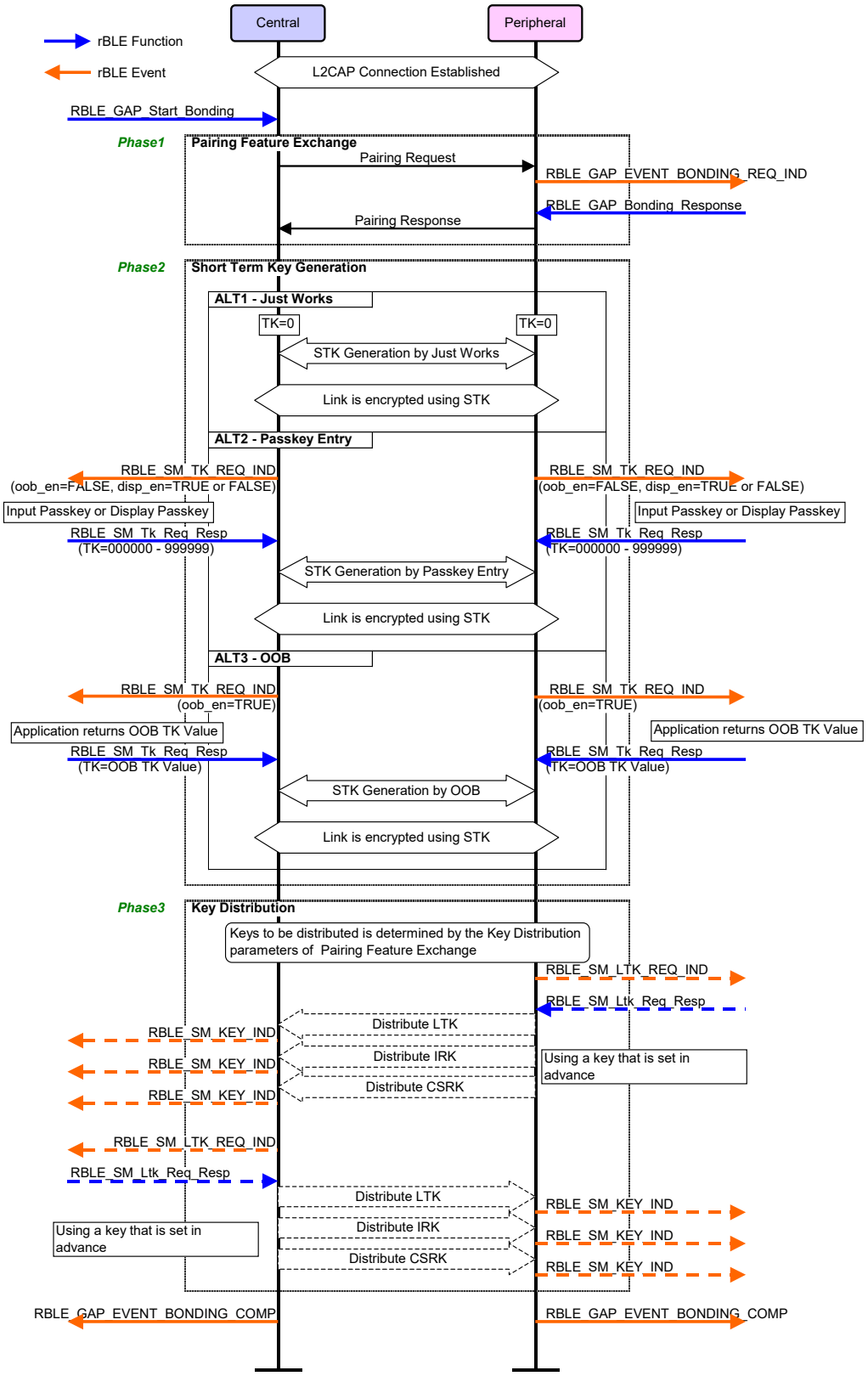


図 A-13　Bonding Procedure - Central Initiate

A. 14　Bonding Procedure - Peripheral Request

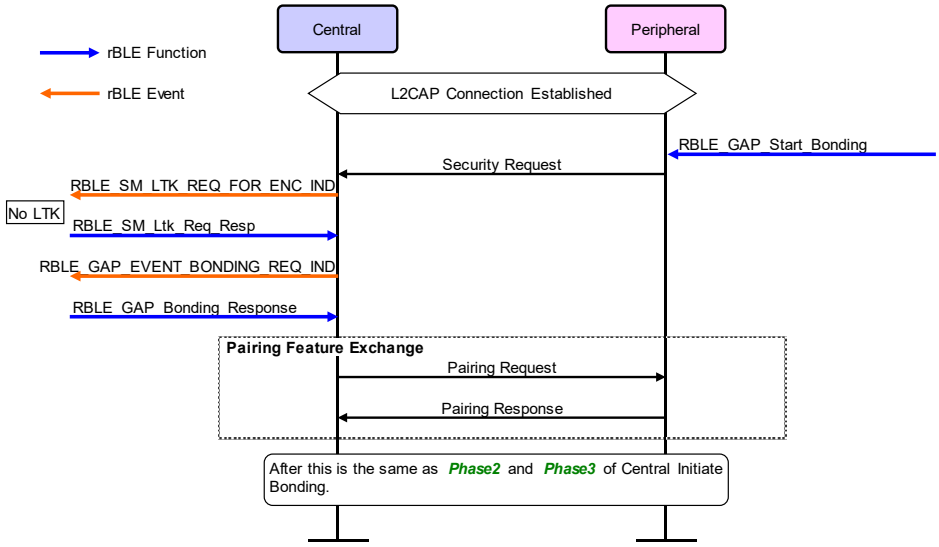


図 A-14　Bonding Procedure - Peripheral Request

A. 15　Bonding Procedure - Central Initiate, Peripheral Reject

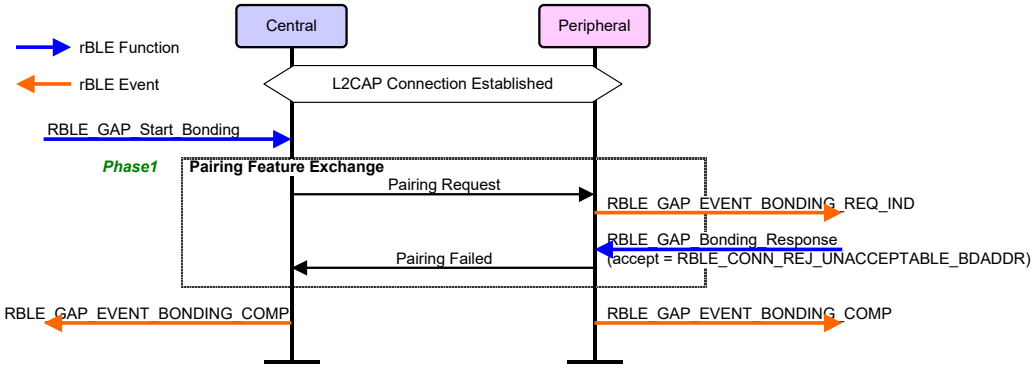


図 A-15　Bonding Procedure - Central Initiate, Peripheral Reject

A. 16　Bonding Procedure - Peripheral Request, Central Reject

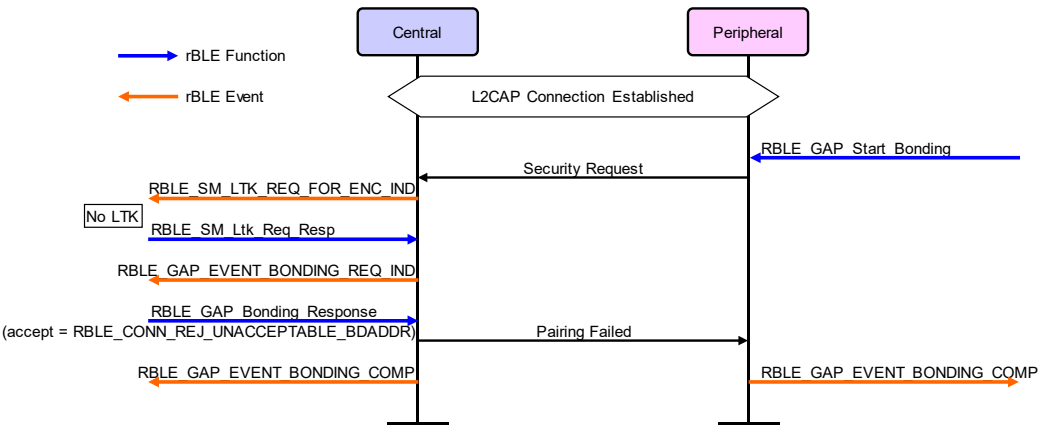


図 A-16　Bonding Procedure - Peripheral Request, Central Reject

A. 17　Central Initiated Link Layer Encryption

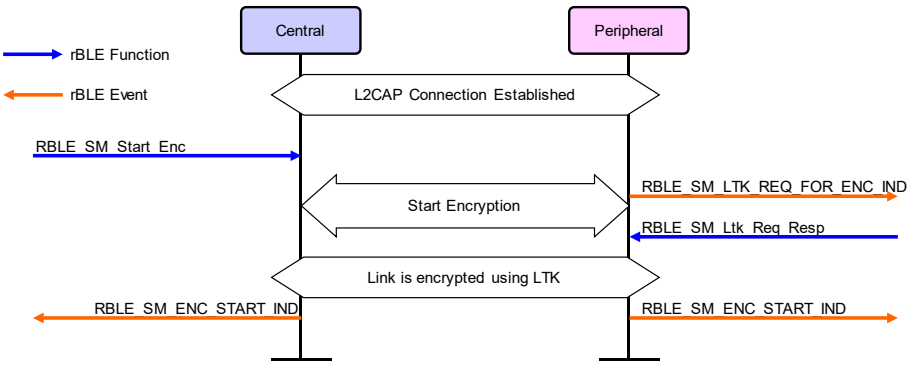


図 A-17　Central Initiated Link Layer Encryption

A. 18　Peripheral request, Central Initiated Link Layer Encryption

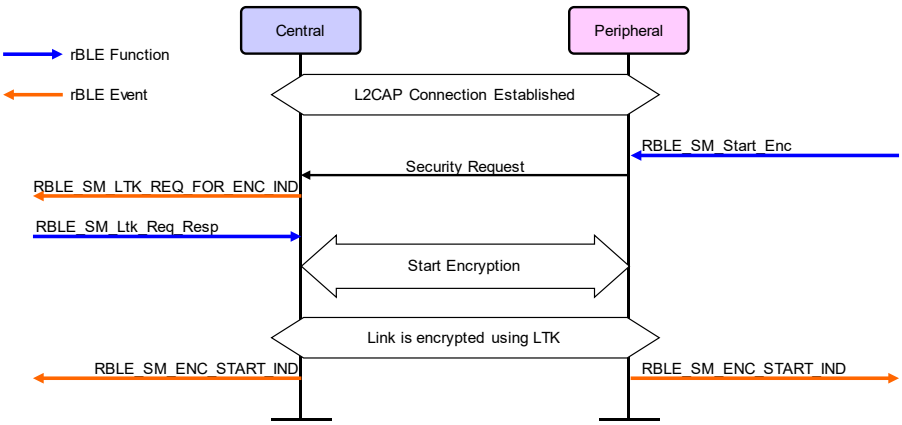


図 A-18　Peripheral request, Central Initiated Link Layer Encryption

A. 19　GATT Discover All Primary Services

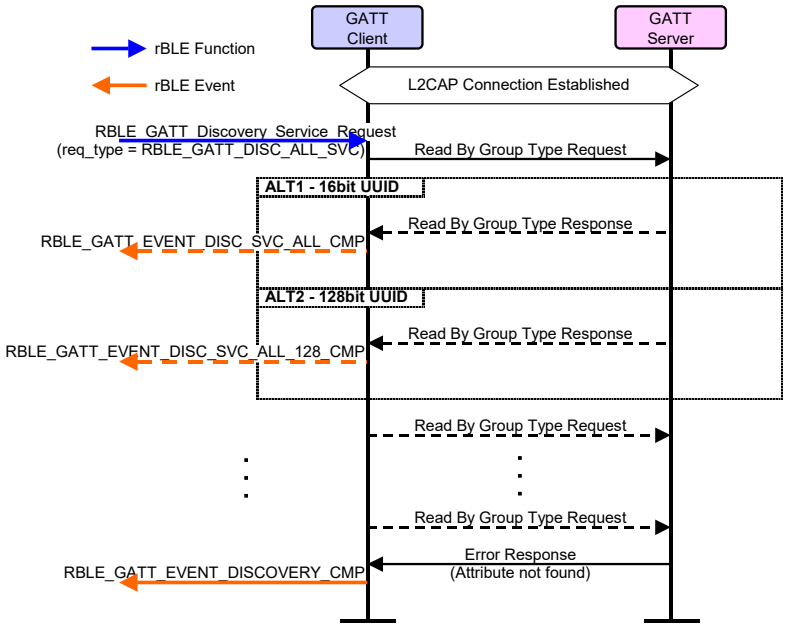


図 A-19　Discover All Primary Services

A. 20　GATT Discover Primary Services by UUID

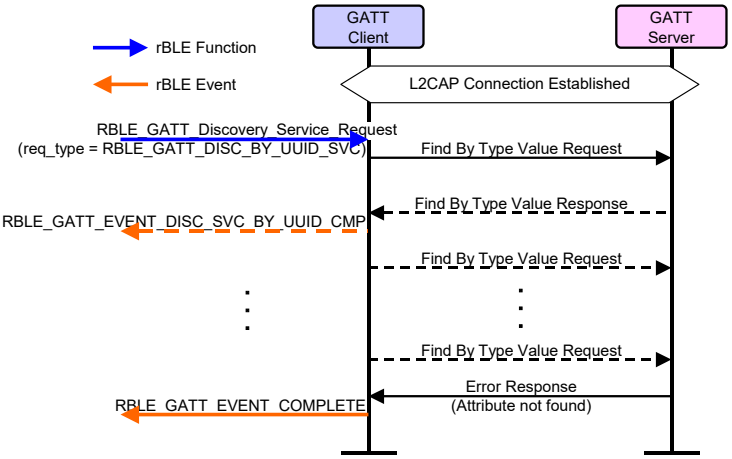


図 A-20　Discover Primary Services by UUID

A. 21　GATT Discover Included Services

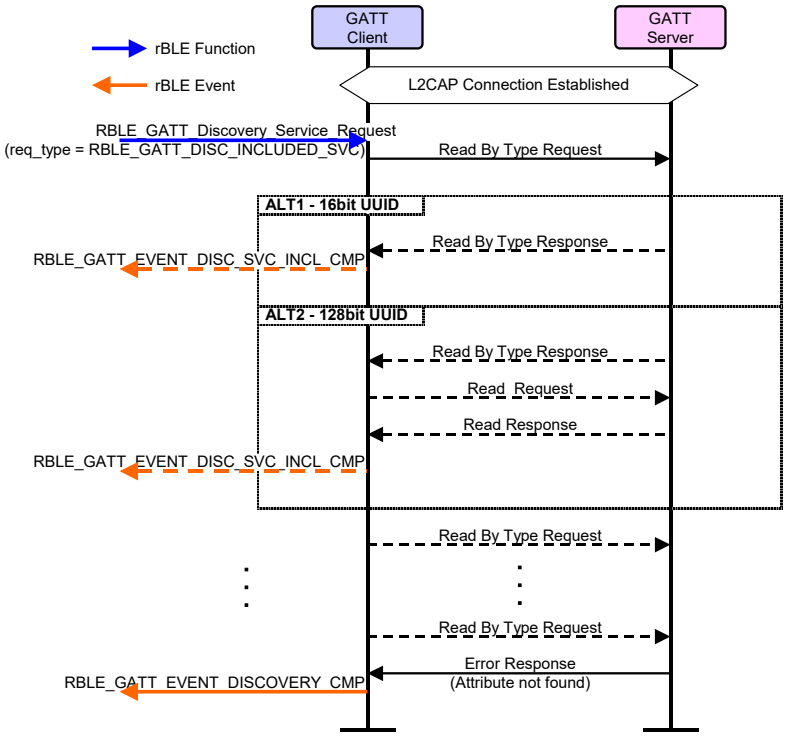


図 A-21　Discover Included Services

A. 22　GATT Discover All Characteristics

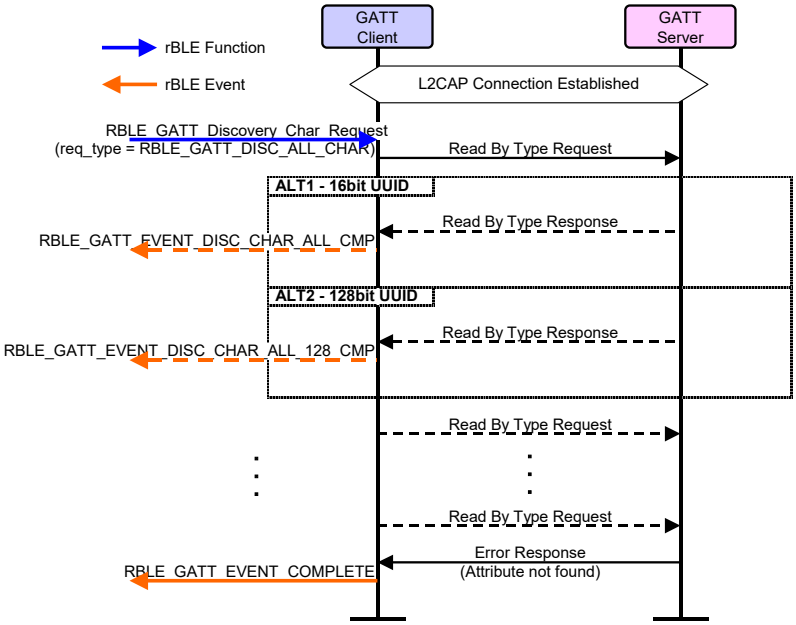


図 A-22　Discover All Characteristics

A. 23　GATT Discover Characteristics by UUID

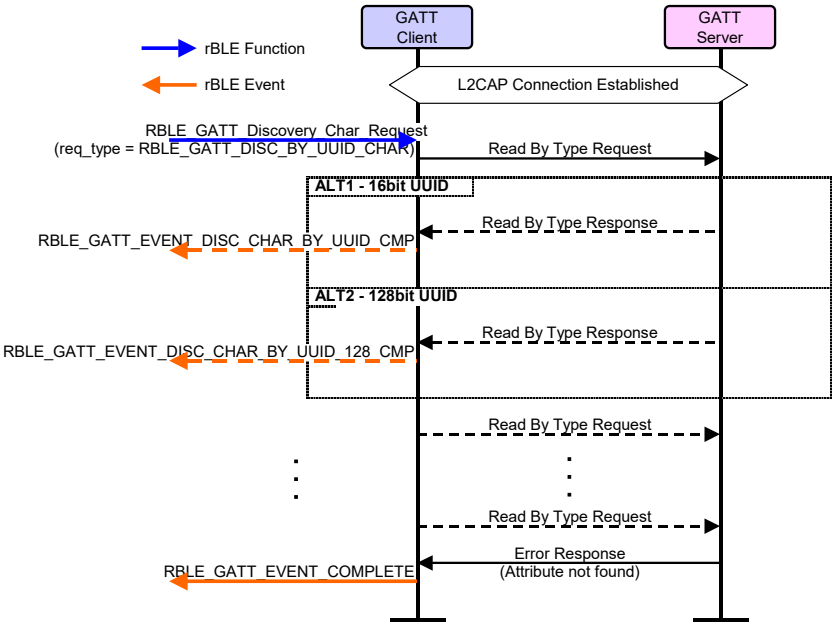


図 A-23　Discover Characteristics by UUID

A. 24　GATT Read Characteristic Value

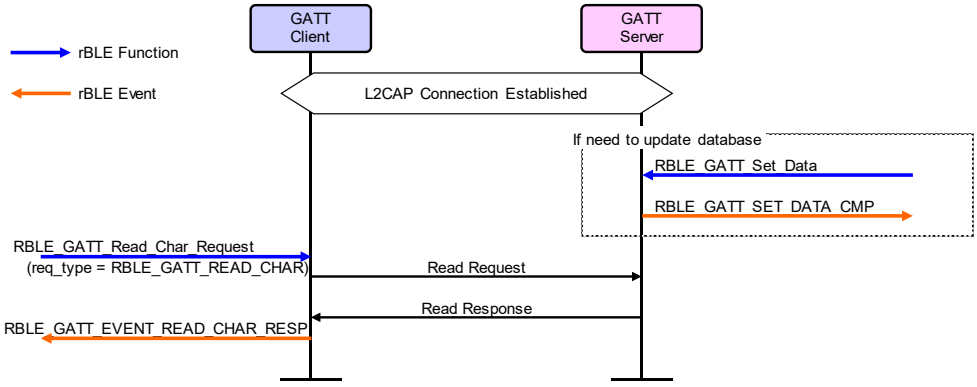


図 A-24　Read Characteristic Value

A. 25　GATT Read Using Characteristic UUID

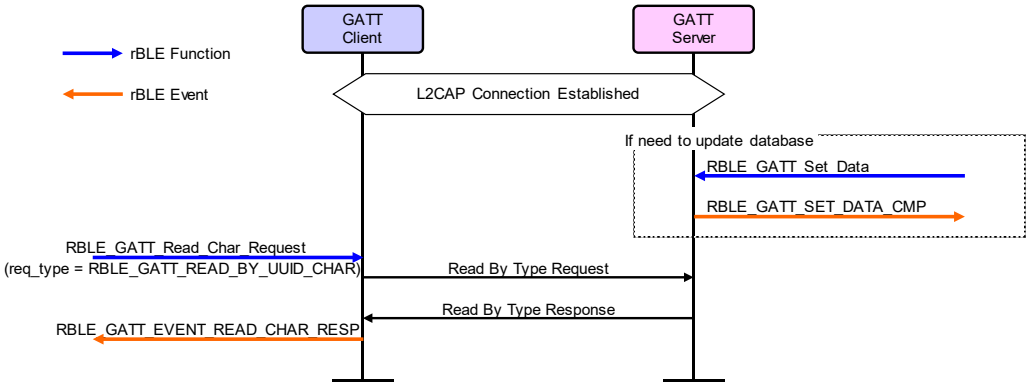


図 A-25　Read Using Characteristic UUID

A. 26　GATT Read Long Characteristic Values

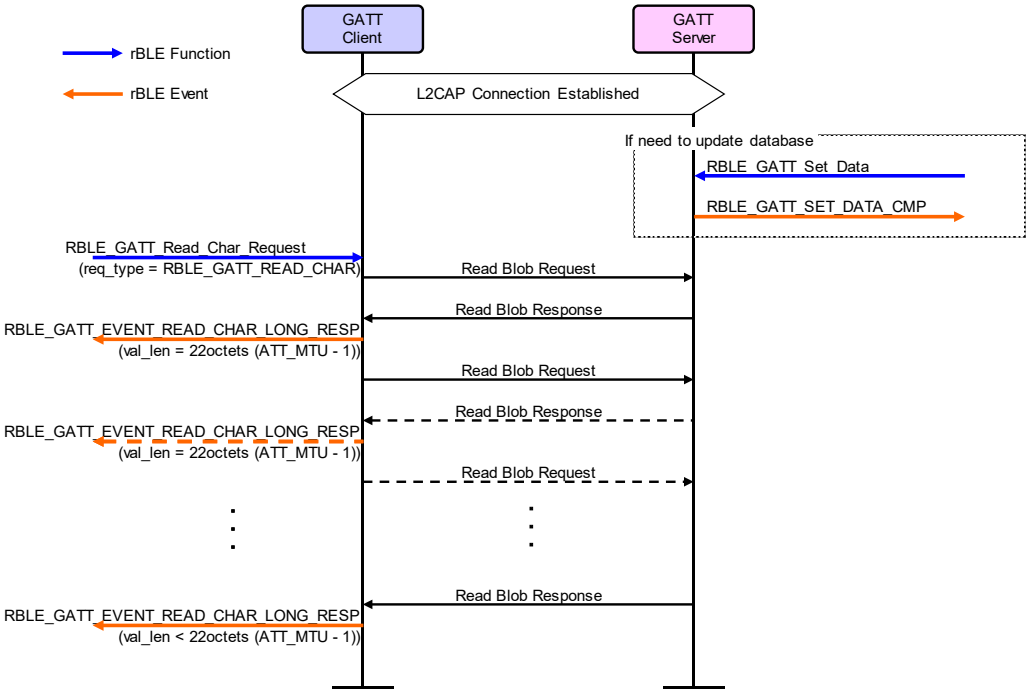


図 A-26　Read Long Characteristic Values

A. 27　GATT Read Multiple Characteristic Values

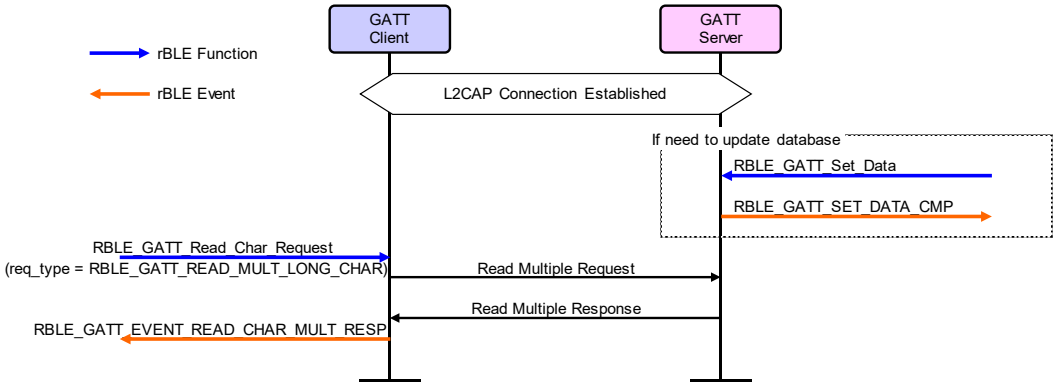


図 A-27　Read Multiple Characteristic Values

A. 28　GATT Read Characteristic Descriptors

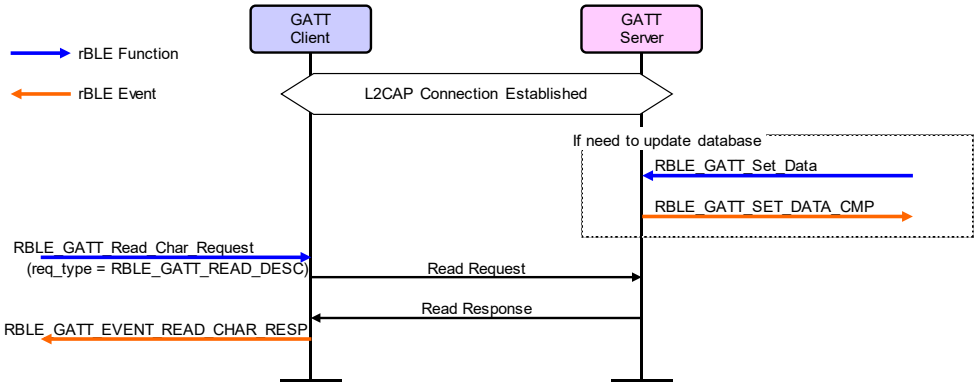


図 A-28　Read Characteristic Descriptors

A. 29　GATT Read Long Characteristic Descriptors

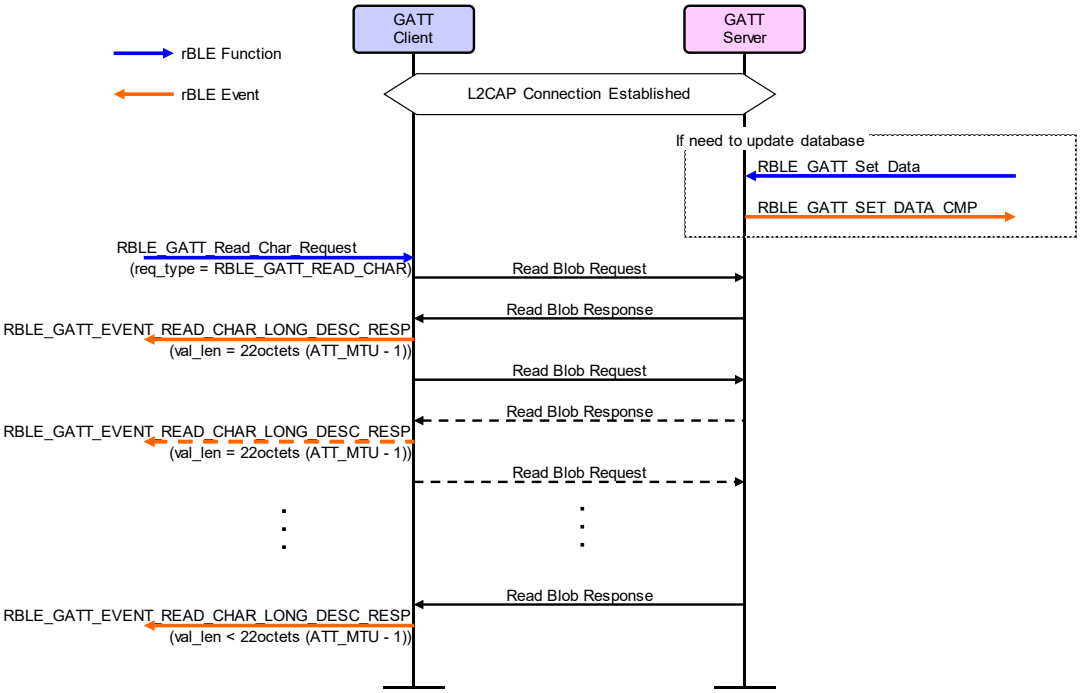


図 A-29　Read Long Characteristic Descriptors

A. 30　GATT Write Without Response

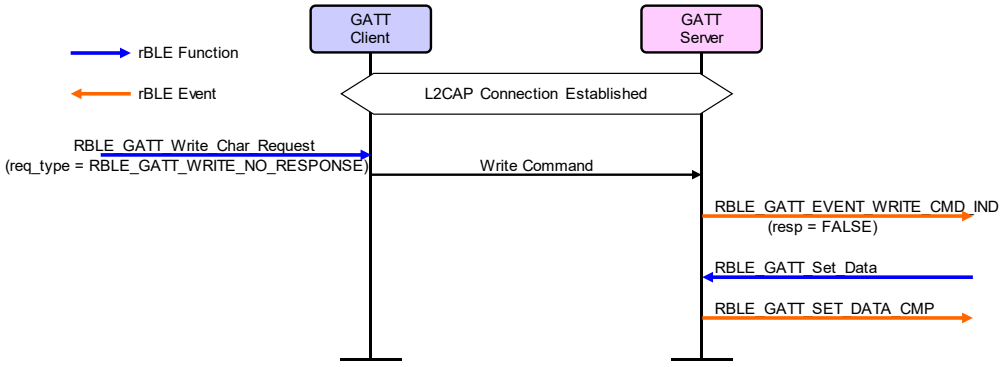


図 A-30　Write Without Response

A. 31　GATT Signed Write Without Response

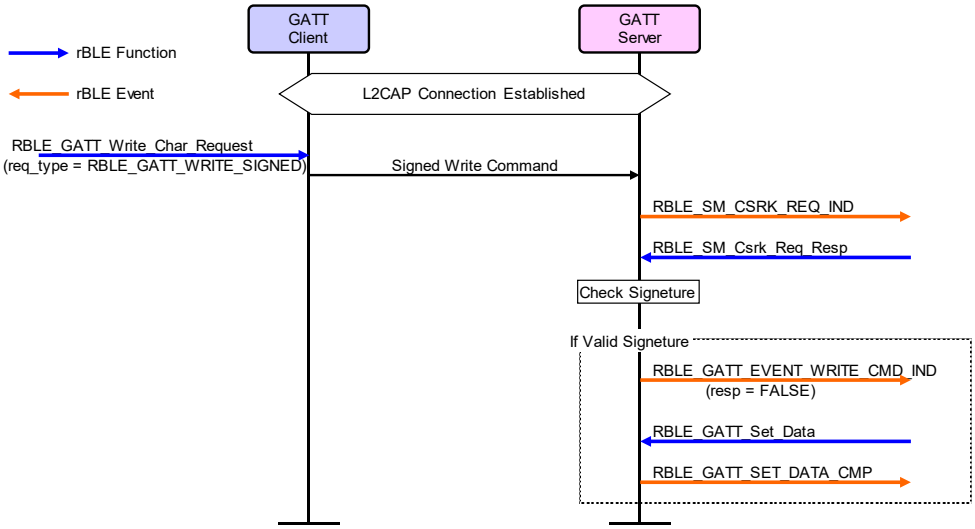


図 A-31　Signed Write Without Response

A. 32　GATT Write Characteristic Value / Write Characteristic Descriptor

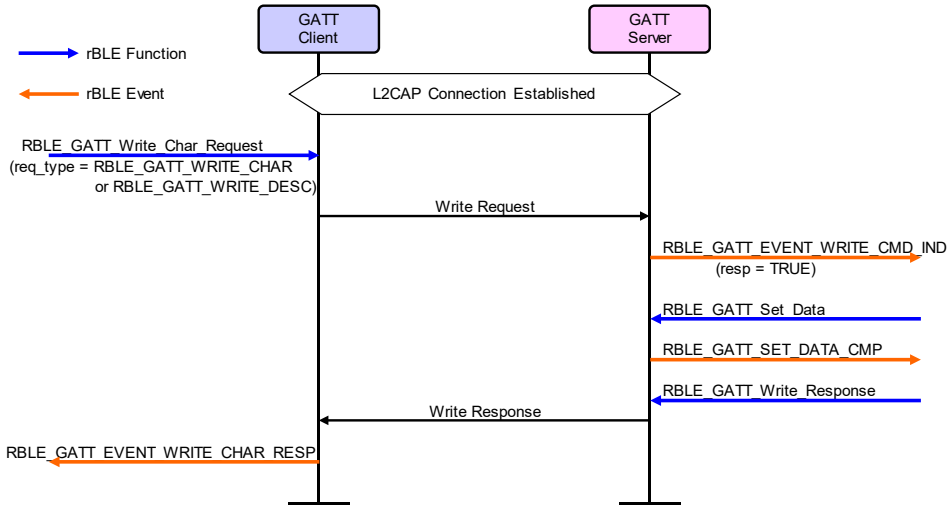


図 A-32　Write Characteristic Value / Write Characteristic Descriptor

A. 33　GATT Write Long Characteristic Value / Write Long Characteristic Descriptor

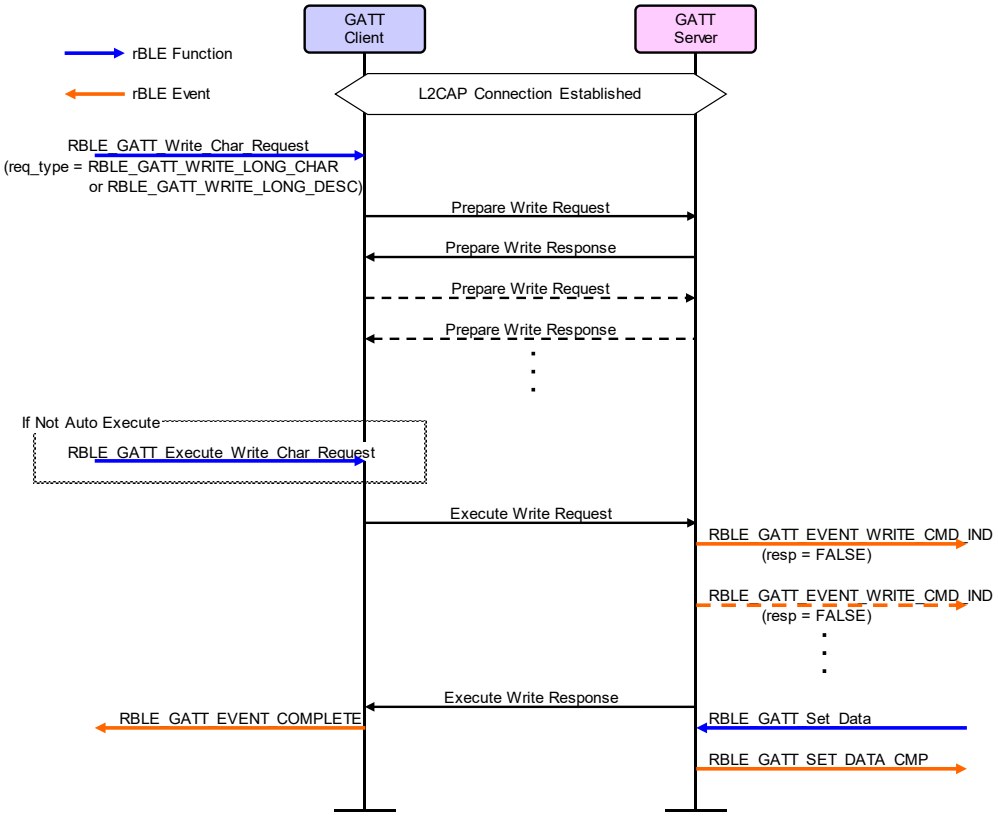


図 A-33　Write Long Characteristic Value / Write Long Characteristic Descriptor

A. 34　GATT Reliable Writes

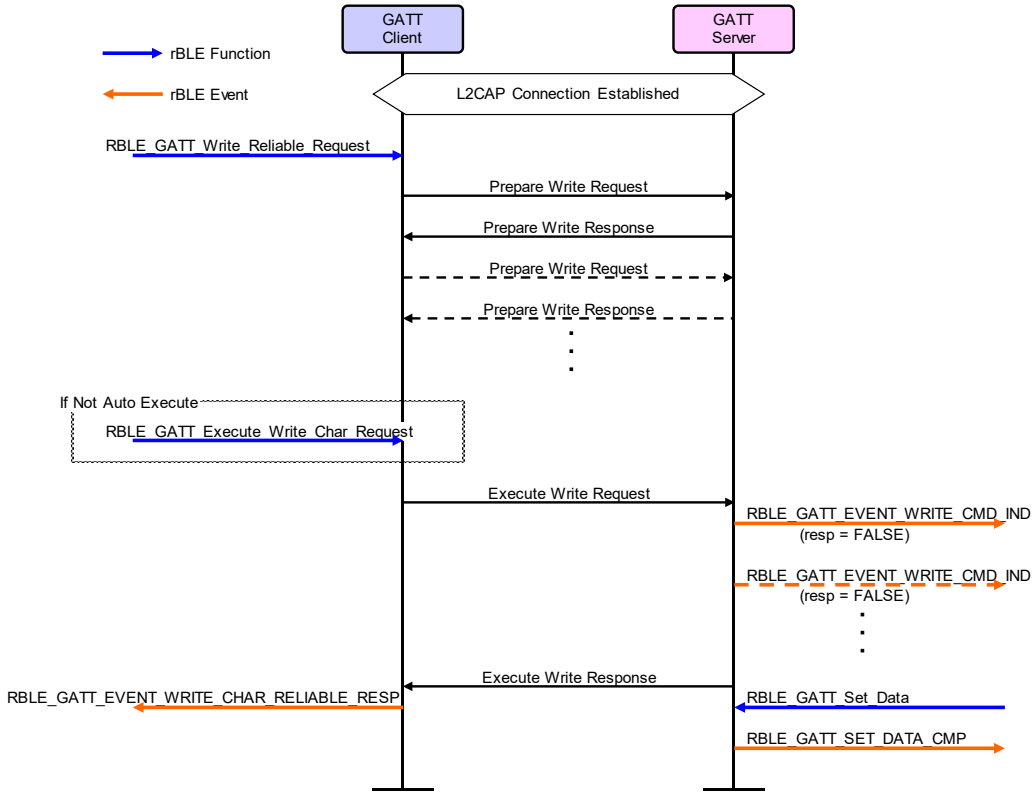


図 A-34　Reliable Writes

A. 35　GATT Notifications

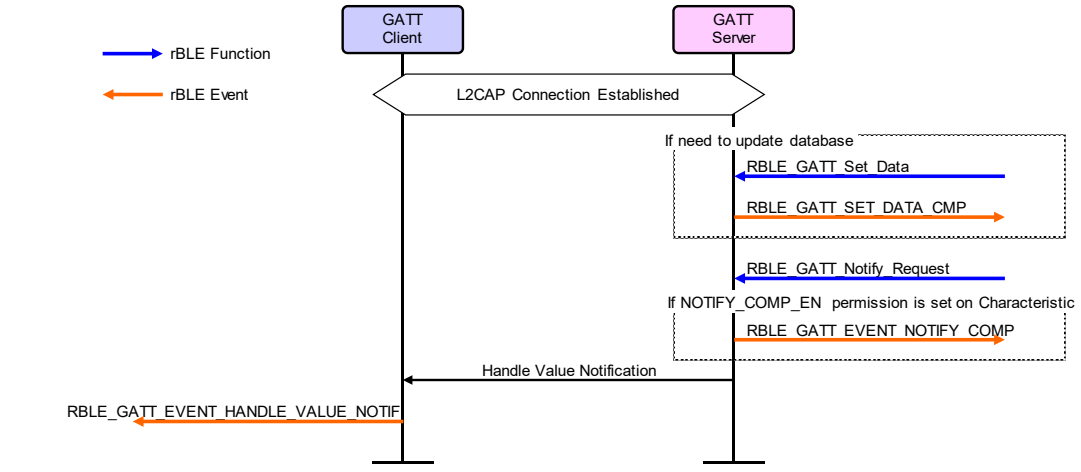


図 A-35　Notifications

A. 36　GATT Indications

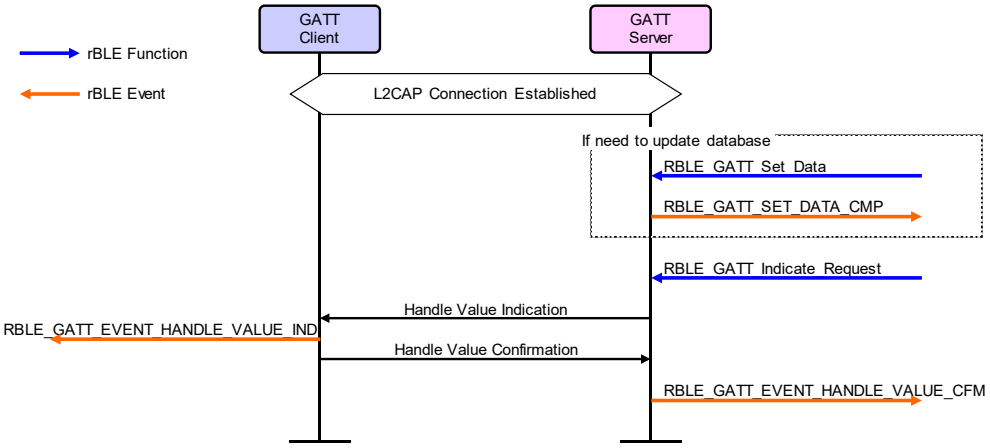


図 A-36　Indications

A. 37　Receiver Test

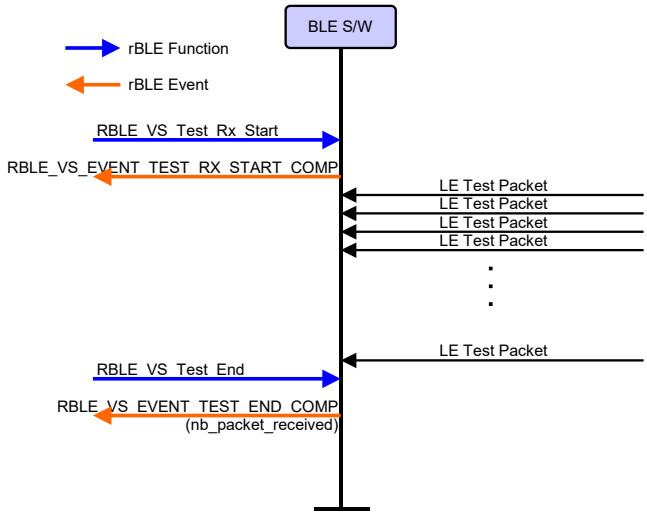


図 A-37　Receiver Test

A. 38　Transmitter Test

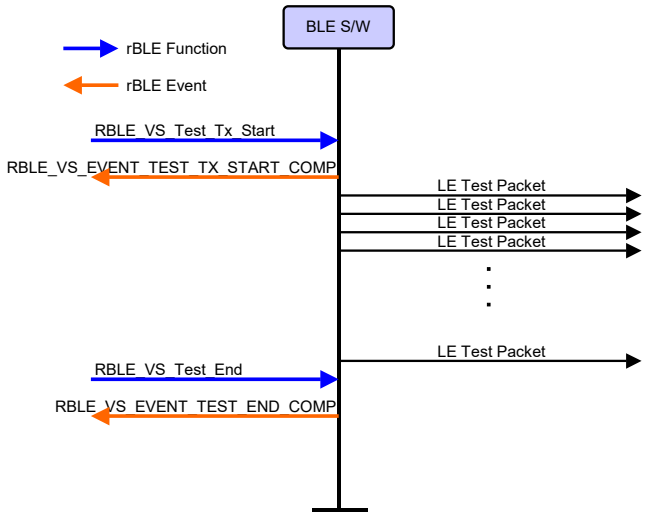


図 A-38　Transmitter Test

A. 39　Extended Receiver Test

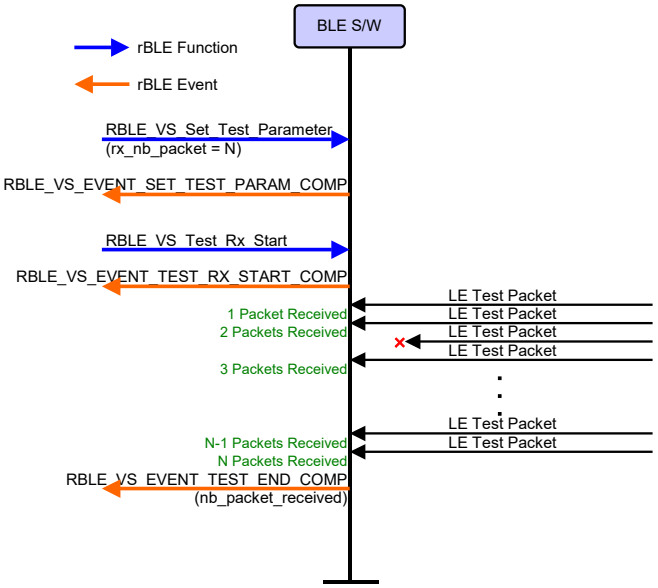


図 A-39　Extended Receiver Test

A. 40　Read RSSI during Receiver Test

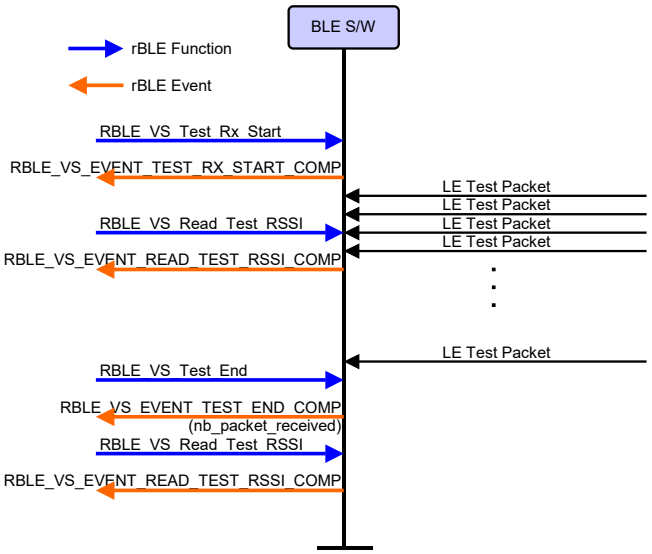


図 A-40　Read RSSI during Receiver Test

付録B　表の見方

本付録では、rBLE API の関数およびイベントを定義する表の見方について説明します。

B.1　関数定義表の見方

以下に、関数定義表に記載している内容について示します。

Parameters エリアはこの関数の引数について説明しています。
先頭列の斜体は関数の引数を意味します。各変数の最終列にてその引数の説明をしています。

引数の次列が斜体の場合は、引数(構造体)のメンバであることを示します。

引数名と引数説明の間に、その引数の取り得る値について説明している場合があります。

背景色が緑の表は、関数定義を意味します。このエリアには関数プロトタイプを記載しています。				
このエリアでは、関数の説明とこの関数実行後に通知されるイベントについて説明しています。				
Parameters:				
	引数 1	引数 1 の説明です。		
	引数 2	*メンバ 1*	メンバ 1 の取り得る値 1	メンバ 1 の取り得る値 1 の説明です。
			メンバ 1 の取り得る値 2	メンバ 1 の取り得る値 2 の説明です。
		メンバ 2	メンバ 2 の説明です。	
Return:				
	戻り値として取り得る値 1	戻り値として取り得る値 1 の説明です。		
	戻り値として取り得る値 2	戻り値として取り得る値 2 の説明です。		

Return エリアは関数の戻り値について説明しています。
先頭列は戻り値として取り得る値、次列はその戻り値について説明しています。

B.2　イベント定義表の見方

以下に、イベント定義表に記載している内容について示します。

Parameters エリアではこのイベントのパラメータについて説明しています。
先頭列はイベントパラメータ構造体のメンバを斜体にて列挙しています。
各パラメータの最終列にてそのパラメータの説明をしています。

パラメータの次列が斜体の場合は、パラメータ(構造体)のメンバであることを示します。

背景色が橙の表は、イベント定義を意味します。このエリアにはイベントタイプを記載しています。			
このエリアでは、イベントにて通知される内容について説明しています。			
Parameters:			
	パラメータ 1	パラメータ 1 の説明です。	
	パラメータ 2	*メンバ 1*	メンバ 1 の説明です。
		メンバ 2	メンバ 2 の説明です。
		メンバ 3	メンバ 3 の説明です。
	パラメータ 3	パラメータ 3 の取り得る値 1	パラメータ 3 の取り得る値 1 の説明です。
		パラメータ 3 の取り得る値 2	パラメータ 3 の取り得る値 2 の説明です。

パラメータ名とパラメータ説明の間に、そのパラメータの取り得る値について説明している場合があります。

付録C　参考文献

1. Bluetooth Core Specification v4.2, Bluetooth SIG
2. Find Me Profile Specification v1.0, Bluetooth SIG
3. Immediate Alert Service Specification v1.0, Bluetooth SIG
4. Proximity Profile Specification v1.0, Bluetooth SIG
5. Link Loss Service Specification v1.0, Bluetooth SIG
6. Tx Power Service Specification v1.0, Bluetooth SIG
7. Health Thermometer Profile Specification v1.0, Bluetooth SIG
8. Health Thermometer Service Specification v1.0, Bluetooth SIG
9. Device Information Service Specification v1.1, Bluetooth SIG
10. Blood Pressure Profile Specification v1.0, Bluetooth SIG
11. Blood Pressure Service Specification v1.0, Bluetooth SIG
12. HID over GATT Profile Specification v1.0, Bluetooth SIG
13. HID Service Specification v1.0, Bluetooth SIG
14. Battery Service Specification v1.0, Bluetooth SIG
15. Scan Parameters Profile Specification v1.0, Bluetooth SIG
16. Scan Parameters Service Specification v1.0, Bluetooth SIG
17. Heart Rate Profile Specification v1.0, Bluetooth SIG
18. Heart Rate Service Specification v1.0, Bluetooth SIG
19. Cycling Speed and Cadence Profile Specification v1.0, Bluetooth SIG
20. Cycling Speed and Cadence Service Specification v1.0, Bluetooth SIG
21. Cycling Power Profile Specification v1.0, Bluetooth SIG
22. Cycling Power Service Specification v1.0, Bluetooth SIG
23. Glucose Profile Specification v1.0, Bluetooth SIG
24. Glucose Service Specification v1.0, Bluetooth SIG
25. Time Profile Specification v1.0, Bluetooth SIG
26. Current Time Service Specification v1.0, Bluetooth SIG
27. Next DST Change Service Specification v1.0, Bluetooth SIG
28. Reference Time Update State Service Specification v1.0, Bluetooth SIG
29. Alert Notification Service Specification v1.0, Bluetooth SIG
30. Alert Notification Profile Specification v1.0, Bluetooth SIG
31. Location and Navigation Service Specification v1.0, Bluetooth SIG
32. Location and Navigation Profile Specification v1.0, Bluetooth SIG
33. Phone Alert Status Service Specification v1.0, Bluetooth SIG
34. Phone Alert Status Profile Specification v1.0, Bluetooth SIG
35. Company ID https://www.bluetooth.com/specifications/assigned-numbers/company-identifiers
36. Services UUID https://www.bluetooth.com/specifications/assigned-numbers/
37. Characteristics UUID https://www.bluetooth.com/specifications/assigned-numbers/
38. Personal Health Devices Transcoding White Paper v1.6, Bluetooth SIG

付録D　用語説明

用語	英語	説明
特性	Characteristic	特性はサービスを識別する値で、各サービスにて公開する特性やそのフォーマットが定義されます。
ロール	Role	役割。それぞれのデバイスが、プロファイルやサービスで規定される役割を果たすことで、ユースケースの実現が可能になります。
コネクションハンドル	Connection Handle	リモートデバイスとの接続を識別するための Controller スタックによって決定されるハンドルです。ハンドルの有効範囲は 0x0000～0x0EFF です。
UUID	Universally Unique Identifier	一意に識別するための識別子です。BLE 規格ではサービスや特性等を識別するために 16bit の UUID が定義されています。
BD アドレス	Bluetooth Device Address	Bluetooth デバイスを識別するための 48bit のアドレスです。BLE 規格ではパブリックアドレスとランダムアドレスが規定されており、少なくともどちらか一方をサポートする必要があります。
パブリックアドレス	Public Address	IEEE に登録し割り当てられた 24bit の OUI(Organizationally Unique Identifier)を含むアドレスです。
ランダムアドレス	Ramdom Address	乱数を含むアドレスで、以下の 3 つに分類されます。 • スタティックアドレス • Non-resolvable private アドレス • Resolvable private アドレス
スタティックアドレス	Static Address	上位 2bit は共に 1 で、残 46bit は全てが 1 または 0 ではない乱数からなるアドレスです。電源断まではそのスタティックアドレスを変更できません。
Non-resolvable private アドレス	Non-resolvable private Address	上位 2bit は共に 0 で、残 46bit は全てが 1 または 0 ではない乱数からなるアドレスです。スタティックおよびパブリックアドレスと等しくてはなりません。 短い期間でアドレスを変更することで攻撃者からの追跡を困難にする目的で使用されます。
Resolvable private アドレス	Resolvable private Address	IRK と 24bit の乱数から生成されるアドレスです。上位 2bit は 0 と 1、上位の残 22bit は全てが 1 または 0 ではない乱数で、下位 24bit は IRK と上位の乱数を元に計算されます。 短い期間でアドレスを変更することで攻撃者からの追跡を困難にする目的で使用されます。 IRK を対向機に配布することで、対向機はその IRK を使用してデバイスを特定することが可能です。
Broadcaster	Broadcaster	GAP のロールのひとつで、Advertising データを送信します。
Observer	Observer	GAP のロールのひとつで、Advertising データを受信します。

Central	Central	GAP のロールのひとつで、物理リンクの確立を行います。Link Layer では Master と呼ばれます。
Peripheral	Peripheral	GAP のロールのひとつで、物理リンクの確立を受け入れます。Link Layer では Slave と呼ばれます。
Advertising	Advertising	接続確立や、データ送信の目的の為に特定チャネル上でデータを送信します。
Scan	Scan	Advertising データを受信します。Scan には、ただ受信するのみの Passive Scan と、SCAN_REQ を送信することで追加情報を要求する Active Scan があります。
White List	White List	接続済みやボンディング済みなどの既知デバイスを White List に登録しておくことで、Advertising データや接続要求を受け取ることを許可するデバイスをフィルタリングすることが可能です。
デバイス名	Device Name	Bluetooth デバイスに任意につけられたデバイスを識別するためのユーザフレンドリーな名前です。BLE 規格では、GAP の特性として GATT サーバによって対向機に公開されます。
Reconection Address	Reconection Address	Non-resolvable private アドレスを使用して、短い期間でアドレスを変更する場合、攻撃者だけでなく対向機もデバイスの特定が困難になります。そのため対向機の公開する Reconection Address 特性に新しい Reconection Address を設定することで再接続時のアドレスを通知します。
スキャンインターバル	Scan Interval	Advertising データの受信を行う間隔です。
スキャンウインドウ	Scan Window	スキャンインターバルごとに Advertising データの受信をおこなう期間です。
コネクションインターバル	Connecton Interval	接続確立後に定期的にデータの送受信を行う間隔です。
コネクションイベント	Connecton Event	コネクションインターバルごとにデータの送受信を行う期間です。
スレーブレイテンシー	Slave Latency	スレーブレイテンシーの回数分のコネクションインターバルにおいて Slave デバイスは受信をする必要がありません。
スーパービジョンタイムアウト	Supervision Timeout	対向機からの応答がなく、リンクが切断されたとみなすタイムアウト時間です。
Passkey Entry	Passkey Entry	ペアリング方式の一つで、互いのデバイスで 6 桁の数値入力または、一方で 6 桁の数値表示、もう一方でその数値入力を行います。
Just Works	Just Works	ペアリング方式の一つで、ユーザアクションを必要としません。
OOB	OOB	ペアリング方式の一つで、Bluetooth 以外の通信方式で取得したデータを使用してペアリングを行います。
IRK	Identity Resolving Key	Resolvable private アドレスの生成や解決に用いる 128bit のキーです。

CSRK	Connection Signature Resolving Key	データ署名の作成および、受信データの署名の確認に使用される 128bit のキーです。
LTK	Long Term Key	暗号化に使用される 128bit のキーです。使用するキーサイズはペアリング時に同意されたサイズになります。
STK	Short Term Key	キー交換時に暗号化するために使用される 128bit のキーです。TK を用いて生成されます。
TK	Temporary Key	STK 生成に必要となる 128bit のキーです。Just Works の場合は 0、Passkey Entry は入力された 6 桁の数値、OOB は OOB データが TK の値となります。

Bluetooth Low Energy プロトコルスタック
API リファレンスマニュアル　基本編

発行年月日 2018 年 3 月 30 日 Rev.1.19

発行　　ルネサス エレクトロニクス株式会社
　　〒135-0061　東京都江東区豊洲 3-2-24（豊洲フォレシア）

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社 〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/

© 2018 Renesas Electronics Corporation. All rights reserved.
Colophon 3.1

Bluetooth Low Energy プロトコルスタック

R01UW0088JJ0119

ルネサス エレクトロニクス株式会社