UG118 : *Bluetooth* ®プロファイル Toolkit 開発者ガイド

Bluetooth GATT のサービスと特性は、Bluetooth データ交換の基盤です。これらを使用して、心拍数モニタなどのデバイスによって表示されるデータの構造、アクセス・タイプ、およびセキュリティ・プロパティを記述します。Bluetooth のサービスと特性は、明確に定義され構造化された形式を採用しており、XML マークアップ言語を使用して簡単に記述できます。

Profile Toolkit は、Bluetooth サービスと特性を記述するための XML ベースのマークアップ言語です。GATT データベースとも呼ばれ、人間とマシンが簡単に判読できる形式のマークアップ言語です。このガイドでは、Profile Toolkit で使用される XML 構文について順を追って説明するとともに、独自の Bluetooth サービスと特性を簡単に記述する方法、アクセスおよびセキュリティのプロパティを構成する方法、およびファームウェアの一部として GATT データベースを含める方法について説明します。

このガイドには実用的な例も記載されており、標準化された Bluetooth およびベンダー固有の独自サービスの使用を紹介しています。これらの例は、独自の開発を開始する際に活用できます。

要点

- Bluetooth GATT のプロファイル、サービス、特性、および属性プロトコルを理解する
- Profile Toolkit を使用して GATT データベースを構築する

第 1 章 プロファイル、サービス、特性、および属性プロトコルを理解する

このセクションでは、Bluetooth のプロファイル、サービス、特性についての基本的な説明と、GATT サーバとクライアント間のデータ交換で属性プロトコルがどのように使用されるのかについても説明します。これらのトピックの詳細については、Bluetooth SIG ウェブサイト（https://www.bluetooth.com/specifications/specs/）をご覧ください。

1.1 GATT ベースの Bluetooth プロファイルおよびサービス

Bluetooth プロファイルは、データを交換する構造を指定します。プロファイルは、プロファイルで使用されるサービスや特性などの要素を定義しますが、セキュリティおよび接続確立パラメータの定義を含む場合もあります。一般的に、プロファイルは、心拍数や鳴動のモニタリングなど、高水準の使用事例を実現するために必要な 1 つまたは複数のサービスで構成されます。標準化されたプロファイルにより、デバイスおよびソフトウェアのベンダーは相互運用可能なデバイスやアプリケーションを構築できます。

1.2 サービス

サービスは、完全な使用事例ではなく、バッテリのモニタリングや温度データなど、デバイスの特定機能を実現するために使用される 1 つまたは複数の特性で構成されたデータの集合体です。

1.3 特性

特性は、データの公開/交換や情報の制御を行うためにサービスで使用される値です。特性には、明確に定義された規定の形式があります。値へのアクセス手段、満たす必要があるセキュリティ要件、オプションで特性の値を表示または解釈する方法についての情報も含まれます。特性には、値の記述、または特性データの表示/通知の許可の構成の記述を行う記述子が含まれている場合もあります。

1.4 属性プロトコル

この属性プロトコルは、GATT サーバと GATT クライアント間のデータ交換を可能にします。また、このプロトコルは、2 つの GATT 当事者間でデータのクエリ、書き込み、表示、通知を行う方法、および情報の制御方法といった一連の操作も提供します。

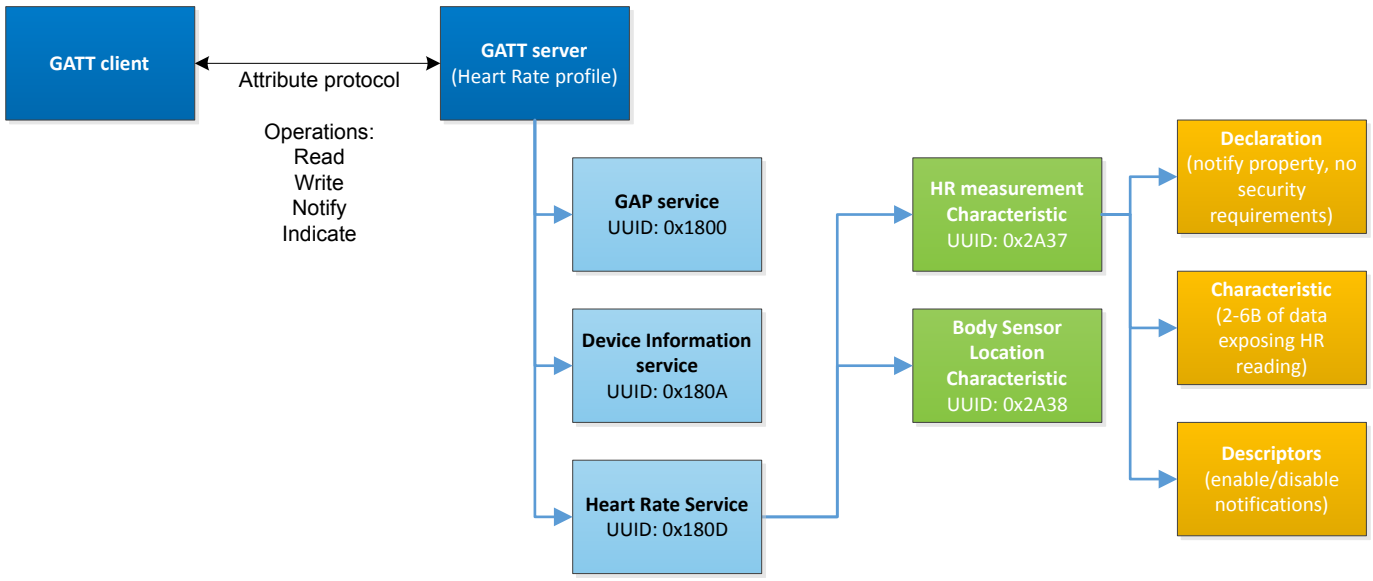


図 1.1. プロファイル、サービス、および特性の関係

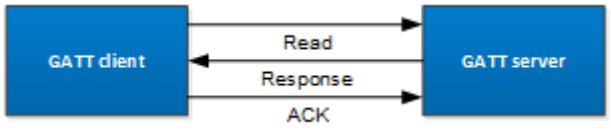


図 1.2. 属性の読み取り操作

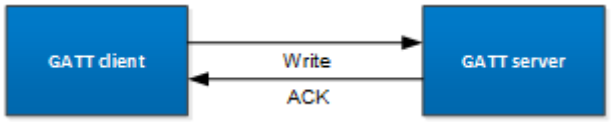


図 1.3. 属性の書き込み操作

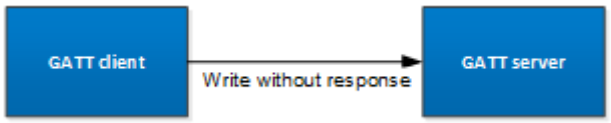


図 1.4. 属性の書き込み操作（応答なし）

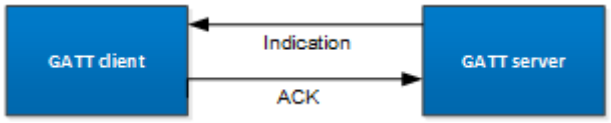


図 1.5. 属性の表示操作

図 1.6. 属性の通知操作

第 2 章 Profile Toolkit を使用して GATT データベースを構築する

本文書のこのセクションでは、Bluetooth プロファイル Toolkit で使用される XML 構文を説明し、Bluetooth サービスと特性を構築する際に使用できるさまざまなオプションについて順を追って説明します。

いくつかの実用的な GATT データベースの例も示します。

2.1 一般的な制限

以下の表に、EFR32BG デバイスでサポートされている GATT データベースの制限を示します。

項目	制限	注記
特性の最大数	無制限ですが、実際にはデータベース内の属性の総数により制限されます。	**const="true"**のプロパティを持たないすべての特性がこのカウントに含まれています。
type="user" 特性の最大長	255 バイト	これらの特性はアプリケーションによって処理されるため、アプリケーションで使用できる RAM の容量によってこの値が制限されます。 注：GATT 手順 **Write Long Characteristic Values**、**Reliable Writes** および **Read Multiple Characteristic Values** は、これらの特性ではサポートされていません。
type="utf-8/hex" 特性の最大長	255 バイト	**const="true"** の 場合**は、デバイスのフラッシュの空き領域によって、この制限が定義されます。const="false"** の場合は、特性には値を保存するために RAM が割り当てられます。使用されるデバイスで使用可能なフラッシュの空き領域によって、これが定義されます。
単一の GATT データベースの属性の最大数	255	通常、1 つの特性は 3 ～ 5 の属性を使用します。
通知可能な特性の最大数	64	
機能の最大数	16	機能のロジック状態によって、各サービスと特性の可視性が決定します。

2.2 <gatt>

GATT データベースは、サービスおよび特性とともに XML 属性 **<gatt>** 内に記述する必要があります。

Parameter	Description
out	Filename for the GATT C source file **Value:** Any UTF-8 string. Must be valid as a filename and end with '.c' **Default:** gatt_db.c
header	Filename for the GATT C header file **Value:** Any UTF-8 string. Must be valid as a filename and end with '.h' **Default:** gatt_db.h
db_name	GATT database structure name and the prefix for data structures in the GATT C source file. **Value:** Any UTF-8 string; must be valid in C. **Default:** bg_gattdb_data
name	Free text, not used by the database compiler **Value:** Any UTF-8 string **Default:** Nothing
prefix	Prefix to add to each 'id' name for defining the handle macro that can be referenced from the C application. **Value:** Any UTF-8 string. Must be valid in C. **Default:** gattdb_ For example: If prefix="gattdb_" and id="temp_measurement" for a particular characteristic, then the following will be generated in the GATT C header file: #define gattdb_temp_measurement X (where X is the handle number for the temp_measurement characteristic)
generic_attribute_service	If it is set to true, Generic Attribute service and its service_changed characteristic will be added in the beginning of the database. The Bluetooth stack takes care of database structure change detection and will send service_changed notifications to clients when a change is detected. In addition, this will enable the GATT-caching feature introduced in Bluetooth 5.1. **Values:** **true:** Generic Attribute service is automatically added to the GATT database and GATT caching is enabled. **false:** Generic Attribute service is not automatically added to the GATT database and GATT caching is disabled. **Default:** false
gatt_caching	The GATT caching feature is enabled by default if generic_attribute_service is set to true. However, it can be disabled by setting this attribute to false.

例：GATT データベースの定義。

```
<?xml version="1.0" encoding="UTF-8" ?> <gatt out="my_gatt_db.c" header="my_gatt_db.h" db_name="my_gatt_db_" prefix="my_gatt_"
generic_attribute_service="true" name="My GATT database"> … </gatt>
```

2.3 <capabilities_declare>

GATT データベースのサービスと特性は、**機能**を使用して表示または非表示にできます。機能は、<capability> 要素で宣言する必要があり、GATT データベースのすべての機能は、<capability> 要素のシーケンスで構成される <capabilities_declare> 要素で最初に宣言される必要があります。データベース内の機能の最大数は 16 です。

この新しい機能は、レガシ GATT XML データベース（Silicon Labs Bluetooth スタック・バージョン 2.4.x 以前）には影響しません。これらのデータベースでは機能が明示的に宣言されないため、すべてのサービスと特性がリモート GATT クライアントに表示されたままになります。

例：機能の宣言

```
<capabilities_declare> <capability enable="false">機 能 _1</capability> <capability enable="false">機 能 _2</capability> </capabilities_declare>
```

2.3.1 <capability>

各機能は、<capability> 要素を使用して <capabilities_declare> 要素内で個別に宣言される必要があります。<capability> 要素には "enable" という名前の 1 つの属性があり、データベースの初期化時に機能のデフォルトの状態を示します。

<capability> 要素のテキスト値は、生成されたデータベースの C ヘッダの対象機能に対する識別子名になります。したがって、この値は C で有効である必要があります。

機能の継承

サービスと特性によって、使用する機能を宣言できます。機能が宣言されない場合は、次の継承規則が適用されます。

1. 機能を宣言しないサービスには、<capabilities_declare> 要素のすべての機能が付与される。
2. 機能を宣言しない特性には、その特性が属するサービスのすべての機能が付与される。サービスによって <capabilities_declare> の機能のサブセットが宣言された場合は、そのサブセットのみが特性によって継承されます。
3. 特性のすべての属性が特性の機能を継承します。

可視性

機能を有効化または無効化して、次のロジックに従って GATT クライアントに対してサービスと特性を表示または非表示にできます。

1. 機能の**少なくとも 1 つ**が**有効**になっている場合、サービスとそのすべての特性は**表示**に設定されます。
2. 機能の**すべて**が**無効**になっている場合、サービスとそのすべての特性は**非表示**に設定されます。
3. 機能の**少なくとも 1 つ**が**有効**になっている場合、特性とそのすべての属性は**表示**に設定されます。
4. 機能の**すべて**が**無効になっている場合、特性とそのすべての属性は非表示に設定されます。**

Parameter	Description
enable	Sets the default state of a capability at database initialization. **Values:** **true:** Capability is enabled. **false:** Capability is disabled. **Default:** true

例：機能の宣言

```
<capabilities_declare>
<!-- This capability is enabled by default and the identifier is cap_light -->
<capability enable="true">cap_light</capability>
<!-- This capability is disabled by default and the identifier is cap_color -->
<capability enable="false">cap_color</capability>
</capabilities_declare>
```

2.4 <service>

GATT サービスの定義は、XML 属性 **<service>** およびそのパラメータによって行われます。

以下の表に、関連する値を定義するために使用できるパラメータを示します。

Parameter	Description
uuid	Universally Unique Identifier. The UUID uniquely identifies a service. 16-bit values are used for the services defined by the Bluetooth SIG and 128-bit UUIDs can be used for vendor specific implementations. **Range:** **0x0000 – 0xFFFF**: Reserved for Bluetooth SIG standardized services **0x00000000-0000-0000-0000-000000000000 - 0xFFFFFFFF-FFFF-FFFF-FFFF-FFFFFFFFFFFF**: Reserved for vendor specific services.
id	The ID is used to identify a service within the service database and can be used as a reference from other services (include statement). Typically, this does not need to be used. **Value:** Any UTF-8 string
type	The type field defines whether the service is a primary or a secondary service. Typically this does not need to be used. **Values:** **primary**: a primary service **secondary**: a secondary service **Default**: primary
advertise	This field defines if the service UUID is included in the advertisement data. The advertisement data can contain up to 13 16-bit UUIDs or one (1) 128-bit UUID. **Values:** **true**: UUID included in advertisement data **false**: UUID not included in advertisement data **Default**: false **Note:** You can override the advertisement data with the GAP API, in which case this is not valid.

例：汎用アクセス・プロファイル (GAP) サービスの定義。

```
<!-- Generic Access Service -->
<service uuid="1800">
      ...
</service>
```

例：ベンダー固有サービスの定義。

```
<!-- A vendor specific service -->
<service uuid="25be6a60-2040-11e5-bd86-0002a5d5c51b">
      ...
</service>
```

例：UUID を持つ Heart Rate サービスはアドバタイズ・データおよび ID “hrs” に含まれます。

```
<!-- Heart Rate Service -->
<service uuid="180D" id="hrs" advertise="true">
      ...
</service>
```

Note: http://www.itu.int/en/ITU-T/asn1/Pages/UUID/uuids.aspx で独自の 128 ビット UUID を生成できます。

2.4.1 <capabilities>

サービスは、<capabilities> 要素を使用して機能を宣言できます。要素は、<capability> 要素のシーケンスで構成されます。この要素の識別子は <capabilities_declare> 要素**にも含める必要があります**。属性 "enable" はこのコンテキストで宣言された機能に影響を与えないため、除外できます。

サービスが機能を宣言しない場合は、**継承規則**に従って **<capabilities_declare>** の **すべての機能**が付与されます。

サービスとそのすべての特性は、機能の**少なくとも 1 つ**が**有効**になっている場合は**表示**に設定され、**機能のすべて**が**無効**になっている場合は**非表示**に設定されます。

例：機能の宣言

```
<capabilities> <capability>cap_light</capability> <capability>cap_color</capability> </capabilities>
```

2.4.2 <informativeText>

XML 要素 <informativeText> は情報提供目的（コメント）で使用でき、実際の GATT データベースには使用できません。

2.4.3 <include>

サービスは、XML 属性 **<include>** を使用して、別のサービスに含めることができます。

Parameter	Description
id	ID of the included service **Value:** ID of another service

例：GAP サービス内に Heart Rate サービスを含める。

```
<!-- Generic Access Service -->
<service uuid="1800">

      <!-- Include HR Service -->
      <include id="hrs" />
      ...
</service>
```

2.5 <characteristic>

サービスによって公開されるすべての特性は、XML 属性 **<characteristic>** およびそのパラメータで定義されます。これらは、**<service>** XML 属性タグ内で使用する必要があります。

以下の表に、関連する値を定義するために使用できるパラメータを示します。

Parameter	Description
uuid	Universally Unique Identifier. The UUID uniquely identifies a characteristic. 16-bit values are used for the services defined by the Bluetooth SIG and 128-bit UUIDs can be used for vendor specific implementations. **Range:** **0x0000 - 0xFFFF**: Reserved for Bluetooth SIG standardized characteristics. **0x00000000-0000-0000-0000-000000000000 to 0xFFFFFFFF-FFFF-FFFF-FFFF-FFFFFFFFFFFF** : Reserved for vendor specific characteristics.
id	The ID is used to identify a characteristic. The ID is used within a C application to read and write characteristic values or to detect if notifications or indications are enabled or disabled for a specific characteristic. When the project is built, the generated GATT C header file contains a macro with the characteristic 'id' and corresponding handle value. **Value:**Any UTF-8 string
const	Defines if the value stored in the characteristic is a constant. **Default:** false
name	Free text, not used by the database compiler. **Value:**Any UTF-8 string **Default:** Nothing

例：GAP サービスにデバイス名特性を追加する。

```
<!-- Generic Access Service -->
<service uuid="1800">

      <!-- Device name -->
      <characteristic uuid="2a00">
            …
      </characteristic>
      …
</service>
```

例：ID を持つベンダー固有のサービスにベンダー固有の特性を追加する。

```
<!-- A vendor specific service -->
<service uuid="25be6a60-2040-11e5-bd86-0002a5d5c51b">

      <!-- My proprietary data -->
      <characteristic uuid="59cd69c0-2043-11e5-a717-0002a5d5c51b" id="mydata" >
            …
      </characteristic>
      …
</service>
```

2.5.1 <capabilities>

特性は、<capabilities> 要素を使用して機能を宣言できます。要素は、<capability> 要素のシーケンスで構成されます。この要素の識別子は、親サービスでも**宣言される（または完全に継承される）必要があります**。属性 "enable" はこのコンテキストで宣言された機能に影響を与えないため、除外できます。

特性が機能を宣言しない場合は、**継承規則**に従って、その特性が属する**サービス**の**すべての機能**が付与されます。特性のすべての属性が特性の機能を継承します。

特性とそのすべての属性は、機能の**少なくとも 1 つ**が**有効**になっている場合は**表示**に設定され、**機能のすべて**が**無効**になっている場合は**非表示**に設定されます。

例：機能の宣言

```
<capabilities>
<capability>cap_light</capability>
<capability>cap_color</capability>
</capabilities>
```

2.5.2 <properties>

特性のアクセス・プロパティとそのアクセス許可レベルは、XML 属性**<properties>**の子属性によって定義され、**<characteristic>**XML 属性タグ内で使用する必要があります。特性には、同時に複数のアクセス・プロパティを設定できます。たとえば、読み取り、書き込み、または両方が可能です。各アクセス・プロパティには、異なる許可レベル（たとえば、暗号化や認証済み）を割り当てることができます。

次の表に、可能なアクセス・プロパティを示します。表の各行は、**<properties>** 属性の下の新しい属性です。

Attribute	Description
read	Characteristic can be read by a remote device. **Values:** **true**: Characteristic can be read **false**: Characteristic cannot be read **Default**: false
write	Characteristic can be written by a remote device **Values:** **true**: Characteristic can be written **false**: Characteristic cannot be written **Default**: false
write_no_response	Characteristic can be written by a remote device. Write without response is not acknowledged over the Attribute Protocol. **Values:** **true**: Characteristic can be written **false**: Characteristic cannot be written **Default**: false
notify	Characteristic has the notify property and characteristic value changes are notified over the Attribute Protocol. Notifications are not acknowledged over the Attribute Protocol. **Values:** **true**: Characteristic has notify property. **false**: Characteristic does not have notify property. **Default**: false **Note**: The *notify* attribute is stored in the SIG defined Client Characteristic Configuration Descriptor (a descriptor with the UUID 0x2902, which will be autogenerated when notifications are enabled). If you manually add a CCCD to the characteristic, the descriptor’ s value will overwrite this setting.
indicate	Characteristic has the indicate property and characteristic value changes are indicated over the Attribute Protocol. Indications are acknowledged over the Attribute Protocol. **Values:** **true**: Characteristic has indicate property. **false**: Characteristic does not have indicate property. **Default**: false **Note**: The *indicate* attribute is stored in the SIG defined Client Characteristic Configuration Descriptor (a descriptor with the UUID 0x2902, which will be autogenerated when indications are enabled). If you manually add a CCCD to the characteristic, the descriptor’ s value will overwrite this setting.

Attribute	Description
reliable_write	Allows using a reliable write procedure to modify an attribute; this is just a hint to a GATT client. The Bluetooth stack always allows the use of reliable writes to modify attributes. **Values:** **true**: Reliable write enabled. **false**: Reliable write disenabled. **Default**: false

次の表は、権限レベルまたはセキュリティ要件を示しています。セキュリティ要件は、どのアクセスプロパティにも割り当てることができます。

Attribute	Description
authenticated	Accessing the characteristic value requires an authentication. To access the characteristic with this property the remote device has to be bonded using MITM protection and the connection must be also encrypted. **Values:** **true**: Authentication is required **false**: Authentication is not required **Default**: false
encrypted	Accessing the characteristic value requires an encrypted link. Devices with iOS 9.1 or higher must also be bonded at least with Just Works pairing. **Values:** **true**: Encryption is required **false**: Encryption is not required **Default**: false
bonded	Accessing the characteristic value requires an encrypted link. Devices must also be bonded at least with Just Works pairing. **Values:** **true**: Bonding and encryption are required **false**: Bonding is not required **Default**: false

例：*const* および *read* プロパティを持つデバイス名の特性。

```
<!-Device Name-->
<characteristic const = true uuid="2a00">
<properties>
    <read authenticated="false" bonded="false" encrypted="false"/>
</properties>
</characteristic>
```

例：リモート・デバイスによって値の変更を可能にする *read* および *write* プロパティを持つデバイス名特性。

```
<!-Device Name-->
<characteristic uuid="2a00">
<properties>
    <read authenticated="false" bonded="false" encrypted="false"/>
        <write authenticated="false" bonded="false" encrypted="false"/>
</properties>
</characteristic>
```

例：*notify* プロパティを持つ心拍数測定特性。

```
<!-Heart Rate Measurement -->
<characteristic uuid="180D">
<properties>
    <notify authenticated="false" bonded="false" encrypted="false"/>
</properties>
</characteristic>
```

例：*encrypted read* プロパティを持つ特性。

```
<!-Device Name-->
<characteristic uuid="1234">
<properties>
        <read authenticated="false" bonded="false" encrypted="true"/>
</properties>
</characteristic>
```

例：*authenticated write* プロパティを持つ特性。

```
<!-Device Name-->
<characteristic uuid="1234">
<properties>
    <write authenticated="true" bonded="false" encrypted="false"/>
</properties>
</characteristic>
```

例：*authenticated indicate* プロパティを持つ特性。

```
<!-Descriptor value changed -->
<characteristic uuid="2A7D">
<properties>
    <indicate authenticated="true" bonded="false" encrypted="false"/>
</properties>
</characteristic>
```

2.5.3 <value>

特性のデータ・タイプと長さは、XML 属性 **<value>** およびそのパラメータで定義されます。これらは、**<characteristic>** XML 属性タグ内で使用する必要があります。

以下の表に、関連する値を定義するために使用できるパラメータを示します。

Parameter	Description
length	Defines a fixed length for the characteristic or the maximum length if ***variable_length*** is true. If length is not defined and there is a value (e.g. data exists inside <value></value>), then the value length is used to define the length. If both ***length*** and value are defined, then the following rules apply: 1. If ***variable_length*** is false and ***length*** is bigger than the value's length, then the value will be padded with 0's at the end to match the attribute's ***length***. 2. If ***length*** is smaller than the value's length, then the value will be clipped to match ***length***, regardless of whether ***variable_length*** is true or false. **Range:** **0 - 255**: Length in bytes if ***type*** is 'hex', 'utf-8', or 'user' **0 - 512:** Length in bytes if **type** is 'user' **Default**: 0
variable_length	Defines that the value is of variable length. The maximum length must also be defined with the ***length*** attribute or by defining a value. If both ***length*** and value are defined, then the rules described in ***length*** apply. **Values:** **true**: Value is of variable length **false**: Value has a fixed length **Default**: false
type	Defines the data type. **Values:** **hex**: Value type is hex **utf-8**: Value is a string **user:** When the characteristic type is marked as type="user", the application is responsible for initializing the characteristic value and also providing it, for example, when read operation occurs. The Bluetooth stack does not initialize the value or automatically provide the value when it is being read. When this is set, the Bluetooth stack generates ***gatt_server_user_read_request*** or ***gatt_server_user_write_request***, which must be handled by the application. **Default**: utf-8

例：*notify* プロパティと 2 バイトの固定長を持つ心拍数測定特性。

```
<!-Heart Rate Measurement -->
<characteristic uuid="180D">
<value length="2" type="hex" variable_length = "false"/>
<properties>
<notify authenticated="false" bonded="false" encrypted="false"/>
</properties>
</characteristic>
```

例：最大長が 20 バイトの可変長のベンダー固有特性。

```
<!-My proprietary data -->
<characteristic uuid="59cd69c0-2043-11e5-a717-0002a5d5c51b" id="mydata">
<value variable_length="true" length="20" type="hex" />
```

```
<properties>
    <notify authenticated="false" bonded="false" encrypted="false"/>
</properties>
</characteristic>
```

例：特性の値と長さは、<value> タグ内に実際の値を入力して定義することもできます。

```
<characteristic const="true" id="device_name" name="Device Name" sourceId="org.bluetooth.characteristic.gap.device_name" uuid="2A00">
<value length="17" type="utf-8" variable_length="false">EFR32 BGM111</value>
<properties>
        <read authenticated="false" bonded="false" encrypted="false"/>
</properties>
</characteristic>
```

上記の例の場合、値は “**EFR32 BGM111**” で、長さは 17 バイトです。

例：値の長さより**大きい**長さで length と value の両方を定義する場合。

```
<!-- Device name -->
<characteristic uuid="2a00">
     <properties read="true" />
     <value type="hex" length="4" variable_length="false">0102</value>
</characteristic>
```

上記の例では、***length*** が値の長さより大きく、value に 0 が埋め込まれた状態になるため、値は “**01020000**” になります。

例：値の長さより**小さい**長さで length と value の両方を定義する場合。

```
<!-- Device name -->
<characteristic uuid="2a00">
     <properties read="true" />
     <value type="hex" length="2" variable_length="false">01020304</value>
</characteristic>
```

上記の例では、***length*** が値の長さより小さく、その結果、値が ***length*** と一致するようにクリップされるため、値は “**0102**” になります。

2.5.4 <descriptor>

XML 要素 <descriptor> は、汎用特性記述子を定義するために使用できます。

記述子のプロパティは <properties> 要素によって定義され、読み取りアクセスおよび/または書き込みアクセスのみが許可されます。値は特性値の場合と同じように、<value> 要素によって定義されます。

注：Client Characteristic Configuration Descriptor (UUID: 0x2902)を手動で追加した場合、値はキャラクタリスティックの*通知/表示*のプロパティを 上書きします。手動で追加しない場合は、 キャラクタリスティックが通知や表示を有効にしたときに CCCD が自動的に生成されます。

例：タイプ UUID 2908 の特性記述子の追加。

```
<characteristic uuid="2a4d" id="hid_input">
<properties notify="true" read="true" />
<value length="3" />
      <descriptor const="false" discoverable="true" id="" name="Custom Descriptor" sourceId="" uuid="2908">
        <properties>
          <read authenticated="false" bonded="false" encrypted="false"/>
        </properties>
        <value length="0" type="hex" variable_length="false">00</value>
      </descriptor>
</characteristic>
```

2.5.5 <description>

特性ユーザ説明値は、XML 属性 <description> で定義されます。これらは、<characteristic> XML 属性タグ内で使用する必要があります。

特性のユーザ説明はオプション値です。これはリモート・デバイスによって使用され、これを使用して、ユーザに分かりやすい特性の説明をアプリケーションのユーザ・インターフェイスに表示したりすることができます。

例：定数文字列「心拍数測定」

```
<characteristic uuid="2a37">
<properties>
    <notify authenticated="false" bonded="false" encrypted="false"/>
</properties>
<description> Heart Rate Measurement </description>
</characteristic>
```

プロパティ要素を使用して、属性のリモート変更を許可できます。

例：リモート読み取りを許可するが、書き込みには結合が必要

```
<characteristic uuid="2a37">
    <properties>
        <read authenticated="false" bonded="false" encrypted="true"/>
        <write authenticated="false" bonded="true" encrypted="false"/>
    </properties>
</characteristic>
```

Note: 説明が書き込み可能な場合、GATT パーサは、Bluetooth 準拠に設定された *writable_auxiliaries* ビットを使用して拡張プロパティ属性を自動的に追加します。

2.5.6 <aggregate>

XML 要素 <aggregate> を使用すると、ID を属性ハンドルに自動的に変換することにより、集計された特性形式記述子を作成できます。

属性 ID は、特性表示形式記述子を参照する必要があります。

例：特性集計を追加

```
<characteristic uuid="da8a80c0-829d-498f-b70b-e85c95e0f839"> <properties notify="true" read="true"/> <value length="10" />
<aggregate> <attribute id="format1" /> <attribute id="format2" /> </aggregate> </characteristic>
```

2.6 GATT の例

例：定数としてデバイス名特性および外観特性、*read* プロパティを持つ完全な GAP サービス。

```
<?xml version="1.0" encoding="UTF-8" ?>
<gatt>

<!--Generic Access-->
<service advertise="false" name="Generic Access" requirement="mandatory" sourceId="org.bluetooth.service.generic_access"
type="primary" uuid="1800">
    <informativeText>Abstract: The generic_access service contains generic information about the device. All available
       Characteristics are readonly. </informativeText>
    <!--Device Name-->
   <characteristic const="true" id="device_name" name="Device Name" sourceId="org.bluetooth.characteristic.gap.device_name"
     uuid="2A00">
      <informativeText/>
      <value length="17" type="utf-8" variable_length="false">EFR32 BGM111</value>
      <properties>
        <read authenticated="false" bonded="false" encrypted="false"/>
      </properties>
    </characteristic>

    <!--Appearance-->
    <characteristic const="true" name="Appearance" sourceId="org.bluetooth.characteristic.gap.appearance" uuid="2A01">
       <informativeText>Abstract: The external appearance of this device. The values are composed of a category (10-bits) and sub-
categories (6-bits). </informativeText>
      <value length="2" type="hex" variable_length="false">0000</value>
      <properties>
        <read authenticated="false" bonded="false" encrypted="false"/>
      </properties>
    </characteristic>
  </service>
</gatt>
```

例：リンク損失および即時アラートサービス。

```
<?xml version="1.0" encoding="UTF-8" ?>

<gatt>
<!--Link Loss-->
   <service advertise="false" id="link_loss" name="Link Loss" requirement="mandatory" sourceId="org.bluetooth.service.link_loss"
type="primary" uuid="1803">
    <!--Alert Level-->
        <characteristic const="false" id="alert_level" name="Alert Level" sourceId="org.bluetooth.characteristic.alert_level"
uuid="2A06">
      <value length="1" type="hex" variable_length="false"/>
      <properties>
        <read authenticated="false" bonded="false" encrypted="false"/>
        <write authenticated="false" bonded="false" encrypted="false"/>
      </properties>
    </characteristic>
  </service>

  <!--Immediate Alert-->
          <service advertise="false" id="immediate_alert" name="Immediate Alert" requirement="mandatory"
sourceId="org.bluetooth.service.immediate_alert" type="primary" uuid="1802">
    <!--Alert Level-->
        <characteristic const="false" id="alert_level" name="Alert Level" sourceId="org.bluetooth.characteristic.alert_level"
uuid="2A06">
      <value length="1" type="hex" variable_length="false"/>
      <properties>
        <write_no_response authenticated="false" bonded="false" encrypted="false"/>
      </properties>
    </characteristic>
  </service>

</gatt>
```

例：機能を持つ GATT データベース

```
<gatt db_name="light_gattdb" out="gatt_db.c" header="gatt_db.h" generic_attribute_service="true">
  <capabilities_declare>
    <capability enable="true">cap_light</capability>
    <capability enable="false">cap_color</capability>
  </capabilities_declare>

  <!--Light Service-->
    <service advertise="false" id="light_service" name="Light Service" requirement="mandatory" sourceId="" type="primary"
uuid="257f993d-756e-baa6-e69c-8b101e4e6b3f">
    <informativeText>Info about custom service</informativeText>
    <capabilities>
      <capability>cap_light</capability>
      <capability>cap_color</capability>
    </capabilities>

    <!--Ligth Control-->
    <characteristic const="false" id="light_control" name="Ligth Control" sourceId="" uuid="85e82a1c-8423-610b-9fea-5ad999445231">
      <description>User description</description>
      <informativeText/>
      <capabilities>
        <capability>cap_light</capability>
      </capabilities>
      <value length="0" type="user" variable_length="false"/>
      <properties>
        <read authenticated="false" bonded="false" encrypted="false"/>
        <write authenticated="false" bonded="false" encrypted="false"/>
        <write_no_response authenticated="false" bonded="false" encrypted="false"/>
        <reliable_write authenticated="false" bonded="false" encrypted="false"/>
        <indicate authenticated="false" bonded="false" encrypted="false"/>
        <notify authenticated="false" bonded="false" encrypted="false"/>
      </properties>
    </characteristic>

    <!--Color control-->
    <characteristic const="false" id="color_control" name="Color control" sourceId="" uuid="16b90591-c54a-e7c9-413e-a82748a1e783">
      <informativeText/>
      <capabilities>
        <capability>cap_color</capability>
      </capabilities>
      <value length="0" type="user" variable_length="false"/>
      <properties>
        <read authenticated="false" bonded="false" encrypted="false"/>
        <write authenticated="false" bonded="false" encrypted="false"/>
      </properties>
    </characteristic>
  </service>
</gatt>
```

- 機能 cap_light および cap_color が有効になっている場合は、light_service 全体が表示に設定されます。
- 機能 cap_light が無効になっている場合は、特性 light_control が非表示に設定されます。
- 機能 cap_color が無効になっている場合は、特性 color_control が非表示に設定されます。

第 3 章 コードファイルの生成

本文書のこのセクションでは、Simplicity Studio または bgbuild Python スクリプトとお使いの IDE を使用して、gatt_configuration.btconf XML ファイルから gatt_db.c/h コードファイルを生成する方法について説明します。

3.1 Simplicity Studio を使う

Simplicity Studio で GATT コンフィギュレーターを使用する場合、コードファイル (gatt_db.c/gatt_db.h) は、設定が保存されるたびに自動的に生成されます。

3.2 bgbuild を使う

コードファイルは、SDK に含まれる bgbuild Python スクリプトを使用して、IDE から独立して生成できます。

```
$GSDK_PATH¥protocol¥bluetooth¥bin¥gatt¥bgbuild.py
```

$GSDK_PATH は、~/SimplicityStudio/SDKs/gecko_sdk/ など、選択した GSDK のインストール・ディレクトリです。

このスクリプトでは、pip install Jinja2 を呼び出して、Python 3 と Jinja2 パッケージをインストールする必要があります。このスクリプトは、Simplicity Studio 5 / SDK v4.x で作成された GATT 構成、またはこのユーザーガイドに従って手動で書き込まれた GATT 構成のみを解析します。GATT コンフィギュレーターは、古い gatt.xml 形式をインポートできます。

必須引数:

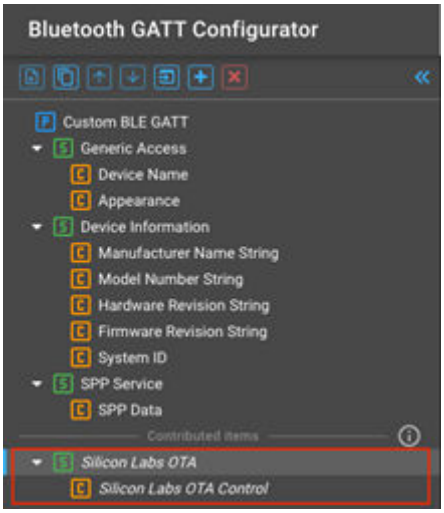
- GATT XML ファイルへのパスか、XML ファイルを探すためのディレクトリ。「;」で個別に入力

オプションの引数：

- -h, --help: ヘルプ・メッセージを表示
- -o OUTDIR, --outdir OUTDIR: ファイルを生成する出力ディレクトリ（Simplicity Studio 5 は自動生成フォルダにソースを生成します。プロジェクト構造の別の場所にソースを生成すると、競合が発生する可能性があります。）

```
python $GSDK_PATH/protocol/bluetooth/bin/gatt/bgbuild.py ~/SimplicityStudio/v5_workspace/btmesh_soc_empty/config/btconf/gatt_configuration.btconf -o ~/SimplicityStudio/v5_workspace/btmesh_soc_empty/autogen/
```

Bluetooth GATT コンフィギュレーターでは、GATT に **Contributed アイテム** セクションを含めることができます。これらは、ユーザーインターフェイスから変更できない追加の XML ファイルですが、Simplicity Studio のソース生成にはこれらのファイルが含まれます。



bgbuild で同じ結果を生成するには、config/btconf フォルダ全体を含める必要があります。

```
python $GSDK_PATH/protocol/bluetooth/bin/gatt/bgbuild.py ~/SimplicityStudio/v5_workspace/btmesh_soc_empty/config/btconf/ -o ~/SimplicityStudio/v5_workspace/btmesh_soc_empty/autogen/
```

第 4 章 修正履歴

改訂 2.8

2023 年 10 月

- 日本語訳のみのセクション 3.2 bgbuild を使う で、“pip install jinja22” を “pip install jinja2” に修正。
- セクション 3.2 bgbuild を使う で、コードファイルを手動で生成するプロセスを更新。

Smart. Connected. Energy-Friendly.

IoT Portfolio
www.silabs.com/products

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com