# 一、蓝牙学习一（简介）

————————————————

版权声明：本文为CSDN博主「t\_guest」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/qq\_26226375/article/details/127906179

————————————————

1.简介

蓝牙分为经典蓝牙（BT-Bluetooth）和低功耗蓝牙（BLE-Bluetooth Low Energy），本次主要学习BLE。

BLE分了很多个版本，现在用的比较多的就是4.2和5.X。那4.2到5.0之间有哪些升级呢？

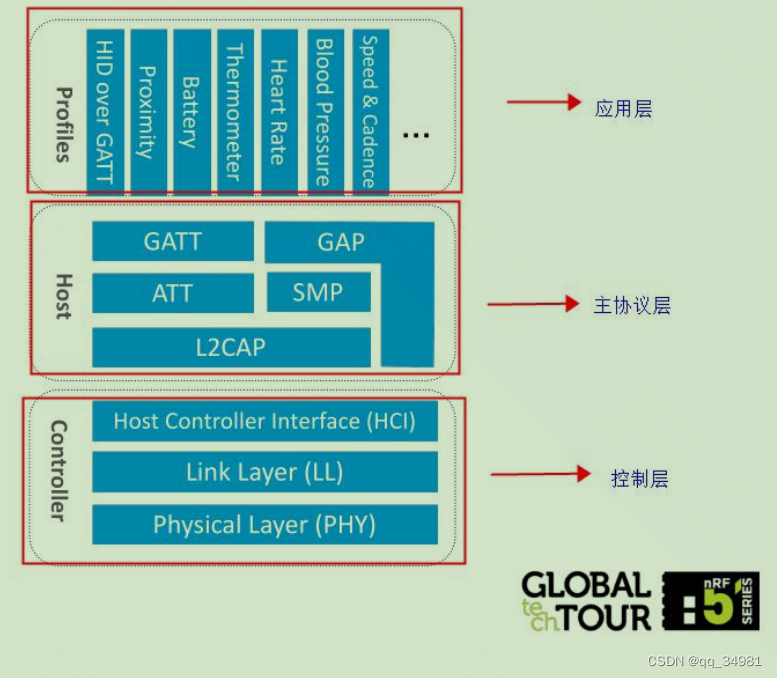
首先，4.2版本传输速度只有1Mbps，广播包最大长度为31字节。而5.0版本开始，传输速度就已经增加到2Mbps了，并且广播包的最大长度也增加为254字节。除此之外，5.X版本还增加了Mesh的功能，且通讯距离也增加至300米以上。

2.架构

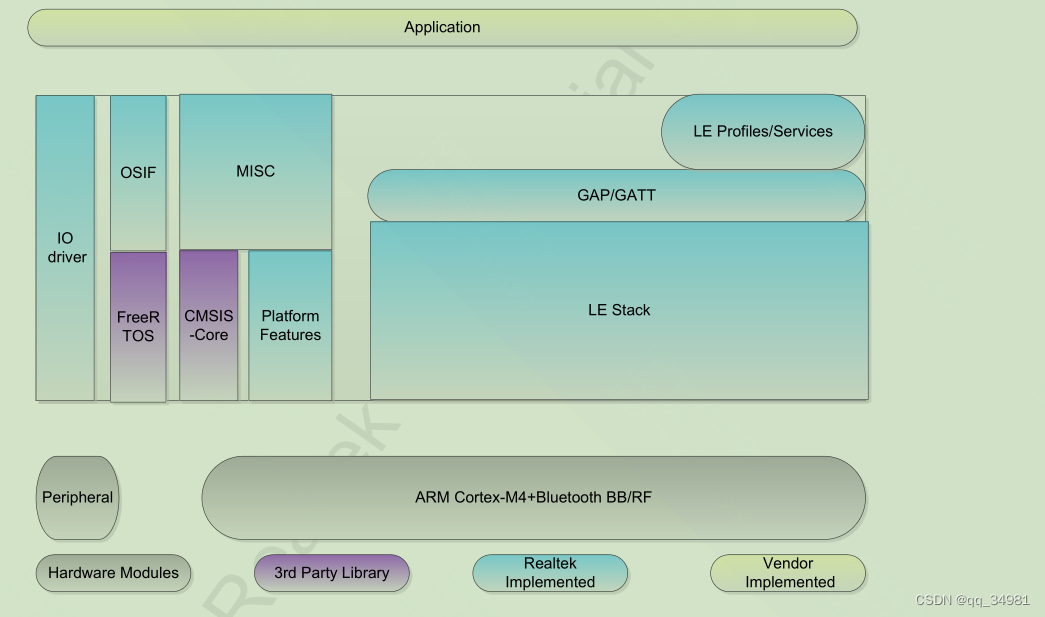
蓝牙里把**蓝牙协议的实现代码称为协议栈(protocol stack)**，BLE协议栈就是实现低功耗蓝牙协议的代码，理解和掌握BLE协议是实现BLE协议栈的前提。

每个厂商提供的SDK中包含的BLE协议栈都会有些许不同。这里主要以Nordic和RTL两个厂商来进行分析。

先看Nordic的架构



再看RTL的架构



乍一看两者的区别很大。但是再定睛一看，区别还是很大。开玩笑，其实两者区别不大，只要搞懂底层基础的概念，这些东西就都是纸老虎。

先看Nordic的架构，从上图可以看到。**一个完整的BLE程序被分为了三层，分别是应用层、主协议层和控制层。**这些层中分别还包含了各自的内容。

其中的术语，这里大概解释一下。

PHY层（Physical layer-物理层）。PHY层用来指定BLE所用的无线频段，调制解调方式和方法等。PHY层做的好不好，直接决定了整了BLE芯片的功耗、灵敏度等射频指标。

LL层（Link Layer-链路层）。**LL层是整个协议栈的核心**。也是BLE协议栈的难点和重点。像Nordic的BLE协议栈能同时支持20个Link连接，就是LL层的功劳。

**LL层要做的事情非常多**，比如具体选择哪个射频通道进行通信。怎么识别空中数据包。具体在哪个时间点把数据包发送出去。怎么保证数据的完整性。ACK如何接收。如何进行重发。以及如何对链路进行管理和控制等等。

LL层只负责把数据发送出去或者接收回来，对数据进行怎样的解析则由GAP或ATT来负责。

HCI（Host controller interface）。HCI是可选择的。HCI主要用于2颗芯片实现BLE协议栈的场合，用来规范两者之间的通讯协议和通信命令等。

（主机与蓝牙芯片之间的接口。）

**GAP层（Generic access profile-通用访问配置文件）**。GAP是对LL层payload（有效数据包）如何进行解析的两种方式的一种，而且也是最简单的一种。

GAP简单的对LL payload进行一些规范和定义，因此GAP能实现的功能极其有限。GAP目前主要用来进行广播，扫描和发起连接。

L2CAP层（logic link control and adaptation protocol-**逻辑链路控制和适配协议**）.L2CAP对LL进行了一次简单的封装，LL只关心传输的数据本身，

**L2CAP就要区分是加密通道还是普通通道**，同时还要对连接间隔进行管理。

SMP（Secure manager protocol-加密管理协议）。**SMP用来管理BLE连接的加密和安全。**如何保证连接的安全性，同时不影响用户的体验，这些都是SMP要考虑的工作。

ATT（Attribute protocol-属性协议）。简单来说，**ATT层用来定义用户命令及命令操作的数据。**比如读取某个数据或者写某个数据。BLE协议栈中，开发者接触最多的就是ATT。BLE引入了attribute概念，用来描述一条一条的数据。Attrubute除了定义数据，同时定义该数据可以使用的ATT命令，因此这一层被称为ATT层。

GATT（Generic attribute profile-通用属性配置文件）。GATT用来规范attribute中的数据内容，并运用group（分组）的概念对attribute进行分类管理。没有GATT，BLE协议栈也能跑。但互联互通就会出问题，也正是因为有了GATT和各种各样的应用profile，BLE摆脱了Zigbee等无线协议的兼容性困境，成为了出货量最大的2.4G无线通信产品。

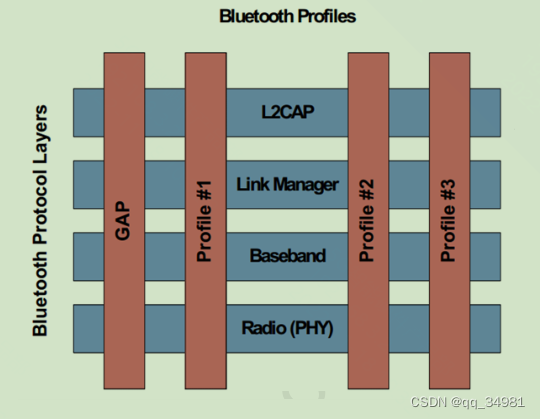
ATT和GATT的关系就像车辆和交通规则一样。ATT就是车辆，而GATT就是规范车辆该如何行驶的交通规则。只有GATT存在的时候，多个ATT见面才知道该如何形式，到底是拐弯让直行还是坐车道超车。



从上边的名字可以看到，主要分为protocol和profile。其中L2CAP、SMP、ATT属于protocol，而GAP和GATT属于profile。

profile和protocol有什么区别呢? 在蓝牙核心规范（Bluetooth Core Specification）中，profile的定义不同于protocol的定义。Protocol被定义为各层协议。而Profile从使用蓝牙核心规范中各层协议的角度，定义蓝牙应用互操作性的实现。Profile定义Protocol中的可用特性和功能，以及蓝牙设备互操作性的实现，使蓝牙协议栈适用于各种场景的应用开发。

在蓝牙核心规范中，Profile和Protocol的关联如下图:



Profile由红色矩形框表示，包括GAP、Profile#1、Profile#2和Profile#3。蓝牙核心规范中的Profile分为两种类型：GAP，途中红色矩形框所示的GAP；基于GATT的Profile，图中红色矩形框所示的Profile#123。

可能初学者看完上述之后还是云里雾里的。没有关系，我们在后续的文章中，会对这些进行单独讲解，也会再重复提到。只要跟着走，后续这些问题都会弄清楚的。加油！

# 蓝牙学习二（连接和通讯简述）

————————————————

版权声明：本文为CSDN博主「t\_guest」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：<https://blog.csdn.net/qq_26226375/article/details/127919515>

————————————————

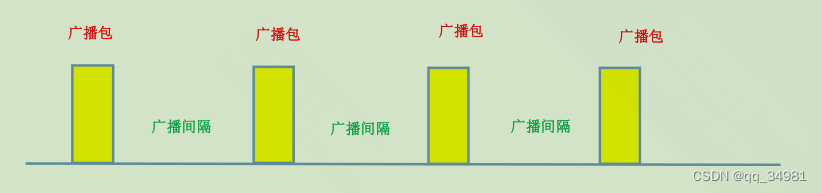
1.简介

蓝牙的通信是双向的，为了创建和维护一个BLE通信连接，在蓝牙中引入了“角色”这一概念，一个BLE设备不是主机（集中器）就是从机（外围设备）角色，这是根据是谁发起这个连接来确定的。主机（集中器）设备总是连接的发起者，而从机（外围设备）总是被连接者。整个访问与连接过程都是在GAP（Generic Access Profile-通用访问规范）进行实现的。

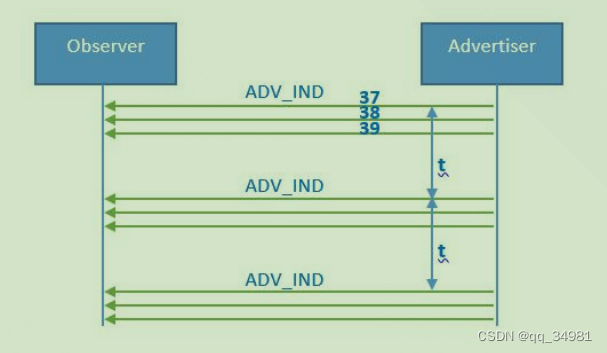
2.从机广播

从机要被主机连接，那么它就必须先被主机发现。这个时候，从机设备把自身信息通过广播的形式发射出去。

比如设备A需要先进行广播，即设备A（Advertiser）不断发送广播信号，t 为广播间隔。每发送一次广播包，我们称为一次广播事件（advertising event），因此 t 也称为广播事件间隔。广播事件时间是一段一段的，每次都会持续一段时间，蓝牙芯片也只有在广播事件期间才会打开射频模块发射广播，这个时间功耗比较高（几十毫安），其余时间蓝牙芯片都处于idle待机状态，因此平均功耗就非常低（几微安）。



当广播发出的时候，每个广播事件包含了三个广播包，分别在37/38/39三个广播信道上进行发送。广播的内容是相同的。下图observer为主机观察者，advertiser就是从机广播者。

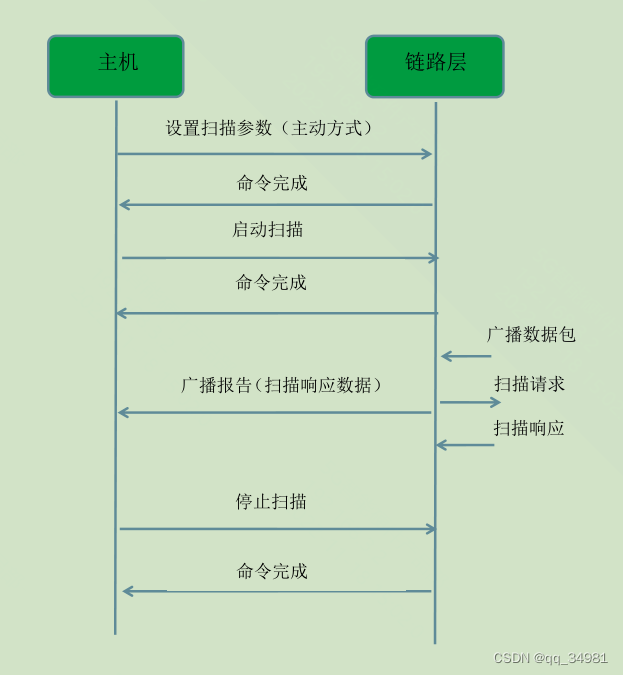


3.主机扫描

设备A不断发送广播给主机（Observer），如果主机不开启扫描窗口，是收不到设备A的广播的。主机不但要开启射频接收窗口，而且主机的射频接收窗口要跟从机的广播发送窗口匹配才行。由于这种配对成功时一个概率事件，所以手机扫到设备A也是一个概率事件。也就是说，从机（Advertiser）可能很快就被主机（Observer）发现，也有可能从机（Advertiser）要很久才能被主机（Observer）发现。

下图表示了主机主动扫描广播的过程：

主机设置自身的扫描参数，设置成功后，控制器根据设置参数开启扫描窗口。当控制器收到扫描数据包后，向主机发送给一个广播报告事件（adv\_report），该事件同样包括了链路层数据包的广播类型。因此，主机能够判断对端设备是否可以连接或扫描，并且区分出广播数据包和扫描响应数据包。



4.被动扫描

既然有主动扫描，当然也就有被动扫描。在被动扫描中，扫描者设备应该仅仅去监听广播包，而不向广播者设备发送任何数据。

一旦上面的扫描参数设置完毕，主机就可以启动扫描。如果控制器接收到的符合过滤策略和其他规划的广播数据包，则发送一个Advertising Report事件给主机。除了广播者的设备地址外，报告事件还包括广播数据包中的数据，以及接收广播数据包时的信号接收强度。可以利用该信号强度以及位于广播数据包中的发射功率，共同确定信号的路径损失，从而给出大致的范围，这个应用就是防丢器和蓝牙定位。被动扫描不需要向主机发送任何数据。

下图为被动扫描广播的过程：



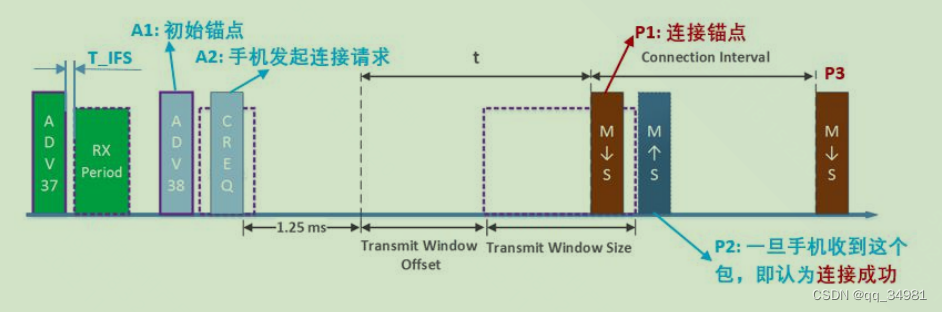
5.建立连接

手机（主机）在收到A1广播包ADV\_IND后，以此为初始点，T\_IFS时间后，给Advertiser（从机）发送一个connection request命令，即A2数据包，告诉advertiser（从机）将要进行连接，做好准备。Advertiser（从机）根据connect\_req命令信息做好接收准备。connect\_req其实就是告诉advertiser手机（主机）将在Transmit Window期间发送第一个同步包（P1），请advertiser（从机）在这段时间内打开接收窗口。设备B收到P1后，T\_IFS时间后将给手机（主机）回复数据包P2。一旦手机（主机）收到数据包P2，连接即认为建立成功。后续手机（主机）将以P1为锚点（原点），Connection Interval为周期，周期性地给设备B发送Packet。

主 A1 A2 P1 P2

↑ ↓ ↓ ↑

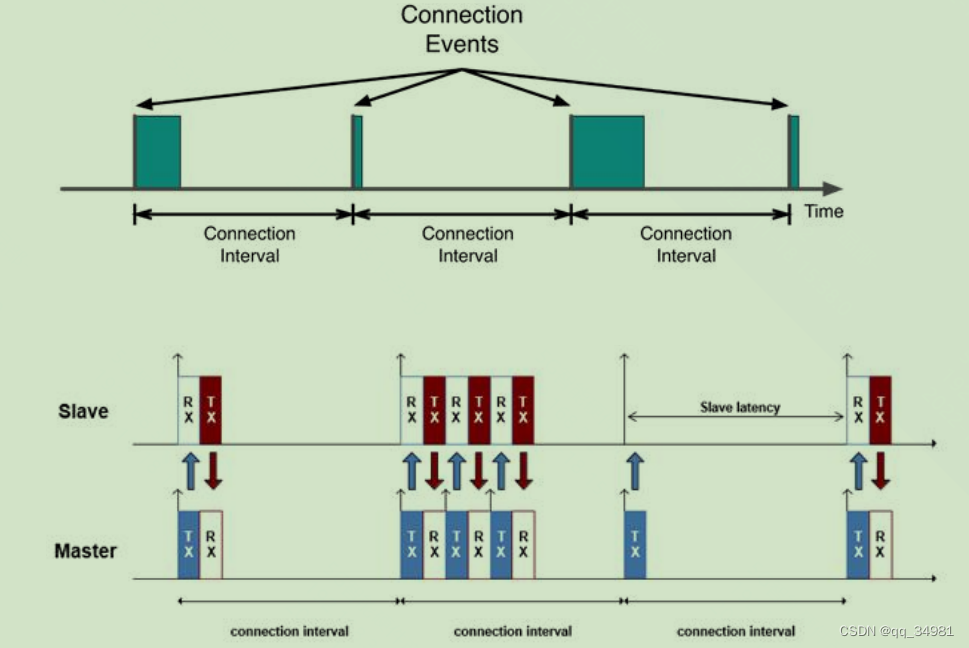
从



6.发送与接收数据

连接成功后，主机和从机在每一个connection interval开始的时候，都必须交互一次，即主机给从机发一个包，从机再给主机发一个包。整个交互过程被称为一个connection event。蓝牙芯片只有在connection event期间才把射频模块打开，此时功耗比较高，其余时间蓝牙芯片都处于idle状态，因此平均功耗非常低。

主机不可能时时刻刻都有数据给从机，所以主机大部分时间都是发的空包（empty packet）给从机。同样从机也不是时时刻刻都有数据给主机。因此从机回复给主机的包，大部分也是空包。**另外，一个connection event期间，主机可以发多个包给从机，以提高吞吐率（IOS一个连接间隔最多交互4次，安卓一个连接间隔最多交互6次）。**综上所述，连接成功后的通信时序图如下：



图中，主从数据发送的数据包TX和RX表示方向性的数据通道，也就是蓝牙的空中属性，空中操作时间都是采用蓝牙操作句柄来进行的。因为句柄能够唯一表示各个属性。空中特性的性质包括：

从机->主机方向：

通知：从机端上报数据给主机，不需要主机回复一个响应

指示：从机端上报数据给主机，需要主机端回复一个确认

通知和指示之间不同之处在于指示有应用层上的确认，而通知没有。

主机->从机方向：

写

没有回应的写

读

# 蓝牙学习三（GAP）

————————————————

版权声明：本文为CSDN博主「t\_guest」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/qq\_26226375/article/details/128017160

————————————————

1.简介

GAP（Generic Access Profile-通用访问配置文件）与应用层紧密相连，所以要想了解BLE，GAP是必须认识的东西。

在第一章中我们说过GAP层，GAP层目前主要用来进行广播、扫描和发起连接。GAP保证了不同的BLE设备可以互相发现对方并建立连接。有不清楚的，建议再回去看一下。

GAP定义了蓝牙设备如何发现和建立与其他设备的安全或不安全连接。它处理一些一般模式的业务，比如询问、命名和搜索，还处理一些安全问题，比如担保。同时还处理一些有关连接的业务，比如链路建立、信道和连接建立。GAP规定的是一些一般性的运行任务。因此，它具有强制性，并作为所有其他蓝牙应用规范的基础。GAP是所有其他配置文件的基础，它定义了在蓝牙设备之间建立基带链路的通用方法，除此之外，GAP还定义了下列内容：

必须在所有蓝牙设备中实施的功能

发现和链接设备的通用步骤

基本用户界面术语

**GAP是应用层能够直接访问BLE协议栈的最底层，它包括管理广播和连接事件的有关参数。**GAP模块代表了所有蓝牙设备的共用基础功能，如传输，协议或者应用规范所使用的模式和访问过程。**GAP的服务包括设备发现，连接方式，安全，认证，关联模型和服务发现等。**

总之一句话，GAP是保证BLE设备能互相发现和连接的规范。GAP就好比一种语言，大家都说同一种语言才能相互听懂，相互知道说的是什么。否则就只能大眼瞪小眼。

BLE协议栈的GAP层负责处理设备的接入方式和过程，包括设备发现，链路建立，链路终止，启动安全功能，设备配置。

GAP还定义了以下四种角色类型：

**广播者（Broadcaster）：**用于只通过广播发送数据的应用。广播发送者，是不可连接的设备。广播者角色只是广播信息而不接受信息（ibeacon类似应用）。

**观察者（Observer）**：用于接收广播数据的应用。扫描广播，不能够启动连接。观察者只监听空气中的事件（抓包器）。

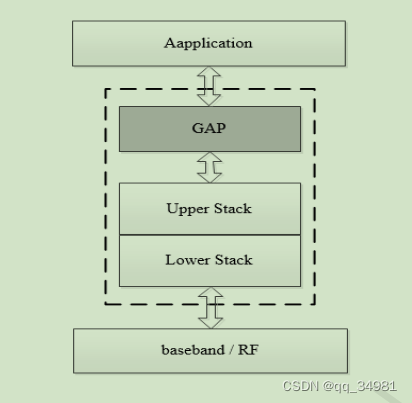
**外围设备（Peripheral）**：用于通过广播发送数据并且可以建立链路的应用。广播发送者，是可连接的设备，在单一链路层连接时作为从机。

**中央设备（Central）**：用于接收广播数据并且建立一条或多条链路的应用。扫描广播启动连接，在单一或多链路层连接时作为主机，支持三个同时连接。是吗？？

2. GAP结构

2.1GAP的位置

GAP层作为蓝牙协议层的组成模块，如下图所示，虚线框内的部分为蓝牙协议层。Application在蓝牙协议层之上，baseband/RF位于蓝牙协议层之上。GAP层给application提供访问Upper stack的接口。



2.2GAP的功能

GAP提供如下功能

广播（Advertising）：设置和获取广播参数，启动和停止

扫描（Scan）：设置和获取扫描参数，启动和停止

连接（Connection）：设置连接参数，创建连接，终止连接，更新连接参数

配对：设置配对参数，启动配对。使用passkey entry方式时输入/显示passkey，删除绑定设备密钥

密钥管理：根据设备地址和地址类型查找密钥，保存/加载绑定设备信息的密钥，解析random address

其他：

设置GAP公共参数，例如外貌和名字。

获取支持的最大BLE链路数目

修改白名单（white list）

生成/设置本地设备random address

配置本地设备identiy address

广播

中央设备能够与外围设备建立连接，外围设备必须处于广播状态，它每经过一个时间间隔发送一次广播数据包，这个时间间隔被称为广播间隔。他的范围是20ms到10.24s。广播间隔影响建立连接的时间。

中央设备在发送一个连接请求来发起连接之前，必须接收到一个广播数据包。外围设备发送一个广播数据包之后一小段时间内只监听连接请求。

一个广播数据包最多能携带31字节的数据，它通常包含用户可读的名字、关于设备发送数据包的有关信息、用于表示此设备是否可被发现的标志等类似的标志。

当中央设备接收到广播数据包后，它可以发送请求更多数据包的请求，称为扫描回应。如果它被设置成主动扫描。外围设备将会发送一个扫描响应作为对集中器请求的回应。扫描回应最多可以携带31字节的数据。

扫描

扫描是中央设备监听广播数据包和发送扫描请求的过程。它有2个定时参数需要特别注意：**扫描窗口和扫描间隔。**

对于每一个扫描间隔，中央设备扫描的时间等于一个扫描窗口，这就意味着如果扫描窗口等于扫描间隔，那么集中器将处于连接扫描之中。扫描窗口和扫描间隔之比称为扫描占空比。

连接

中央设备和外围设备第一次交换数据定义为连接状态。在一个连接状态中，中央设备将会在一个特定定义的间隔从外围设备请求数据，这个间隔称为连接间隔，它由中央设备决定并应用于连接，但是外围设备可以发送连接参数更新请求给中央设备。根据蓝牙核心规范，连接间隔必须在7.5ms到4S之间。

如果外围设备在一个时间帧内没有回应集中器的数据包，被称为连接监管超时，连接则被认为丢失。

可以通过在每一个连接间隔中传输多个数据包以获得更高的数据吞吐量（IOS一次连接间隔最多可以交互4次，安卓为6次）。在蓝牙5.0里，每个传输数据包最多可以写到247个字节的应用数据。但是如果电流消耗是重点，同时外围设备也没有数据要发送，则可以忽略一定数量的连接间隔，这个忽略连接间隔的数目称为从机延迟（从机潜伏）。

**在一个连接中，除了广播信道，设备间在频带的所有信道中进行通信。**

【广播、扫描、连接】

2.3 GAP层设备状态

GAP层设备状态由advertising、scan、connection组成。每个状态都有相应的子状态，

advertising

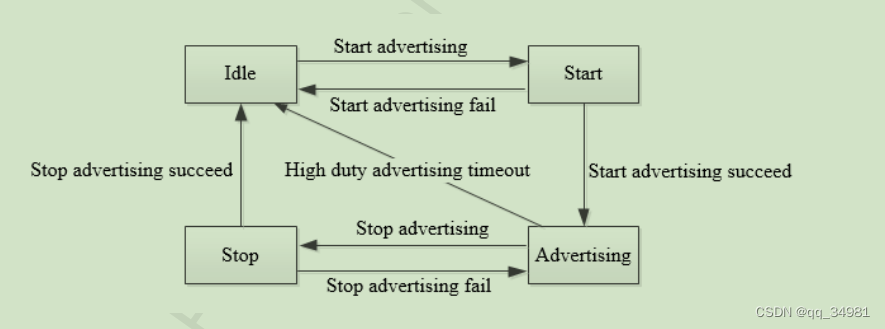
advertising有四个子状态，分别为idle、start、advertising和stop。

idle：默认状态，不发送广播

start：在idle启动广播后，启动广播的流程尚未完成。start为临时状态，若成功启动广播，则进入广播状态。若启动失败，则回到idle状态

advertising：成功启动广播后，设备发送广播。若广播类型为快速广播，一旦超时，则进入idle状态。

stop：在停止广播之后，停止广播的流程尚未完成。stop为临时状态，若成功停止广播，则进入idle状态。若停止失败，则回到advertising状态。



scan

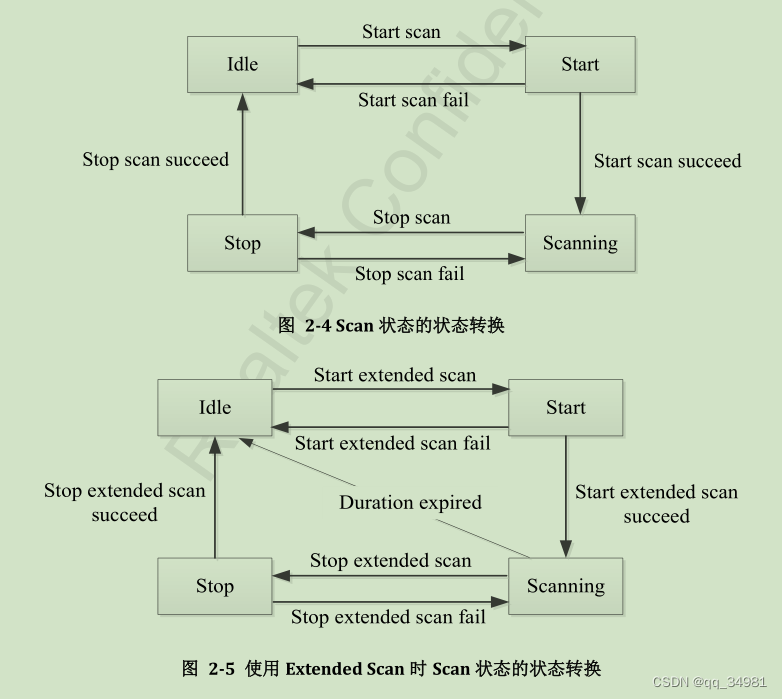
scan状态有四个子状态，idle、start、scanning和stop。

idle：不进行scan

start：在idle状态启动scan之后，启动scan的流程尚未完成。start状态为临时状态，若成功启动scan，则进入scanning状态。若启动scan失败，则返回idle状态。

scanning：成功启动scan。在此状态下，设备进行scan，接收advertisement。使用延时扫描时，若持续时间参数非零，则scan时间超时后，进入idle状态。

stop：在scanning停止之后，停止scan的流程尚未完成。stop状态为临时状态，若成功停止scan，则进入idle状态。若停止scan失败，则返回scanning状态。



3.GAP配置

3.1GAP安全模式

GAP在启动连接时会负责BLE连接的安全功能。只有对通过身份验证的连接而言的某些数据是可读或可写的。一旦形成一个连接，两个设备可以通过一个过程进行连接，称为配对。**进行配对时，密钥建立加密或认证的链接。**例如当两个设备要建立配对关系时，外围设备需要中央设备提供密钥以完成配对。这个密钥可以是固定值或随机值。中央设备发送正确的密钥后，两台设备交换安全密钥并验证链接。

在许多情况下，两个设备会经常建立连接和断开连接。BLE具有一个安全功能，允许两个设备在配对的时候给对方一个长久的安全密钥。此功能被称为绑定。绑定后的两个设备每次重连时能够迅速重新确立加密和认证，并且不需要经过充分的配对过程，只需要提供他们存储的长期密钥信息即可。

加密方式大概分为一下几种

不允许连接

无安全要求（可直接连接）

加密连接

MITM（man in the middle-中间人攻击）保护

LESC（LE secure connections-）加密

签名

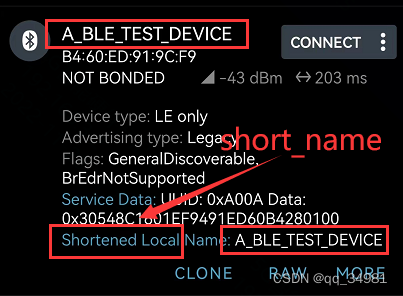
【配对、绑定】

3.2蓝牙设备名称与外观

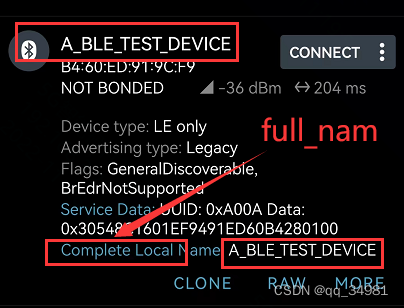
设备名称

当我们需要查找蓝牙设备的时候，大部分情况都是通过设备名称来查找的。协议栈也是支持用户自行修改设备名称的。在设置的时候可以设置short\_name和full\_name。

short\_name：可以成为可选名称，字面意思上可以理解为名称的简写。当一个名称比较长的时候，由于广播的信息容量是有限的。要留空间放置其他信息，没有那么多空间显示全名，因此APP显示的时候为简称。



full\_name：如果名称不是很长，广播数据可以容纳。那么可以通过GAP设置全名显示。



设备外观

当我们在搜寻蓝牙的时候，不但可以找到蓝牙设备的名称，还可以查看到蓝牙设备的外观。外观是什么，就决定了我们找的设备是什么。例如未知、键盘、鼠标、温度计、手表等等。

4.GAP连接

在一个典型的蓝牙系统中，外围设备发送具体的广播数据让所有中央设备知道它是一个可连接的设备。广播内容包含设备地址，还包含了一些额外的数据，比如设备名称。中央设备接收到广播之后发送一个搜索请求给外围设备，外围设备答复一个搜索应答，这就是设备发现的过程。

【（发现过程：）从机：->广播，<-搜索请求，->搜索应答：主机】

设备发现之后，中央设备就知道外围设备是一个可连接的设备。中央设备可以发送一个建立链接的请求给外围设备，开始进行主机和从机设备的互联。

在一个连接中，包含了如下参数：

连接间隔：

在一个BLE连接中，跳频机制需要被使用，这样两个设备之间可以在一个特定的通道上进行数据收发。在一个特定的时间之后，会跳到一个新的通道上，LL层负责通道切换。这个发现设备并收到数据的事件，**就被称为连接事件**。

尽管没有应用程序数据需要收发，两个设备之间仍然需要交换链路层数据来保持连接。连接间隔是两个连接事件之间的时间，使用一个单元是1.25ms的步进。连接间隔从最小值6（7.5ms）到最大值3200（4.0S）。

不同的应用需要不同的连接间隔，一个长时间的连接间隔将会节约更多的能量，因为设备可以在两个连接事件之间睡眠更长的时间。但是他会导致数据发送不及时。如果有数据要发送，那么只能等下一个连接事件到来时才能被发送。

值得注意的是，**连接间隔参数是中央设备进行设置，外围设备只是负责执行。【怎么设置的？】**

从机潜伏周期：

这个参数描述了从机跳过连接事件的次数。这使外围设备具有一定的灵活性，如果它不具有任何数据传送，它可以选择跳过连接事件，并保持睡眠，从而提供了一些积蓄力量。这一决定取决于外围设备。也可以成为从机延迟，从设备能够忽略主设备的链接事件的最大值。比如：从设备延迟为6的话，那么从设备每隔6个锚点可以监听到主设备发送过来的数据包。如果从设备延迟为0，从设备在每个锚点都可以监听主设备的数据包。

监督超时：

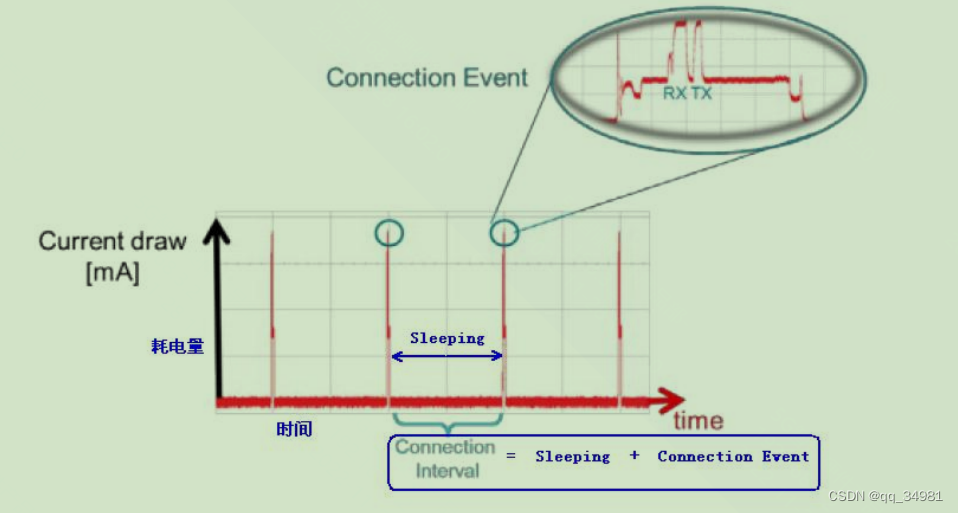
这是两个成功的连接事件之间间隔的最大值。如果超过这个时间还未出现成功的连接事件，那么设备将会考虑失去连接，返回一个未连接状态。这个参数值使用10ms的步进。监督超时时间从最小10（100ms）到最大3200（32.0s）。监督超时时间必须大于有效连接事件。

有效连接事件时间 = 连接间隔 \* （1 + 从机延迟值）。

连接间隔、潜伏周期的不同组合，有好有坏，而如何组合，需要根据实际的应用场景而定。例如短连接间隔，功耗高，数据吞吐量高，发送等待时间短。长链接间隔，功耗低，数据吞吐量低，发送等待时间长。低潜伏值，从机在没有数据发送的情况下功耗高，但从机可以快速的收到主机的数据。高潜伏值，从机在没有数据发送的情况下可以低功耗。但从机无法及时收到主机的数据，但主机能及时收到从机的数据。

在某些情况下，中央设备请求与外围设备建立链接包含的连接参数对外围设备而言是不利的。此时，外围设备可以在连接过程中改变连接参数。这个取决于外围设备的应用程序。**外围设备可以通过设置连接参数更新请求，请求中央设备改变连接参数。**这个请求包含四个参数：最小链接间隔、最大链接间隔、从机潜伏周期、连接超时时间。这些值代表了外围设备针对连接的期望参数，连接间隔是以范围的形式提供的。当中央设备接收到这个请求后，中央设备有权利决定是接受还是拒绝这些参数。

4.1 连接间隔



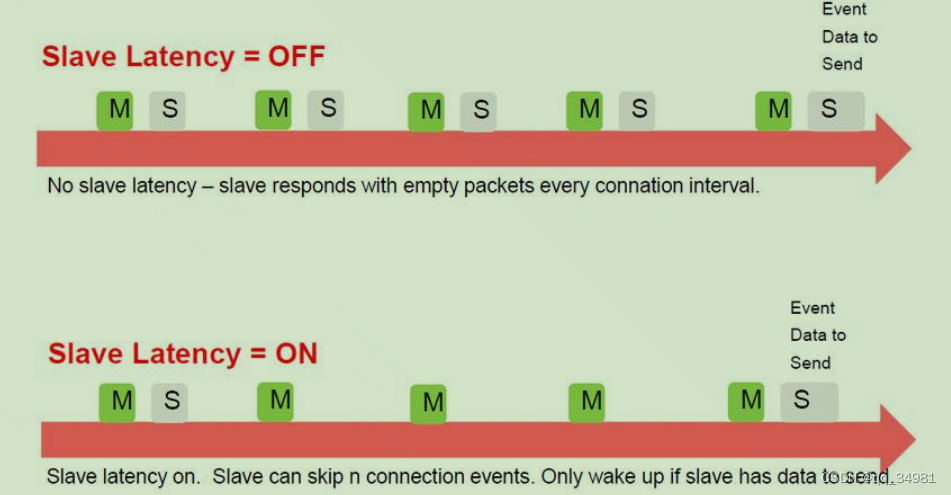
每个尖刺就是一次连接事件（Connection events），两个连接事件的中间为sleeping休眠时间。从图中也可以看出，在每次从机的连接事件中，首先是射频接收，然后才是射频发送。其实也是为了保证主机和从机的射频时序是相匹配的。设备在建立链接之后的大多数事件都是处于sleeping。这种情况下耗电量比较低，只有在连接事件（connection events）中，耗电量才相对较高，但持续时间很短。这也是为什么BLE非常省电的原因之一。

在两个connection interval连接间隔之间，BLE会使用跳频机制。两个设备使用特定的信道发送和接收数据，然后过一段时间后，再使用新的信道（BLE协议栈的LL链路层处理信道的切换）。两个设备在切换信道后发送和接收数据被称为一个连接事件。尽管没有应用数据被发送或接收，两个设备仍旧会交换链路层数据（空包 Empty PDU）来维持连接。在每个连接事件中，都是由主机发送，从机回复。

当大数据传递时，通讯数据包是连续传递的。主机会选择最小连接间隔来进行通讯。无数据传递时，主机会选择最大链接间隔来定期询问从机状态，以保持连接不中断。当从机向主机发送一个连接参数更新请求后，主机会按照请求，修改连接间隔。但这个时间需要在主机设置的最大和最小连接间隔之内。

4.2从机潜伏周期

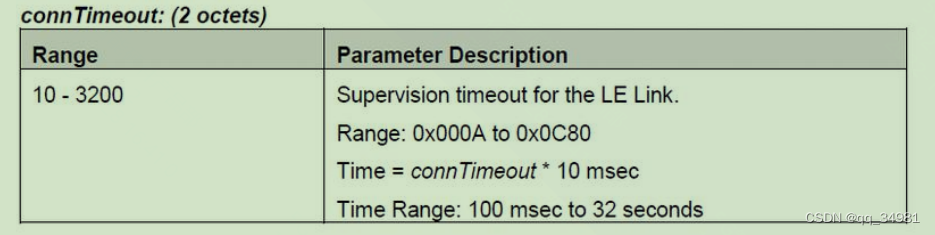
从机设备没有数据要发时，跳过一定数目的连接事件Connection Event。范围0-499。设置为0则每次连接间隔都连接。数字越小，通讯速度越快，但功耗越高。注：如果在距离远或干扰大的情况下，当无数据通信时，这个值设置大，可以减少掉线的概率。比如防丢器：如果这个参数设置为0，表示规定时间内必须响应从机，不然就以为是蓝牙断开了。如果设置为3，假设信号不好时，中间丢了3个包，只接受到了1个就表示连接正常。因为他会跳过其中3个，保证了不掉线的概率。



允许从机在没有数据发送的情况下，跳过一定数目的连接事件。在这些连接事件中，不必回复主机的包。这样就更省电。

4.3连接超时

超时时间，就是两个设备在连接的这段时间没有发生通讯而导致连接自动断开的值。Range(10ms-32s)。用在信号不太好的情况下，给对方一点时间，超过这个时间通讯就建立失败。



连接间隔、从机潜伏周期以及超时时间这三者必须满足以下公式：

超时时间 > (1 + 从机潜伏周期) \* 连接间隔。否则连接就会不正常断开。

这三个连接参数不同情况下对通信速率和功耗的影响：

连接间隔缩短，主机和从机通信更加频繁。提高了数据吞吐速度，缩短了数据发送的时间，增加了功耗。

连接间隔增长，通信频率降低，数据吞吐速度降低，增加了数据发送的时间，降低了功耗。

潜伏间隔减少，每次连接事件中都回复主机包，功耗增加，数据发送速度提高。潜伏间隔增加，功耗降低，数据发送速度降低。

蓝牙协议栈定义了主机决定连接参数的值（连接间隔、潜伏周期、超时时间），从机可以请求更新这些参数，但是由主机决定是否接受，接受的值是多少。所以会出现手机接受参数后和从机请求的参数有偏差的情况，甚至是拒绝（IOS）。安卓和IOS都是在手机和设备建立链接时就会默认设置这些参数，**APP开发时无法修改这些参数，这些默认参数是由手机厂商决定的。**

# 蓝牙学习四（广播）

————————————————

版权声明：本文为CSDN博主「t\_guest」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：<https://blog.csdn.net/qq_26226375/article/details/128091893>

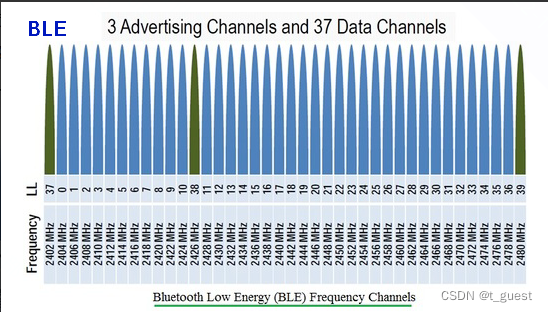
————————————————

1.简介

什么叫做广播，顾名思义就像广场上的大喇叭一样，不停的向外传输着信号。不同的是，大喇叭传输的是音频信号，而蓝牙传输的是射频信号。

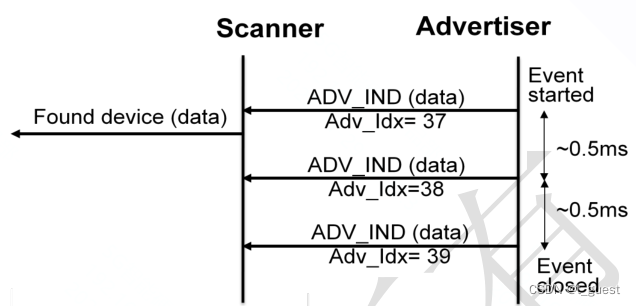
BLE使用的是无线电波传递信息，就是将数据编码，调制到射频信号中发射。BLE使用的射频频率是2.4GHz。跟WIFI、Zigbee等协议使用的是同一频段。

那如何做到使用同一频段而有不相互干扰呢？首先要知道的是2.4G指的不是某一个频率，而指的是一个频段（2400MHz-2483.5MHz）。在这个频段内每隔2M为一个信道，共40个信道。2.4G频段是一个用于短距离，无须执照使用的开放频段。意思就是可以免费使用。为了不占用更多的资源从而造成相互干扰，每个设备在使用时，同一时刻，只会在一个信道进行工作，不会占用其他信道。一个BLE设备，在任一时刻，只能选择40个信道之中的一个进行发射或监听。



BLE将信道划分为广播信道和数据信道。广播信道只有3个，37、38、39。剩下的37个信道全都是数据信道0-36。

在广播事件中，每一个广播事件都会在3个广播信道中进行数据传输，而且每一个事件都是从最小的信道编号开始传输。也就是说当广播事件来了，数据包从广播信道37、38、39中依次进行传输。



2. 广播间隔

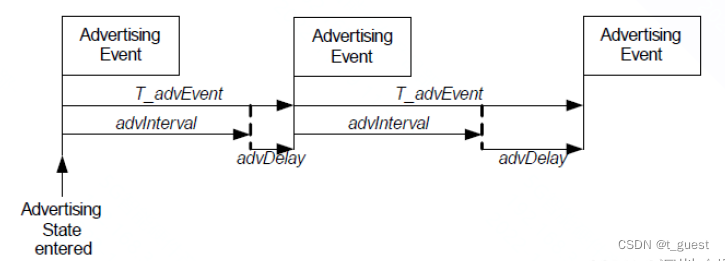
设备每次广播时，会在3个广播信道发送相同的报文。这些报文报文的动作被称为一个广播事件。除了定向广播外，其他广播事件均可以选择20ms-10.28s不等的间隔。通常，一个广播中的设备会每一秒广播一次。两个相邻的广播事件之间的时间被称为“广播间隔”。

设备周期性的发送广播会有一个问题：由于设备间的时钟会不同程度的漂移，**两个设备可能在很长一段时间同时广播而造成干扰。**为了防止这一情况的发生，除定向广播外的其他广播类型，发送时间均会有些许波动。实现方式为，在上一次广播事件后加入“0-10ms"的随机延迟。这意味着，即使两个设备广播间隔相同，并在相同信道及时间点上发送造成了冲突，但他们发送下一个广播事件时也很大概率不会冲突。

所以，两个相邻的广播事件之间的时间间隔T\_advEvent为：

T\_AdvEvent = advInterval + advDelay

其中，advInterval必须是0.625ms的整数倍，范围是20ms-10.24s之间。对于可扫描非定向广播和不可连接非定向广播者两种广播类型，该值最好不小于100ms，即160个0.625ms。advDelay是LL层分配的一个随机数，范围为0-10ms。



在实际的设置中，通过设置Advertising\_Interval\_Min（最小广播间隔）和Advertising\_Interval\_Max（最大广播间隔）这两个参数来调整广播间隔。都是以0.625ms为单位。如果要固定广播间隔为某一个值，需要将这两个参数设置为同一个值即可。

3.广播类型

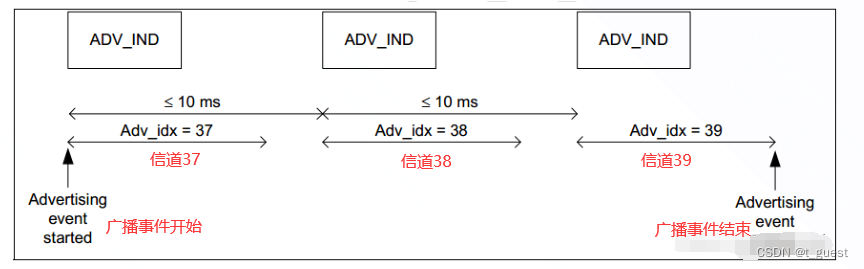
3.1 非定向可连接广播事件（ADV\_IND）

ADV\_IND就是链路层通过广播信道发送广播的事件。发送的PDU（Protocol Data Unit-协议数据单元）是ADV\_IND\_PDU-通用广播报文。这个报文发送之后可以接收由扫描者发送的SCAN\_REQ\_PDU-扫描请求，或者由发起者发送的CONNECT\_REQ\_PDU-连接请求。而接收后链路层需要在同一个信道上进行扫描或回复发起者的应答。当接收的数据报文不符合广播滤波协议，要么就用下一个广播信道进行广播，要么就停止广播事件。如果接收到的SCAN\_REQ\_PDU通过了滤波协议，那么广播者需要在150±2us内在同一信道回复SCAN\_RSP\_PDU-扫描应答报文。如果接收到CONNECT\_REQ\_PDU，则进入连接状态，这个时候并不需要进行应答。

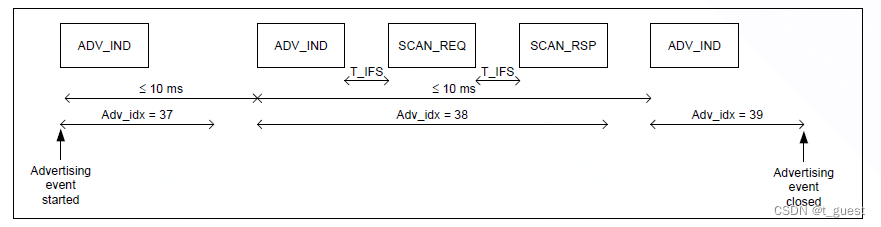
需要注意的是，一个广播事件中，相邻两个ADV\_IND\_PDU之间的时间需要不大于10ms。

接下来分类一下此类广播事件中广播包的发送情况。

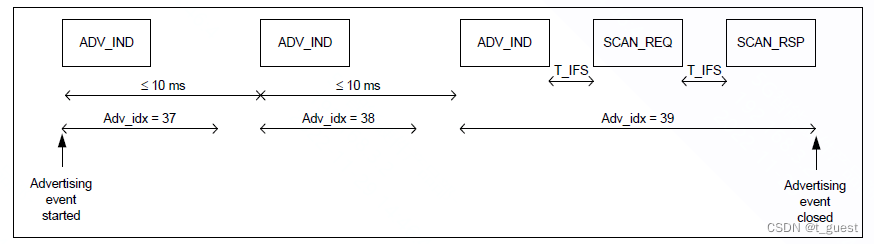
（1）仅仅有广播PDUS



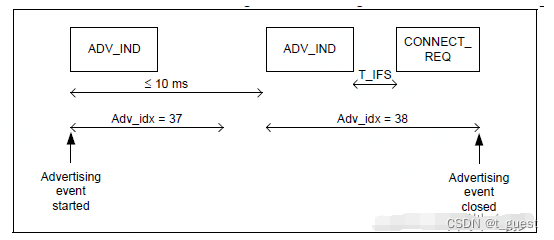
（2）在广播事件中有SCAN\_REQ\_PDUS和SCAN\_RSP\_PDUS。

 注：当有扫描请求包再广播事件中的中间信道上收到时，T\_IFS（帧间隔）为150us。

（3）在广播事件的结尾有SCAN\_REQ和SCAN\_RSP。



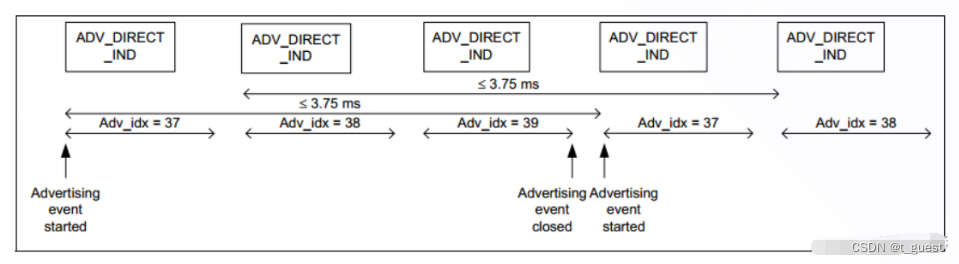
（4）在广播事件的中间接收到CONNECT\_REQ-连接请求包。没有应答



3.2 定向可连接广播事件（ADV\_DIRECT\_IND）

这个广播是为了快速建立链接。这种报文包含两个地址：广播者地址和发起者地址。发起者收到发给自己的定向广播报文后，可以立刻发送连接请求事件作为回应，并立刻进入连接状态。

定向广播事件有特殊的时序要求。完整的广播事件必须每3.75ms之内重复一次，即3.75ms内在37、38、39三个广播信道上全部发送一次报文。这样的方法使得扫描设备只需要扫描3.75ms即可收到定向广播设备的消息。



如果按照定向广播的要去持续发送报文的话，广播信道将充斥着大量的定向广播报文。如此其他设备将无法进行广播。所以，蓝牙协议规定，定向广播不能持续1.28S以上。如果主机没有主动要求停止，或者连接没有建立，控制器会自动停止广播。一旦超过1.28S，主机只能使用通用广播让其他设备连接。

3.3 非定向不可连接事件（ADV\_NONCONN\_IND）

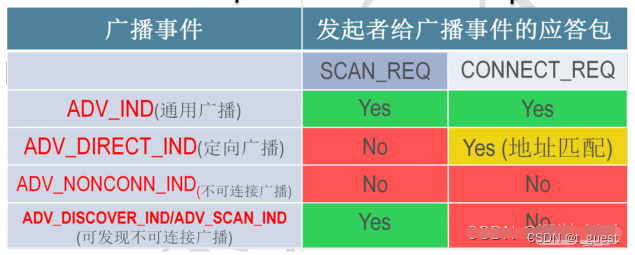
该广播的时间要求与通用广播事件相同。此外，该事件只向外发射广播报文，但是不可以被连接，也不接收任何信息。是唯一一个只有发射而没有接受的广播类型。ibeacon发出的就是这种类型的广播。

3.4 非定向可发现不可连接事件（ADV\_DISCOVER\_IND/ADV\_SCAN\_IND）

该广播的时间要求与通用广播事件相同，应答也是SCAN\_REQ和SCAN\_RSP。这个广播和通用广播的区别是，它不能建立连接。

注：所谓的定向和非定向针对的是广播对象，如果是针对特定的对象进行广播（在广播包PDU中包含目标对象的MAC），则为定向广播。反之为非定向广播。可连接和不可连接指的是是否接受连接请求。如果是不可连接的广播类型，它将不应答连接请求报文。可扫描广播类型会回应扫描请求。

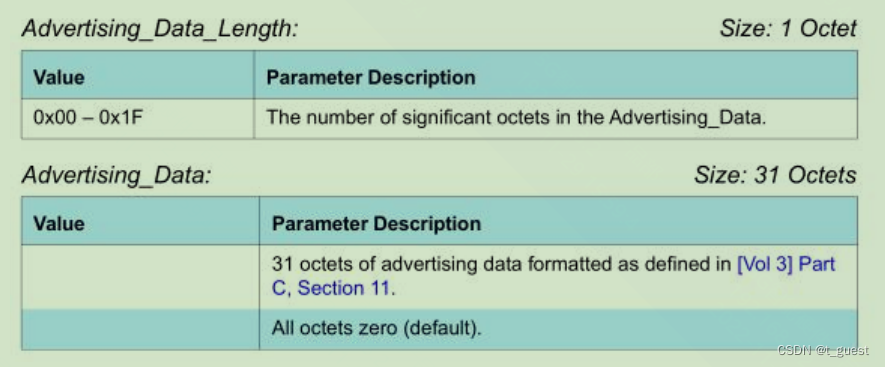
不同的广播类型对应的扫描请求和连接请求如下图：



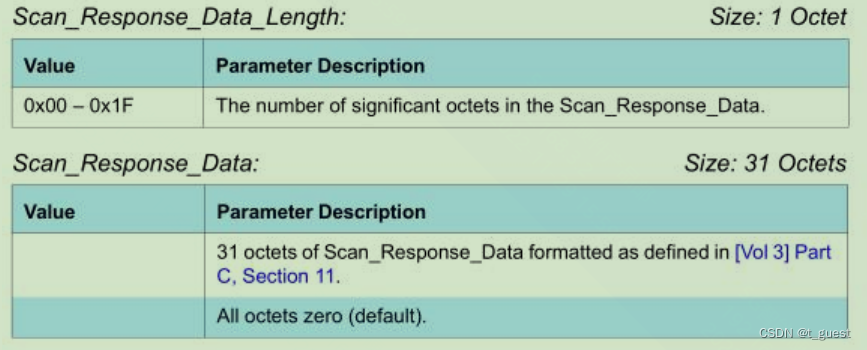
4. 广播响应包

广播包有两种：广播包（Advertising Data）和响应包（Scan Response）。其中广播包是每个设备必须广播的，而响应包是可选的。

广播包在蓝牙5.0协议栈核心中介绍如下：



应答包介绍如下：



每个包都是31字节，数据包中分为有效数据（significant）和无效数据（non-significant）

有效数据部分：包含若干个广播数据单元，称为AD Structure。AD Structure的组成是：第一个字节是长度值Len，表示接下来的Len个字节是数据部分。数据部分的第一个自己表示数据的类型AD Type，剩下的len-1个字节是真正的数据AD Data。

无效数据部分：因为广播包的长度必须是31个字节，如果有效数据部分不到31字节，则剩余部分用0补全。

广播响应包是为了给广播一个额外的31字节数据，用于主机在主动扫描情况下，反馈数据使用。

# 蓝牙学习五（广播包分析wireshark）

软件工具：wireshark

硬件抓包工具：nrf52840 dongle

# 蓝牙学习六（GATT）

————————————————

原文链接：<https://blog.csdn.net/qq_26226375/article/details/128152154>

————————————————

1.简介

GATT-Generic Attribute profle-通用属性配置文件。GATT层是传输真正数据所在的层。包括了一个数据传输和存储架构以及其基本操作。GATT用来规范attribute中的数据内容，并运用group（分组）的概念对attribute进行分类管理。没有GATT，BLE协议栈也能跑。但互联互通就会出问题，也正是因为有了GATT和各种各样的应用profile，BLE摆脱了Zigbee等无线协议的兼容性困境，成为了出货量最大的2.4G无线通信产品。

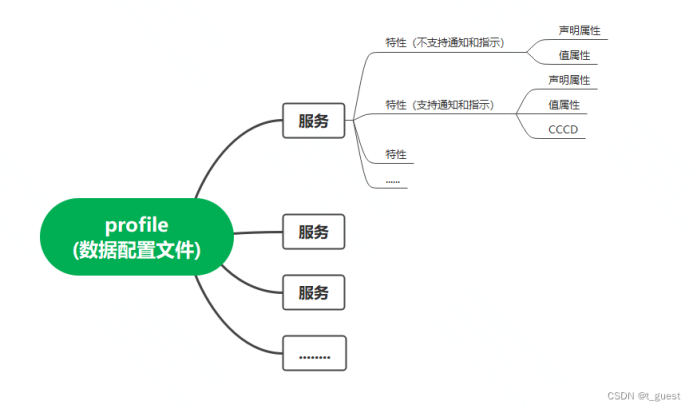
2.角色

GATT定义了两类角色：服务端（server）和客户端（client），GATT角色无需和GAP角色绑定，但是可能由更高层的规范进行指定。

除了GAP定义角色外（GAP定义广播者，扫描者，外围设备，中央设备），BLE还定义了另外2种角色：GATT服务端和GATT客户端。这两种角色完全独立于GAP的角色。提供数据的设备被称为GATT服务端，访问GATT服务端而获得数据的设备被称为GATT客户端。即一个设备可以同时作为客户端和服务端。通常情况下，外围设备为服务端，提供数据。手机作为客户端，访问数据。

3.GATT的规范层次

一个GATT服务器通过一个称为属性表的表格组织数据，这些数据就是用于真正发送的数据。

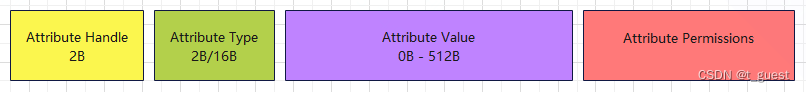


3.1属性（Attribute）

属性协议（Attribute Protocol）简称ATT。是GATT和GAP的基础。

就属性来说，一个属性其实就是一条数据，属性是BLE数据提供单元，也是蓝牙空中传播数据的最上层，BLE开发过程中接触最多的就是这一层。

一个属性包含了句柄、UUID、值。数据的结构如下图：



3.1.1Attribute Handle

属性句柄（Attribute Handle）占2个字节，犹如指向属性实体的指针，对端设备可以通过属性句柄来访问该属性。

属性句柄值的范围：0x0000 ~ 0xFFFF。

3.1.2 Attribute Type

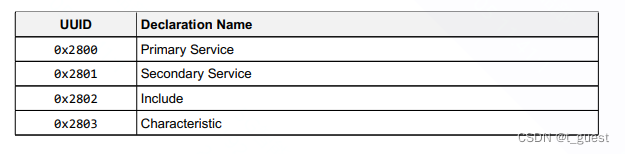
属性类型（Attribute type）是用来区分当前属性是服务项还是特征值等，一般用UUID来表示。

**UUID（universally unique identifier，通用唯一识别码）**是一个软件构建标准，并非BLE独有的概念，**一个合法的UUID，一定是随机的，全球唯一的，**不应该出现两个相同的UUID。但是在一个GATT表中可能有许多属性，这些属性可能有相同的UUID。

那有可能是重复的吗？？

分类

BLE的属性类型主要有四个大类



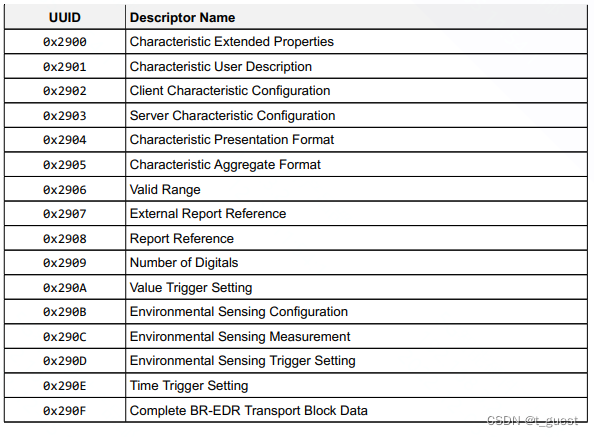
Primary Service（首要服务项）：0x2800

Secondary Service（次要服务项）：0x2801

Include（包含服务项）：0x2802

Characteristic（特征值）：0x2803

常用的UUID：



Characteristic Extended Properties（特性扩展性质）：0x2900

Characteristic User Description（特性用户描述）：0x2901

Client Characteristic Configuration（客户端特性配置）：0x2902

Server Characteristic Configuration（服务器特性配置）：0x2903

Characteristic Presentation Format（特性表示格式）：0x2904

Characteristic Aggregate Format（特性聚合格式）：0x2905

客户端特性配置描述符：支持通知或指示的特性必须使用该描述符

取值范围

0x1800 – 0x26FF ：服务项类型

0x2700 – 0x27FF ：单位

0x2800 – 0x28FF ：属性类型

0x2900 – 0x29FF ：描述符类型

0x2A00 – 0x7FFF ：特征值类型

长度

UUID即可以是2字节也可以是16字节。因为一些常用的UUID，为了减少传输的数据量，BLE协议做了一个转换约定。给定一个固定的16字节模板，只设置2个字节为变化量，其他为常量。2字节的UUID在系统内部会被替换，进而转换成保准的16字节的UUID。

UUID模板为：

IMG_256

而2字节的UUID则替代上边的XXXX。例如心率测量特性使用0X2A37作为它的16位UUID，因此它完整的16字节的UUID为：

0x00002A37-0000-1000-8000-008005F9B34FB。

蓝牙技术联盟所用的基本UUID不能用于任何定制的属性、服务和特性。对于定制的属性，必须使用完整的16字节UUID。

3.1.3 Attribute Value

属性值，就是该条属性数据包含的有效负载。？？

3.1.4 Attribute Permissions

属性权限/属性许可，作用是保护Attribute Value。

Attribute Permissions不会直接在空中包中体现，而是隐含在ATT命令的操作结果中。

权限属性：

访问权限（Access Permission）-只读，只写，读写

禁止访问权限（No Access）-禁止读或者写

加密权限（Encryption Permission）-加密，不加密

认证权限（Authentication Permission）-需要认证，无需认证。认证是指相互确认对方身份。完成认证流程的两个设备，双方建立信任关系，二者之间的通信通道即可认为是安全的。BLE中，认证过程就是配对。

授权权限（Authorization Permission）-需要授权，无需授权

签名权限（Signed Permission）-签名后才能读或写，这个用得比较少

认证和授权功能很容易混淆，其英文拼写也很相似。从上面的概念上看，授权要求设备必须是可信任的，因此授权的管控等级要高于认证--认证的设备未必被授权，授权的设备一定是认证的。理解二者关系后，需要引入一个概念：Trusted Device（可信任设备）一个没有经过认证的设备，被称为Unknown Device（未知设备）；经过了认证后，该设备会在绑定设备信息中被标记为Untrusted，被称为Untrusted Device（不可信设备）；经过了认证，并且在绑定信息中被标记为Trusted的设备被称为Trusted Device（可信设备）。

授权要求设备为Trusted Device（可信任设备）。在实际使用中，经过配对以后设备即为Untrusted Device--认证，在代码中调用API可以设置设备为Trusted Device--授权。

大部分的空中操作事件都是采用句柄来进行的，因为句柄能够唯一识别各个属性。如何使用特性依据他的性质，特性的性质包括：

写

没有回应的写

读

通知：客户端发请求服务端，不需要服务端回复一个响应 ？？

指示：客户端发请求给服务端，需要服务端回复一个响应。？？

写和没有回应的写

这两个权限允许GATT客户端写一个值到GATT服务端的一个特性中。他们之间不同的地方在与没有回应的写事件没有任何应用层上的确认或回应。

读

读性质标明一个GATT客户端可以读取在GATT服务端中特性的值。

通知和指示

通知和指示性质允许GATT服务端在其某个特性改变的时候对GATT客户端进行提醒。通知和指示之间不同之处在与指示有应用层上的确认，而通知没有。

注：Client和Server之间是通过ATT PDU来通讯的，ATT PDU主要包括4类：读、写、通知和指示。如果一个命令需要response，那么会在相应的指令后面加上request。如果一个命令只需要ACK而不需要response，那么后面就不会带request。这里要特别强调一点，BLE所有命令都是“必达”的，也就是说每个命令发出去之后，会立刻等ACK信息，如果收到了ACK包，发起方认为命令发送完成。否则发起方会一直重发该命令，直到超时导致BLE连接断开。换句话说，只要BLE没有断开，那么发送的数据包，一定会被对方收到。

那既然每个ATT命令都必达对方，那么为什么还需要request呢？如果一个命令带有request后缀，那么发起方就可以收到命令的response包，这个response包在应用层是有回调事件的，而ACK包在应用层是没有回调事件的。所以采用request/response方式，应用层可以按顺序地发送一些数据包，这个在很多应用场景中是非常有用的。相反，如果对应用层数据包的顺序没有要求，那么就可以不使用request/response形式。另外request/response有一个副作用：大大降低通信的吞吐率。因为request/response必须在不同的连接间隔中出现。也就是说，如果在间隔1中发送了一个request命令，那么response包必须在间隔2或者稍后间隔中回复，而不能再间隔1中回复。这就导致两个连接间隔最多只能发一个数据包，而不带quest后缀的ATT命令就没有这个问题，在同一个连接间隔中，可以同时发送多个数据包，这样就大大提供数据的吞吐率。

常用的带request的命令：所有读命令，写，指示等。而常用的不带request的命令有没有回应的写，通知等。

3.2 特性

一个特性至少包含2个属性：一个属性用于声明，一个属性用于存放属性的值。

所有通过GATT服务传输的数据必须映射成一系列的特性，可以把特性中的这些数据看成是一个个捆绑起来的数据。每个特性就是一个自我包容而独立的数据点。例如，如果几个数据总是一起变化，那么可以把它们集中在一个特性里。

3.3 描述符

任何在特性中的属性不是定义为属性值，就是为描述符。描述符是一个额外的属性以提供更多的特性信息。它提供一个人类可识别的特性描述的实例。

然而，有一个特别的描述符需要特别注意：客户端特性配置表述符（Client Characteristic Configuration Descriptor,CCCD），这个描述符是给任何支持通知和指示功能的特性额外增加的。在CCCD中写入“1”使能通知功能，写入“2”使能指示功能，写入”0“同时禁止通知和指示功能。

3.4 服务

一个服务包含一个或多个特性，这些特性是逻辑上相关的集合体。

GATT服务一般包含几块具有相关性的功能，比如特定传感器的读取和设置，人机接口的输入和输出。组织具有相关的特性到服务中即实用又有效。因为它使得逻辑上和用户数据上的边界变得更加清晰，同时它也有助于不同应用程序间代码的重用。GATT基于SIG蓝牙技术联盟官方设计，SIG建议根据用户自己的规范设计自己的profile。

3.5 profile（数据配置文件）

一个profile文件可以包含一个或多个服务，一个profile文件包含需要的服务信息或者是与设备如何交互的配置文件选项信息。设备的GAP和GATT的角色都可能在数据的交换过程中改变，因此，这个文件应该包含广播的种类、所使用的连接间隔、所需的安全等级等信息。

注：一个Profile中的属性表不能包含另一个属性表。

4 标准的定制服务和特性

SIG蓝牙技术联盟已经定义了一些profile、服务、特性和根据协议栈的GATT层定义的属性。但是协议栈中只实现了一部分应用的BLE服务。那就意味着，只要协议栈支持GATT，就可能为一个应用建立一个它需要的profile和服务。既然在一个应用中可以支持profile和服务，那么就可以在这个应用中建立一个个人定制的服务。

每个设备都包含以下必要的特征值和服务项：

PROFILE

Generic Access Service（Primary Service）

Device Name（Characteristic）

Apperance（Characteristic）

Generic Attribute Service（Primary Service）

Service Changed（Characteristic）

CCCD（Descriptor）

服务项这种类型本身并不包含数据，仅仅相当于是一个容器，用来容纳特征值。特征值用来保存用户数据，但是它也有自己的UUID。在处理特征值所携带的用户数据之前，需要先对特征值自身进行声明。

特征值在系统中的表达形式是：声明 + 特征值属性。比如上边Generic Access Service服务项的Device Name特征值，它在GATT数据库中表示方式是：

Characteristic 声明: 0x0002, 0x2803, access\_property, 0x2A00

Characteristic 项: 0x0003, 0x2A00, access\_property, data

其中，第一列2个字节代表属性句柄。第二列2个字节代表属性类型（UUID）。0x2803表示该项为“特征值的声明”。0x2A00表示该特征值是Device Name。在第一行特征值声明中，最后一项的属性值就是该特征值的UUID。可以看到，第一行声明里面的属性值0x2A00，而第二行属性值自己的属性类型（UUID）也是0x2A00，信息冗余了。原因是，假如特征值的UUID为自定义的16字节UUID，在特征值的属性类型（UUID）中，只能存放2个字节，而16字节的UUID全称，就只能存放在特征值声明的属性值项中。例如：

Characteristic 声明: 0x0002, 0x2803, access\_property, 0x0000ABCD1212efde1523785feabcd123

Characteristic 项: 0x0003, 0xABCD, access\_property, data

BLE的属性体系在系统中以GattDB表示，即属性数据库。在CyBle\_gatt.c文件中，可以看到cyBle\_gattDB变量，如下：

const CYBLE\_GATTS\_DB\_T cyBle\_gattDB[0x10u] = {

{ 0x0001u, 0x2800u /\* Primary service \*/, 0x00000001u /\* \*/, 0x0007u, {{0x1800u, NULL}} },

{ 0x0002u, 0x2803u /\* Characteristic \*/, 0x00000201u /\* rd \*/, 0x0003u, {{0x2A00u, NULL}} },

{ 0x0003u, 0x2A00u /\* Device Name \*/, 0x00000201u /\* rd \*/, 0x0003u, {{0x0009u, (void \*)&cyBle\_attValuesLen[0]}} },

{ 0x0004u, 0x2803u /\* Characteristic \*/, 0x00000201u /\* rd \*/, 0x0005u, {{0x2A01u, NULL}} },

{ 0x0005u, 0x2A01u /\* Appearance \*/, 0x00000201u /\* rd \*/, 0x0005u, {{0x0002u, (void \*)&cyBle\_attValuesLen[1]}} },

{ 0x0006u, 0x2803u /\* Characteristic \*/, 0x00000201u /\* rd \*/, 0x0007u, {{0x2A04u, NULL}} },

{ 0x0007u, 0x2A04u /\* Peripheral Preferred Connection Par \*/, 0x00000201u /\* rd \*/, 0x0007u, {{0x0008u, (void \*)&cyBle\_attValuesLen[2]}} },

{ 0x0008u, 0x2800u /\* Primary service \*/, 0x00000001u /\* \*/, 0x000Bu, {{0x1801u, NULL}} },

{ 0x0009u, 0x2803u /\* Characteristic \*/, 0x00002201u /\* rd,ind \*/, 0x000Bu, {{0x2A05u, NULL}} },

{ 0x000Au, 0x2A05u /\* Service Changed \*/, 0x00002201u /\* rd,ind \*/, 0x000Bu, {{0x0004u, (void \*)&cyBle\_attValuesLen[3]}} },

{ 0x000Bu, 0x2902u /\* Client Characteristic Configuration \*/, 0x00000A04u /\* rd,wr \*/, 0x000Bu, {{0x0002u, (void \*)&cyBle\_attValuesLen[4]}} },

{ 0x000Cu, 0x2800u /\* Primary service \*/, 0x00000001u /\* \*/, 0x0010u, {{0xCBBBu, NULL}} },

{ 0x000Du, 0x2803u /\* Characteristic \*/, 0x00001A01u /\* rd,wr,ntf \*/, 0x0010u, {{0xCBB1u, NULL}} },

{ 0x000Eu, 0xCBB1u /\* Custom Buffer \*/, 0x00011A04u /\* rd,wr,ntf \*/, 0x0010u, {{0x00C8u, (void \*)&cyBle\_attValuesLen[5]}} },

{ 0x000Fu, 0x2901u /\* Custom Descriptor \*/, 0x00010001u /\* \*/, 0x000Fu, {{0x001Cu, (void \*)&cyBle\_attValuesLen[6]}} },

{ 0x0010u, 0x2902u /\* Client Characteristic Configuration \*/, 0x00010A04u /\* rd,wr \*/, 0x0010u, {{0x0002u, (void \*)&cyBle\_attValuesLen[7]}} },

};

通过注释可以看到，每个特征值都有声明和特征值项两部分组成。

由于每个属性的句柄是递增的，因此属性的声明顺序会影响句柄的计算。

描述符是特征值的补充信息，挂载在特征值之下，它可以开辟一段数据空间以携带数据，客户端可以像操作特征值一样对其进行读写，但是描述符弱于特征值，它不具备Notify/Write等读写属性，远不如特征值灵活。有一种描述符叫CCCD（Client Characteristic Configuration Description），当对特征值设置Notify或Indication时，用以控制Notify和Indication的使能情况。

# 蓝牙学习八（配对与绑定）

————————————————

版权声明：本文为CSDN博主「t\_guest」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/qq\_26226375/article/details/128494718

————————————————

1.简介

Paring（配对）和Bonding（绑定）是实现蓝牙射频通信安全的一种机制，有两点需要注意：

Paring/bonding实现的是蓝牙链路层的安全，对应用层来说是完全透明的。也就是说，不管有没有paring/bonding，应用层数据的收发方式都是一样的，不会因为加了paring/bonding而使应用层数据传输需要做某些特殊处理。

安全有两种选项：加密或签名。目前绝大部分应用都是选择加密。

实现蓝牙通信安全，除了paring/bonding这种底层方式，用户也可以自己在应用层去实现相同的安全功能。从功能上来说是没有太大区别的，不同的是应用层安全自己实现的话，需要自己选择加密算法、密钥生成、密钥交换等。如果不是这方面的专家，很有可能会有安全漏洞。而蓝牙的paring/bonding则把上述过程标准化，放在了蓝牙协议栈里。并且其安全性得到了充分的评估，用户可以“无感”使用安全的蓝牙通信。

paring/bonding是蓝牙sercurity manager（SM）的一部分，SM定义了蓝牙通信的安全架构，里面涉及安全架构、密码工具箱、paring协议等。其中paring协议是关键，所以经常把paring和SM二者等价。

2.配对

配对是找到并确定需要和自己通信的设备，也就是身份确定。接着是安全密钥共享，而这一过程仅仅是由启动加密到得到短期密钥（STK）为止。其包括配对能力交换、设备认证、密钥（固定128bit）生成、连接加密以及机密信息分布等过程。配对的目的有三个：加密连接、认证设备、生成密钥。

区别于传统蓝牙的配对过程，**BLE的配对过程发生在连接过程之后。**

和经典蓝牙一样，协议为处于连接状态的BLE设备，定义了两种Link Layer角色：Master和Slave。Master是连接的发起方，可以决定和连接有关的参数。Slave是连接的接收方，可以请求连接参数，但无法决定。

在SM（Security Manager）的规范中，配对时指"Master和Slave通过协商确立用于加（解）密的Key的过程"，主要由三个阶段组成：

配对是一个三阶段的过程。前两个阶段时必须的，第三阶段时可选的。

第一阶段：配对特征交换。称作“Pairing Feature Exchange”，用于交换双方有关鉴权的需求（authentication requirements），以及双方具有怎样的人机交互能力（IO Capabilities）。

第二阶段：短期密钥（STK）生成。通过SMP协议进行实际的配对操作，根据阶段一“Feature Exchange”的结果，有两种配对方法可选：LE legacy pairing和LE Secure Connections

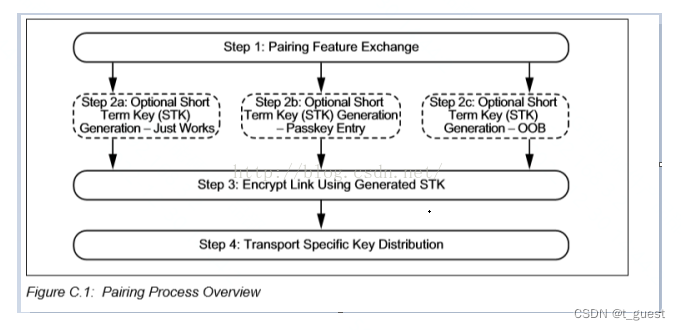
第三阶段：传输特定密钥分配。该阶段时可选的，经过阶段一和阶段二之后，双反已经产生了加密key。因此可以建立加密的连接。加密连接建立后，可以互传一些私密的信息，例如Encryption Information、Identity Information、Identity Address Information等。

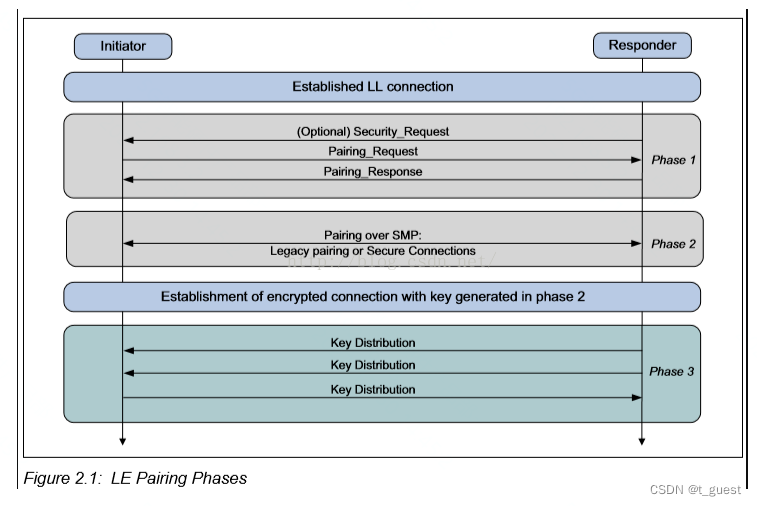
注：STK的生成规则如下：

Just work：没有加密，TK=0X00。Just works方式不能抵抗窃听者和中间人攻击，只有在配对过程时没有遭受攻击，后面加密的链路数据才是可信的。安全级别很低。

passkey entry：密码输入。如果passkey是“019655” ，TK的值就是0x00000000000000000000000000004CC7。（0x4cc7就是19655的16进制数）。将输入的值作为一个6位数的十进制，转换为15字节的十六进制。

OOB（Out of band）：带外的TK值，是一个16字节的随机数，通过非BLE的方式传递给对端。例如二维码或IR红外。对于蓝牙窃听者/攻击者而言，这个data的传输是不可见的，因此会显得安全很多。





2.1 配对请求的数据格式



IO capabilities表明输入，输出的能力。

输入是按键、键盘，输出是显示数字用的界面。

0x00（Display Only）只能 显示000000~999999的数字

0x01（Display YesNo）显示Yes/No的按钮

0x02（Keyboard Only）只能输入000000~999999的数字

0x03（No input No output）没有输入也没有显示，只能用Just work工作方式

0x04（Keyboard Display）能输入000000~999999的数字，也能有屏幕显示。

OOB data flag

0x00 OOB数据没有发送

0x01 OOB数据通过远端设备发送（如IR）

0x02-0xFF 保留

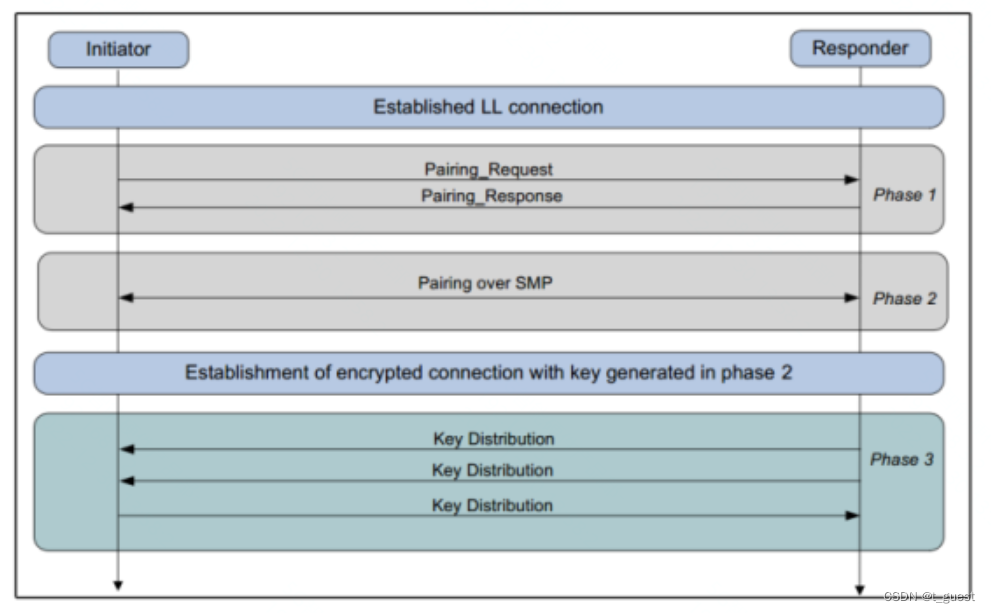
3.绑定

配对过程中会生成一个长期密钥（LTK,long-term key），如果配对双方把这个LTK存储起来放在Flash中（有的时候是长期密钥、身份解析密钥、连接签名解析密钥这三个秘密要的某一个或组合进行交换，然后将交换的密钥存储到数据库中），那么这两个设备再次重连的时候，就可以跳过配对流程，而直接使用LTK对蓝牙连接进行加密，设备的这种状态称为绑定（bonding）。

这里要明确一个概念，配对绑定只有在两个设备之间第一次配对时才会发生，后续的连接由于第一次的配对绑定已经有“Bonding”过程（即存储），如果存储的数据库没有被认为的清空，后续的连接不需要再次配对。并不是所有的通信都需要加密进行数据保护，因此，建立连接之前不一定需要配对和绑定，可以直接建立连接。

4.总结

BLE的配对和绑定是一连串的动作，总结下来可以用下图来表示。



阶段1：配对特征交换，得到临时密钥（TK）值（配对请求、配对响应）

阶段2：身份确认以及短期密钥（STK）生成（通过安全管理协议（SMP）配对），确定自己正在和一个真正想要通信的设备通信，而非第三方。即确定对方身份。

阶段3：传输特定密钥（密钥分配）。绑定所需存储到安全数据库的数据也是在此阶段发送的。

上述三阶段总结：

配对认证：主从机一方提供密码，一方输入密码，如果双方密码一致，那么此密码将作为TK（临时密码）。

加密链路：利用得到的TK（临时密码）等信息计算出STK（短期密码）用来做加密认证

绑定：加密认证通过后，利用STK等信息生成LTK（长期密码），把LTK保存下来，用于下次连接时做加密认证，不需要再次配对就可以加密链路。这就是绑定。

绑定后通讯过程：每次连接时，从机会向主机发送安全请求。如果主从机相互绑定过，主机不会发送配对请求，主机直接利用绑定时保存的LTK发送加密请求。从机也会利用绑定时保存的LTK来做加密回复，三次握手成功后（加密成功，三次握手通讯由底层完成，用户不可见），从机回复主机加密状态success。

建立连接是使用的静态密码，解除绑定/配对是使用的动态密码。回连也需要密码，使用的是动态密码，这是为了防止窃听，起保护作用。为了防止泄露信息，用户也可以自行在应用层去实现相同的功能。

配对和绑定的区别：

1.连接：通讯的基础，通讯数据为明文

2.配对：配对仅仅是为了在连接的基础上加密（通讯数据经过加密为密文）。提高蓝牙链路传输的安全性。不配对也能连接进行通信。

3.绑定：绑定是配对发起时的一个可选配置。把配对信息记录下来，下次不用配对自动进入加密的连接。所以没有在bonding列表里的设备不影响连接。

在不引起误解的情况下，经常把paring当成paring和bonding两者的组合，因为只paring不bonding的应用情况非常少见。

# 关于蓝牙协议中GATT、特性、属性解释

————————————————

版权声明：本文为CSDN博主「t\_guest」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

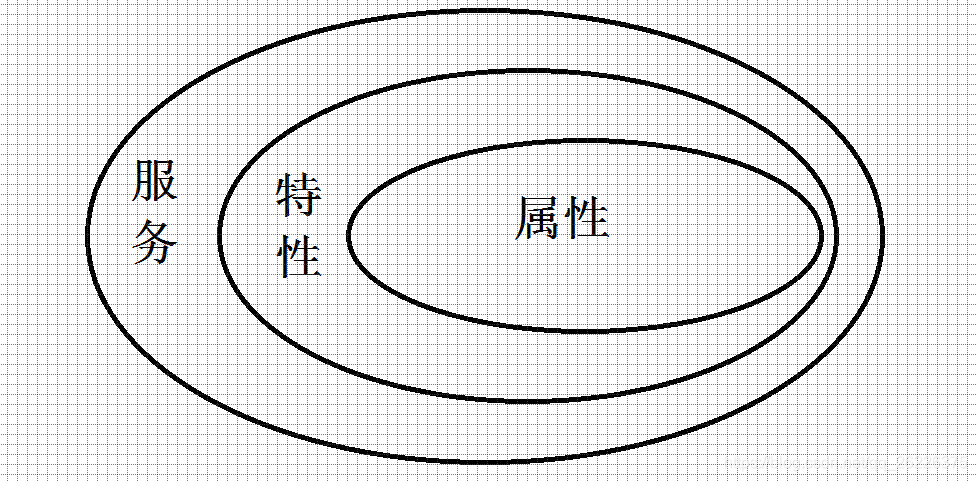
原文链接：<https://blog.csdn.net/qq_26226375/article/details/114370896>

————————————————

在蓝牙的学习中，一上来就要面对一大堆的专业名词。首先就是服务是什么？特性是什么？属性是什么？刚开始的时候查了很多资料都还是没有搞懂。这次就结合代码，尽量用最通俗易懂的方式将这些东西解释并记录下来，也希望能帮助到别人。（如果有不对的，也希望有大神能指出来）

## 通用属性配置文件（Generic Attribute Profile）GATT

GATT主要定义了。服务器server和客户端client。而其他的服务、特性、属性都是在这个协议下的各种角色。他们的关系如下：



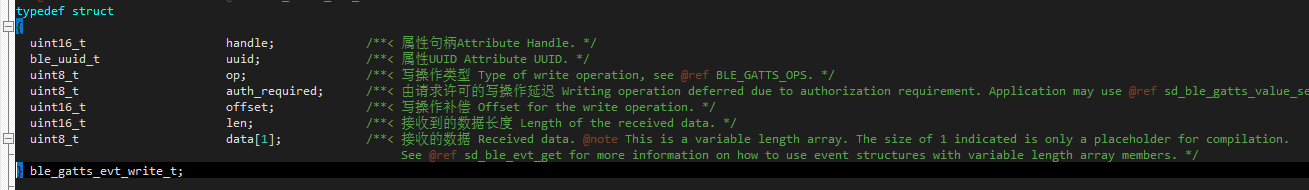
## 属性（Attribute）

属性由三部分组成，属性句柄、属性类型（UUID）、属性值。如下图：

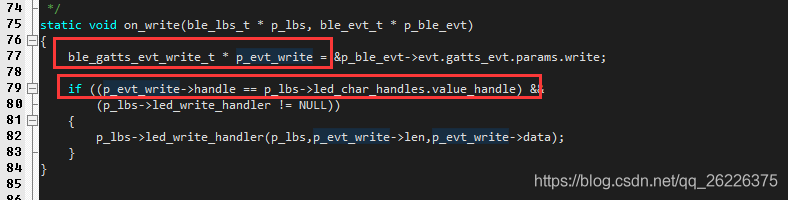
属性句柄(2字节) 属性类型(UUID)(2或6字节) 属性值(0-512字节)

属性句柄是属性在GATT表中的索引。一台设备可以有许多的属性，但是可能会有许多相同功能的属性，比如有许多测温属性，为了识别不同的属性，需要给每个属性加一个索引值。可以理解为内存地址。每一个属性的句柄都是唯一的。

这里已写属性为例，



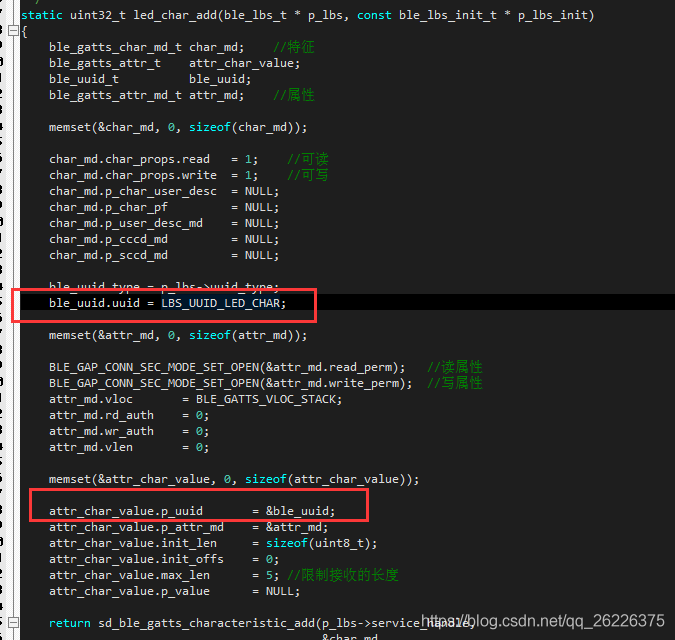
这里可以看到这个事件属性中第一个字节就是属性句柄。那再看是如何调用的。



这个以点亮LED的例程为例，在进行属性指针申请后，将指针指向了底层上报上来的事件属性，然后进行指针的一个寻址。这里代码可以不用看懂，只要知道属性句柄是这个用法即可。一般在开发时，这个值通常是协议栈自动生成的。 比如 你调用一个创建特性的API时，是不需要指定属性句柄这个值的，这个值协议栈内部会自动在每次创建一个属性时赋予一个值。通常就是简单的随着你添加的特征值其属性句柄递增赋值。

那UUID是干嘛的呢？所有的属性都必须要有一个UUID值，UUID是全球唯一的128位的数据，用来识别不同的特性。蓝牙核心规范制定了两种不同的UUID，一种是基本的UUID，一种是代替基本UUID的16为UUID。我们协议栈的后面那里使用的就是代替基本UUID的16为UUID：先增加一个特定的基本UUID，在定义一个十六位的UUID。加载到基本UUID之上。

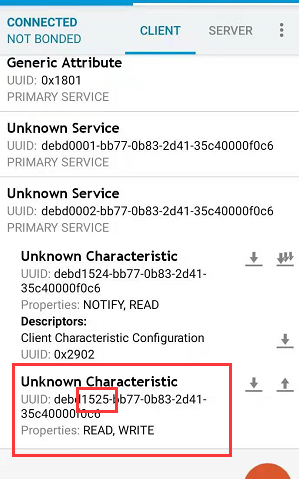
其实这个UUID主要是用来区别一个服务下的不同特性的。



还是以LED的例子。这个函数是添加特性的函数中，

IMG_256

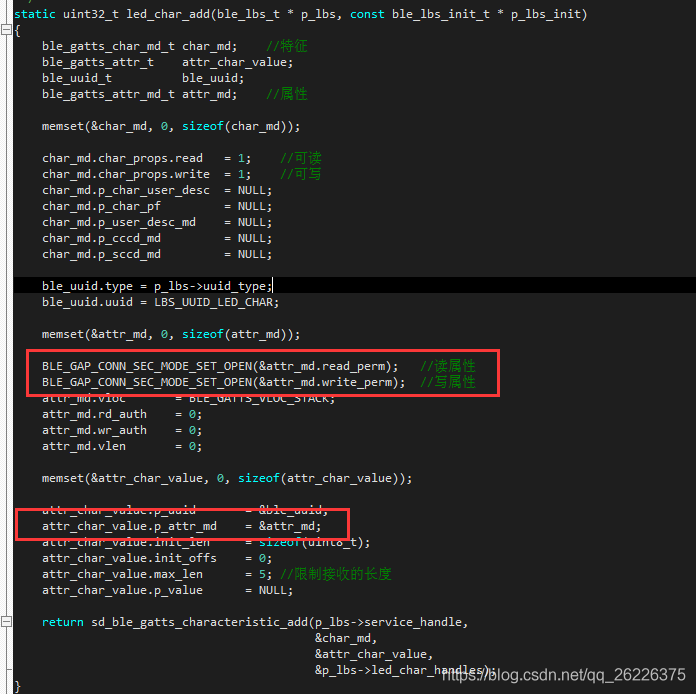
这里可以看到LED的这个UUID是1525.那在手机上连接之后看一下。



看这个实际上是确定了这个特性的UUID。

## 属性权限

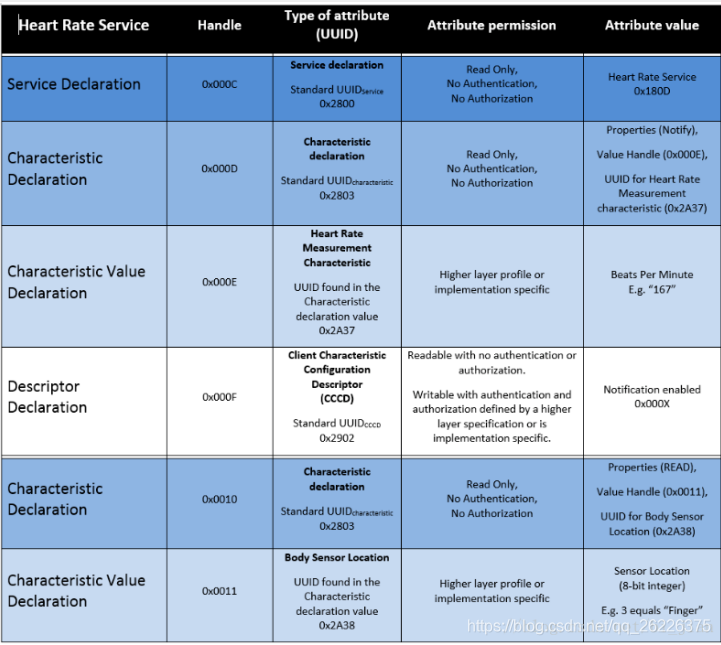
  属性权限定义了与属性交互的规则。它定义了该条属性是否可读可写以及进行读写操作时需要进行哪种授权。这里需要强调的是属性权限只能用来控制对属性值的操作，它不用于属性句柄，属性类型以及属性权限本身。客户端可以通过遍历服务器上的属性表，就可以获悉服务器可以提供的服务，不需要读写属性值。



依旧以LED例子为例，这里可以看到配置了属性值(attr\_char\_value)。而读写属性直接赋值给了属性值。

## 属性值

  属性值可以是任何东西。它可以是每分钟心率数值，可以是灯的开关状态，或者是像“hello world”之类的字符串。有时候它还可以包含某些信息，通过这些信息可以找到其它属性及其性质。



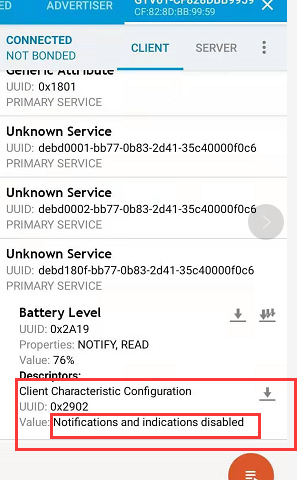
例如在上表中的服务声明属性（属性句柄0x000C）中，属性值是0x180D，这是蓝牙技术联盟定义的服务UUID，用以标识这是哪种服务。特性声明属性（属性句柄0x000D）的属性值包含了下一条属性即特性值声明属性的信息，包括属性句柄，属性类型，属性权限。最后，心率测量特性值属性（属性句柄0x000E）的属性值中包含了每分钟的心率数值。属性值的具体含义由属性类型决定。

## 描述符

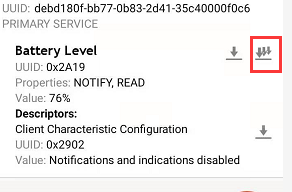
任何在特性中的属性不是定义为属性值就是为描述符。描述符是一个额外的属性以提供更多特性的信息，它提供一个人类可识别的特性描述的实例。

有一个特别的描述符值得特别的提起，客户端特性描述符(Client Charactristic Configuration Descriptor，CCCD)，这个描述符是给任何支持通知或指示功能的图形额外增加的。该描述符是一个2位的数值，分别用于设置通知与指示，但是不允许同时设置。在CCCD中写入“1”使能通知功能。写入“2”使能指示功能。写入“0”同时禁止通知和指示功能。

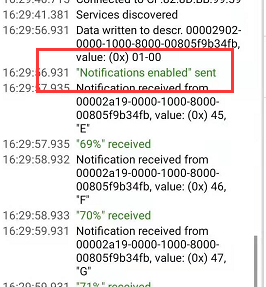
实际上这个CCCD在创建好之后默认都是关闭的，我们通过nRF Connect软件能更直观的看到。以电量采集为例。

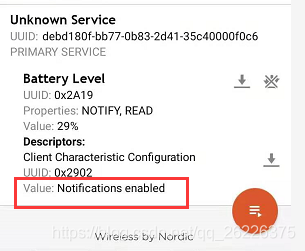


这个是连接上后的状态。当点击“控件”开启notifi时

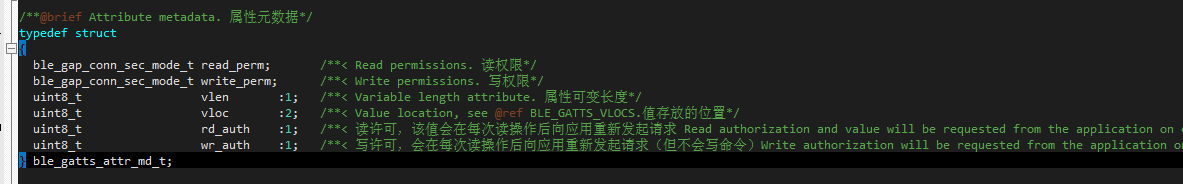


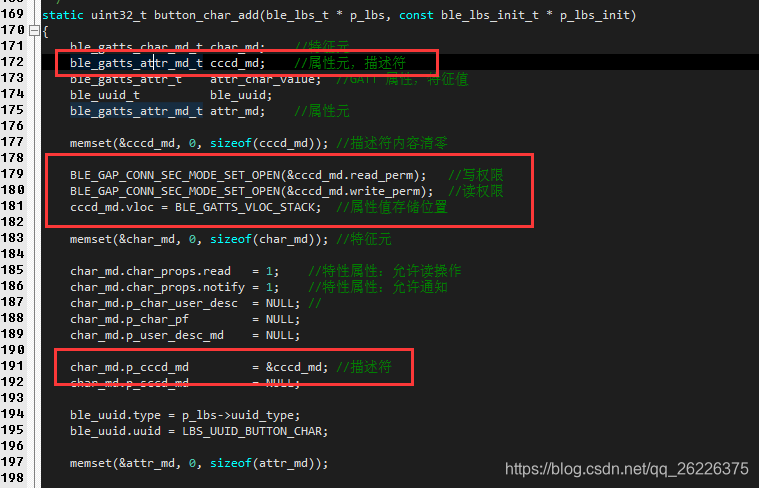
看软件的LOG，显示如下



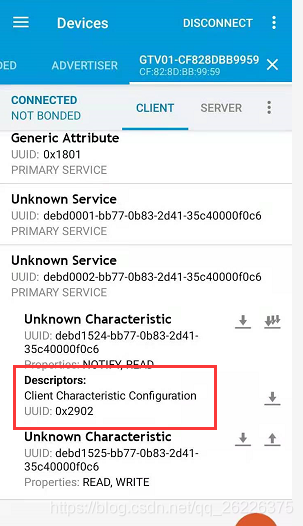


可以看到notifi功能已经开启。这时候Battery Level 的Value部分就会自动变化。

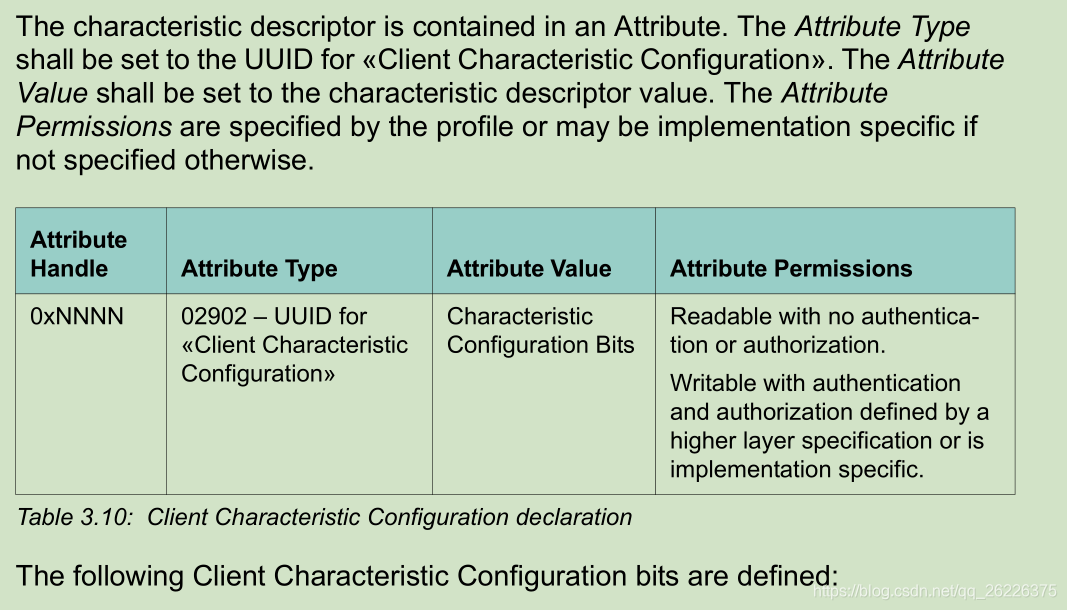




属性符被配置到特性中。这里配置CCCD属性可读可写。



这里有个问题，为什么UUID是2902，在程序里也没有设置呀。查看Bluetooth core文档后发现。



这个是SIG已经定义好的，所有的CCCD的UUID都是2902.。

## 特性(Characteristics)

一个特性至少包含2个属性：一个属性用于声明，一个属性用于存放特性的值。所有通过GATT服务传输的数据必须映射成一系列的特性，可以把特性中的这些数据看成是一个个捆绑起来的数据，每个特性就是一个自我包容而独立的数据点。例如，如果几块数据总是一起变化，那么我们可以把它们集中在一个特性里。

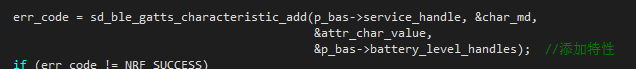
而属性其实也可以称为属性条目，一个特性值就是一个属性条目。但是特性值 属性条目不是单独存在的， 它是属于特性 的一部分。特性由几条属性条目组成， 包括一条 特性声明属性条目， 一条特性值属性条目， 几条描述符属性条目。

如下所示例的一个特性组成，包括一个 声明，一个特性值，两个描述符。



描述符可以有很多，其中特殊的就是CCCD，客户端描述符。在上边有说过。那如果没有notifi的功能，那一个特性也就只有声明条目和属性值条目。

Nordic sdk中提供的API就是创建特性api， 根据我们填入的参数协议栈会自动创建 声明，特性值，以及描述符， 并且API会返回 特性值以及一些存在的描述符的句柄，如下图。



API的第一个参数 service\_handle为 服务句柄，即添加的特性是添加到哪个服务中的，所以首先要创建在服务后才能再服务下面添加特性，sdk中调用创建服务API函数后，会返回被创建的服务的句柄，如下图

IMG_256

之后第二个参数p\_char\_md定义了 特性中的 特性值 支持哪些操作，读，些,notify等。以及所需要的描述符的读写权限等信息。

第三个参数p\_attr\_char\_value 定义了 特性值 的UUID，长度等信息。

第四个参数p\_handles 为协议栈返回的 创建的特性 中的一些句柄，包括特性值句柄，以及存在的描述符句柄。

这里有个问题，为什么每条属性已经有属性权限来控制属性值的读写了，还需要其他的属性性质来控制特性值的读写呢？它们不应该是一样的吗？这是一个值得一提的问题。事实上，控制特性值的属性性质只用在GATT和应用层，仅仅为客户端提供一些线索，客户端可以据此知道从特性值声明属性中能够获取哪些内容。控制属性的属性权限（ATT层）凌驾于特性值性质（GATT层）之上，最终决定对属性值的访问。你可能会问，为什么需要属性权限和性质两个部分呢？答案很简单，因为蓝牙核心规范是这样规定的。可能这个答案很令人困惑和失望，但是蓝牙核心规范决定我们如何设置这些特性，所以这里要提一下。