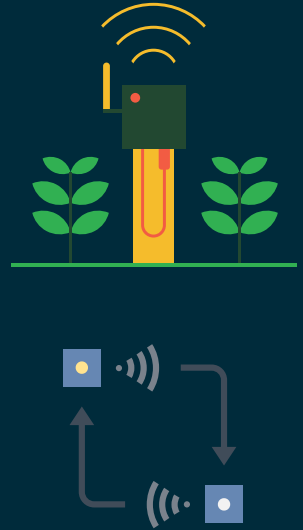
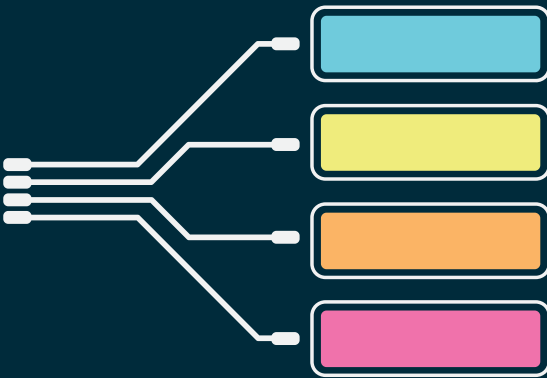


# GATT DESIGN GUIDELINES



---

*Learn how to design the GATT structure for your Bluetooth LE device, including Profiles, Services, and Characteristics*

---

# GATT Design Guidelines

---

While GATT is a pretty flexible framework, there are a few general guidelines to follow when defining the services and characteristics within it. Following are some recommendations:

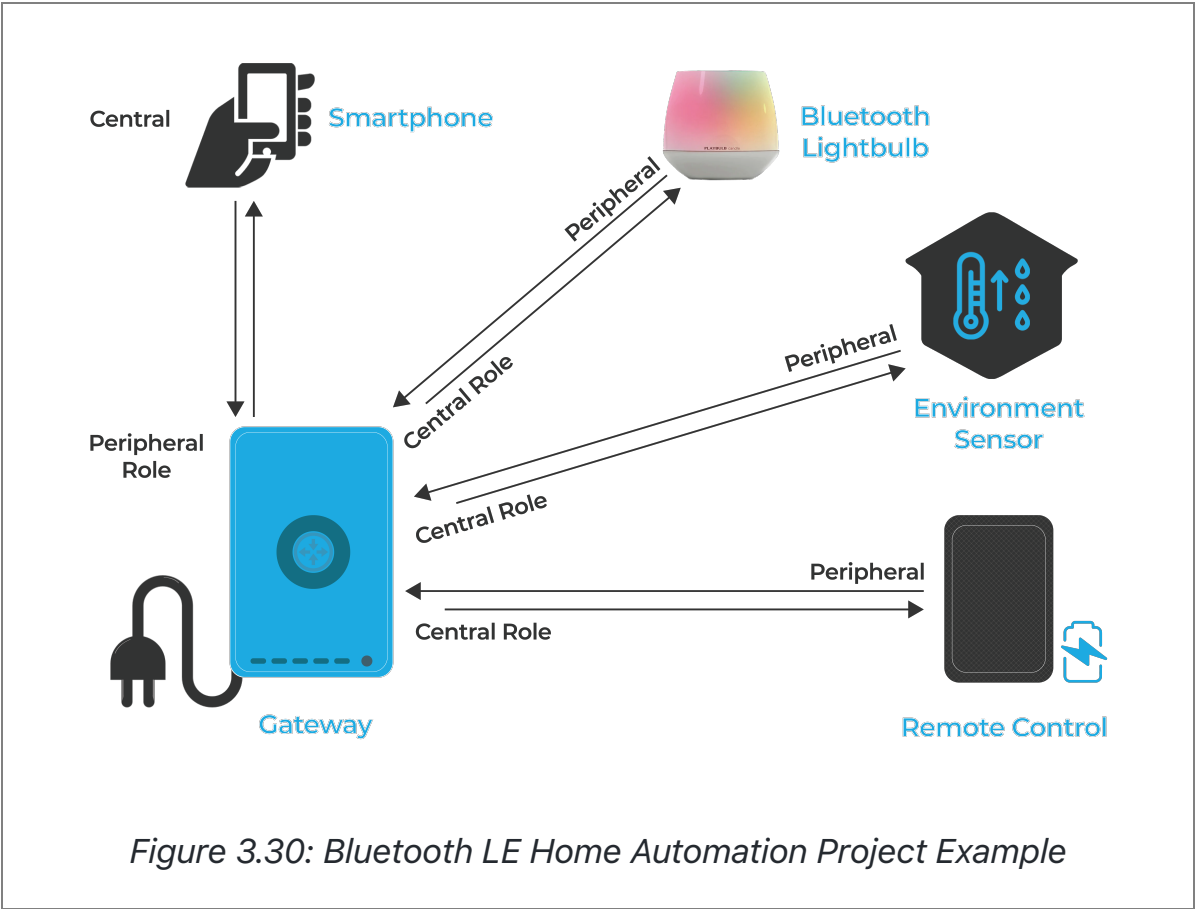
- Implement the following mandatory service and its characteristics:
  - **Generic Access Profile (GAP)** service.
  - **Name** and **Appearance** characteristics within the GAP service.
- One thing to keep in mind is that vendor SDKs usually do not require you to implement this service explicitly, but rather they provide APIs for setting the **name** and **appearance**. The SDK then handles creating the GAP service and setting the characteristics according to the user-provided values.
- Utilize the Bluetooth SIG-adopted profiles, services, and characteristics in your design whenever possible. This has the following benefits:
  - You get the benefit of reducing the size of data packets involving UUIDs for services and characteristics (including advertisement packets, discovery procedures, and others) — since 16-bit UUID values are used instead of 128-bit values.
  - Bluetooth chipset and module vendors usually provide implementations of these profiles, services, and characteristics in their SDKs — reducing development time considerably.
  - Interoperability with other third-party devices and applications, allowing more devices to interface with your device and provide a richer user experience.
- Group characteristics that serve related functionality within a single service.
- Avoid having services with too many characteristics. A good separation of services makes it faster to discover certain characteristics and leads to a better GATT design that's modular and user-friendly.

Let's go over a practical example showing how to design the GATT for a simple Bluetooth LE home automation system.

## GATT Design Exercise

Designing the GATT for your Bluetooth LE device can be a challenge. To make this task easier, let's go through the exercise of designing the GATT for a simple Bluetooth LE smart home system.

The system is a hypothetical one, but one that will help you better understand the steps taken in designing the GATT of a real-life application rather than some other generic abstract system. Here's a diagram showing the different elements of the home automation system and how they interact with each other.



The system consists of multiple elements (devices). Some of these are off-the-shelf components that we don't have control over, while others are devices whose firmware we do have control over and to which we will design their GATT structure.

## General System Description

Let's go ahead and describe the main user scenarios of the system:

1. The homeowner can use the **remote control** to turn on/off the **Bluetooth lightbulb**.
2. The homeowner can monitor changes in the temperature and humidity of the **environment sensor**.
3. The homeowner is notified of the battery levels of the **remote control**, **Bluetooth lightbulb**, and **environment sensor**.

## System Elements

Now, let's go over the different elements within the system.

### 1. Gateway

The **gateway** will act as a Bluetooth LE central when communicating with all the other devices except the smartphone, where it will act as a peripheral. We have control over this device and we will be designing its GATT.

The commands for controlling the Bluetooth lightbulb will be routed from the remote control through the gateway and to the Bluetooth lightbulb.

### 2. Remote Control

The **remote control** is a device that will act as a peripheral only, and one that we will be designing the GATT for.

### 3. Environment Sensor

This is an off-the-shelf device over whose GATT design we have no control, so we will simply be interested in reading the data from it (temperature and humidity readings).

#### 4. Bluetooth Lightbulb

This is another off-the-shelf device over whose GATT we have no control.

#### 5. Smartphone

This is also another existing device over whose GATT we have no control.

## GATT Design Step-by-Step

Let's go through the GATT design process, step-by-step:

### Step 1: Documenting the Different User Scenarios and Data Points

Even though the GATT is usually more focused on the peripheral role (since a peripheral is usually the server exposing the data), the central can still act as the server in some cases for specific data points it needs to expose. Also, since we are designing both sides of the system (peripheral and central on the gateway), it helps to think in terms of what needs to happen from each side since this could affect some aspects of the system and GATT design.

The gateway device acts in both the central and peripheral roles. Each of these roles is used to enable communication with different devices within the system. The main purpose of the gateway is to act as a central device to read data from multiple peripherals. It then exposes this information via the peripheral role to another central device (the smartphone) that is able to relay this data to a **cloud server**.

Let's go through the user scenarios from the gateway's perspective for each of the central and peripheral roles.

#### *Gateway Peripheral Role*

- The remote control notifies the gateway when specific buttons are pressed to turn on/off the Bluetooth lightbulb.
- The following data points within the system need to be reported to a cloud

server via the gateway. These data points get exposed as a GATT Server in the peripheral role to the central device (the smartphone) that has an Internet connection (which allows it to relay this data up to the cloud server).

- Environment sensor temperature reading
- Environment sensor humidity reading
- Bluetooth lightbulb status (on/off)
- Individual battery levels for the environment sensor, Bluetooth lightbulb, and remote control.

### *Gateway Central Role*

The gateway device needs to read some of the data exposed by devices within the system and get notified of other data points exposed by these devices.

The remote control provides one main function: turning the Bluetooth lightbulb on or off. It acts strictly in the peripheral role and needs to expose the following data points:

- **On button press:** the gateway needs to be notified of this event when it occurs.
- **Off button press:** the gateway needs to be notified of this event when it occurs.
- **Battery level:** the gateway needs to be able to read this data and be notified when it changes.

### **Step 2: Define the Services, Characteristics, and Access Permissions**

The next step is to group the **characteristics** into meaningful groups (**services**) based on their functionalities and define the access permissions for each of these characteristics.

### *Gateway*

We have one GATT Server for the gateway, and it exists for the peripheral role. By looking at the data points (characteristics) we listed previously, we can group them into the following services:

- **Environment Sensor Service:**
  - Environment sensor temperature reading characteristic: "Temperature."  
**Access permissions:** Read, notify.
  - Environment sensor humidity reading characteristic: "Humidity."  
**Access permissions:** Read, notify.
  - Battery level characteristic: "Battery Level."  
**Access permissions:** Read, notify.
- **Playbulb Service:**
  - Light status characteristic: "Light Status."  
**Access permissions:** Read, notify.
  - Battery level characteristic: "Battery Level."  
**Access permissions:** Read, notify.
- **Remote Control Service:**
  - Battery level characteristic: "Battery Level."  
**Access permissions:** Read, notify.

In addition to these services, it is mandatory (per the Bluetooth specification) to implement the following service:

- GAP service:
  - Name characteristic: the device name.  
**Access:** Read.
  - Appearance characteristic: a description of the device.  
**Access:** Read.

*Remote Control*

We have one GATT server for the remote control. We can define the following services and characteristics:

- GAP service (mandatory):
  - Device name characteristic: "Device Name."  
**Access permissions:** Read.
  - Appearance characteristic.  
**Access permissions:** Read.
- Battery service:
  - Battery level characteristic: "Battery Level."  
**Access:** Read, notify.
- Button service:
  - On button characteristic: "On Button Press".  
**Access permissions:** Notify.
  - Off button characteristic: "Off Button Press".  
**Access permissions:** Notify.

### **Step 3: Re-use Bluetooth SIG-Adopted Services & Characteristics**

#### *Gateway*

The environment sensor, Bluetooth lightbulb, and remote control services are all custom services since there are no SIG-adopted ones that can be utilized for them. We have three devices for which we need to expose the battery level so that we can reuse the SIG-adopted battery level characteristic.

We will reuse it for each device within each device's service in the gateway GATT. We'll also re-use the mandatory GAP service.

#### *Remote Control*

For the **remote control**, we can reuse both the battery service and the mandatory GAP service.

#### Step 4: Assign UUIDs to Custom Services and Characteristics

For any **custom services** and **characteristics** within the GATT, we can use an online tool to generate UUIDs such as the [Online GUID Generator](#).

A common practice is to choose a base UUID for the custom service and then increment the **3rd** and **4th Most Significant Bytes (MSB)** within the UUID of each included characteristic.

For example, we could choose the UUID:

00000001-1000-2000-3000-111122223333

for a specific service and then

0000000[N]-1000-2000-3000-111122223333, (where  $N > 1$ ) for each of its characteristics.

The only restriction for choosing UUIDs for custom services and characteristics is that they must not collide with the Bluetooth SIG base UUID:

XXXXXXXX-0000-1000-8000-00805F9B34FB

However, following the previously mentioned common practice makes it a bit easier to relate services and their characteristics to one another.

The following tables list the services and characteristics along with their UUIDs for each of the gateway and remote control devices:

Service	Characteristic	UUID	UUID Type
<b>Environment Sensor</b>		13BB0001-5884-4C5D-B75B-8768DE741149	Custom
	Temperature	13BB0002-5884-4C5D-B75B-8768DE741149	Custom
	Humidity	13BB0003-5884-4C5D-B75B-8768DE741150	Custom
	Battery level	0x2A19	SIG-adopted
<b>Playbulb</b>		19210001-D8A0-49CE-8038-2BE02F099430	Custom
	Light status	19210002-D8A0-49CE-8038-2BE02F099430	Custom
	Battery level	0x2A19	SIG-adopted
<b>Remote Control</b>		B49B0001-37C8-4E16-A8C4-49EA4536F44F	Custom
	Battery level	0x2A19	SIG-adopted
<b>GAP</b>		0x1800	SIG-adopted
	Device Name	0x2A00	SIG-adopted
	Appearance	0x2A01	SIG-adopted

Service	Characteristic	UUID	UUID Type
<b>Button</b>		E54B0001-67F5-479E-8711-B3B99198CE6C	Custom
	ON button press	E54B0002-67F5-479E-8711-B3B99198CE6C	Custom
	OFF button press	E54B0003-67F5-479E-8711-B3B99198CE6C	Custom
<b>Battery</b>		0x180F	SIG-adopted
	Battery level	0x2A19	SIG-adopted
<b>GAP</b>		0x1800	SIG-adopted
	Device Name	0x2A00	SIG-adopted
	Appearance	0x2A01	SIG-adopted

## Step 5: Implement the Services and Characteristics Using the Vendor SDK APIs

Each platform, whether it's an embedded or mobile one, has its own APIs for implementing services and characteristics. This is left to the reader to implement for the specific platform they choose for their Bluetooth LE device or application.