[Bluetooth ATT介绍](http://blog.csdn.net/hzl6255/article/details/41930455)

ATT，即Attribute Protocol，用于发现、读、写对端设备的协议(针对BLE设备)

**1 介绍**

ATT允许设备作为服务端提供拥有关联值的属性集   
让作为客户端的设备来发现、读、写这些属性；同时服务端能主动通知客户端

ATT定义了两种角色: 服务端(Server)和客户端(Client)

ATT中的属性包含下面三个内容   
- Attribute Type       : 由UUID(Universally Unique IDentifier)来定义   
- Attribute Handle     : 用来访问Attribute Value   
- A set of Permissions : 控制是否该Attribute可读、可写、属性值是否通过加密链路发送

一个设备可以同时拥有Server和Client   
而一个Server可以支持多个Client

**2 详细内容**

Client通过使用ATT PDUs(Protocol Data Unit)来发现Attribute Handles

**2.1 Attribute Type**

Attribute Type由UUID唯一指定，UUID是一个128-bit值

在使用过程中，为了提高效率，使用的是16-bits Attribute UUID   
128-bit UUID        = 16-bit Attribute UUID\*2^96 + Bluetooth\_Base\_UUID   
Bluetooth\_Base\_UUID = 00000000-0000-1000-8000-00805F9B34FB

更简单的办法如下(xxxx代表十六进制的16-bit UUID)   
0000xxxx-0000-1000-8000-00805F9B34FB

*TIP: 16-bit Attribute UUIDs与SDP 16-bit UUIDs使用相同的命名空间*

**2.2 Attribute Handle**

Attribute Handle是由Server分配的一个唯一且非零16-bit值   
0x0000 : 保留   
0xFFFF : 最大Attribute Handle

**2.3 Attribute Handle Grouping**

Grouping是一由高层协议定义的一组属性，他们位于其他属性组之前   
客户可以请求第一个和最后一个与属性组关联的Handle

**2.4 Attribute Value**

Attribute Value是一个八位组(长度为8或可变)   
当属性值太长时，可通过多个PDUs发送

When transmitting attribute values in a request, a response, a notification or an indication,   
the attribute value length is not sent in any field of the PDU.

The length of a variable length field in the PDU is implicitly given by the length of the packet that carries this PDU.   
This implies that:   
a) only one attribute value can be placed in a single request, response, notification or indication    
   unless the attribute values have lengths known by both the server and client, as defined by the attribute type.   
b) This attribute value will always be the only variable length field of a request, response, notification or indication.   
c) The bearer protocol (e.g. L2CAP) preserves datagram boundaries.

**2.5 Attribute Permissions**

属性都有一组与之相关联的Permission Values

一个给定属性的权限由高层协议定义，并对ATT不可见

当访问一个安全属性需要一个认证的链路，而Client没有足够的权限   
Server则响应一个Error Code(Insufficient Authentication)   
Client收到Error Code后应该尝试认证该链路，成功后即可访问该安全属性

当访问一个安全属性需要一个加密的链路，而链路没有加密   
Server则响应一个Error Response(Insufficient Encryption)   
Client收到Error Response后应该尝试加密该链路，成功后即可访问该安全属性

当访问一个安全属性需要一个加密的链路   
链路有加密，但是对于所需要的安全级别来说加密Key Size太短   
Server则响应一个Error Code(Insufficient Encryption Key Size)   
Client收到Error Response后应该尝试使用更长的Key Size加密该链路，成功后即可访问该安全属性

Attribute Permissions由下列三个权限组合而成   
- Access Permissions         : 决定Client是否可读、写属性值   
    ~ Readable   
    ~ Writable   
    ~ Readable and Writable   
- Authentication Permissions : 决定是否需要一个认证的物理链路(Authenticated Physical Link)   
    ~ Authentication Required   
    ~ No Authentication Required   
- Authorization Permssions   : 决定Client在访问属性值前是否需要授权   
    ~ Authorization Required   
    ~ No Authorization Required

**2.6 Control-Point Attribute**

不可读、可写、可通知(Notified)和可指示(Indicated)的属性被称为Control-Point Attribute

高层协议可使用该属性来使能设备特定过程，比如设备上一个给定过程的命令或指示已经完成

**2.7 Protocol Methods**

ATT使用Protocol Methods来发现、读、写、通知、指示属性   
方法可分为如下几种   
- Request   
- Response   
- Command   
- Notification   
- Indication   
- Confirmation

一些ATT PDUs包含一个认证签名(Authentication Signature)   
允许PDU发送端不需要加密

这些方法和符号位被称为opcode

**2.8 Exchanging MTU Size**

ATT\_MTU定义了Client和Server之间数据包的最大值   
其默认值由高层协议来定义

Client和Server可通过Exchange MTU Request and Response PDUs来交换最**[大数据](http://lib.csdn.net/base/spark" \o "Apache Spark知识库" \t "_blank)**包   
然后均使用交换值中的最小值进行通信

同时作为Server和Client的设备应该使用相同的Client Rx MTU和Server Rx MTU

每个ATT Bearer均有其ATT\_MTU   
当一个设备拥有多个ATT Bearer时，不同ATT Bearer的ATT\_MTU可能不同

**2.9 Long Attribute Value**

在单一数据包中可发送最长的属性大小为[ATT\_MTU –1] octets   
在一个Attribute PDU中至少包含Attribute Opcode

属性值比[ATT\_MTU-1] octets大时属性称为Long Attribute

对于Long Attribute   
- 使用Read Blob Request来读取整个属性(Attribute > [ATT\_MTU-1] octets)   
- 使用Prepare Write Request和Execute Write Request来写整个属性(Attribute > [ATT\_MTU-3] octets)

ATT无法决定属性值是否大于[ATT\_MTU] octets   
但是高层协议Can Tell

最大的属性值为512 octets

**2.10 Atomic Operation**

Server应该将Client的每个请求或命令视为不受影像的原子操作   
如果一个链路由于某种原因断开，高层协议应当对属性值得修改负责

Long Attribute不能被单一的原子操作读、写

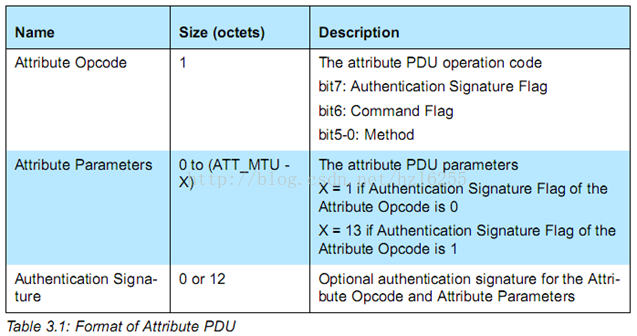
**3 Attribute PDU**

Attribute PDUs有六种类型   
- Requests      : Client->Server, 请求回应

- Responses     : Server->Client, 响应请求.   
- Commands      : Client->Server, 命令   
- Notifications : Server->Client, 服务端通知   
- Indications   : Server->Client, 请求确认   
- Confirmations : Client->Server, Ind确认

Server至少能处理以下请求   
- Find Information Request   
- Read Reques

Attribute PDU格式如下



===Authentication Signature Flag===   
- 1: PDU包含Authentication Signature(12 octets), X为13   
- 0: PDU不包含Authentication Signature, X为1   
当Attribute PDU包含Authentication Signature时，则该PDU不应该通过加密链路传输

===Command Flag===   
- 1: PDU为一个Command

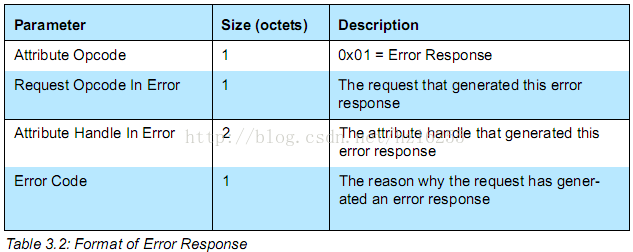
注意: 只有Write Command可能包含一个Authentication Signature

ATT是一种Sequential Protocol，这意味着在执行下一个动作前应该得到相应的回应   
ATT将Request-Response和Indication-Confirmation Pair看出一个单一的事务(Transaction)

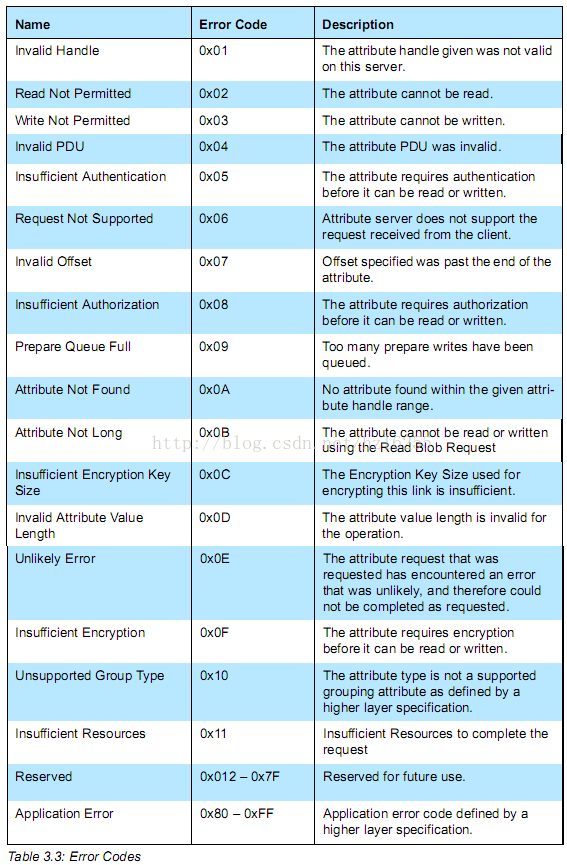
**4 Attribute Protocol PDU**

**4.1 Error Handing**

Error Response用来声明一个给定的请求无法完成，并给出原因



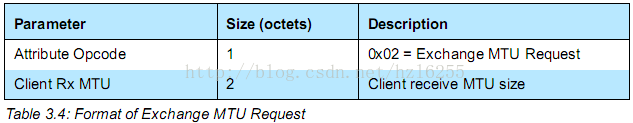
Request Opcode In Error   : 发生错误的Attribute Opcode   
Attribute Handle In Error : Attribute Opcode对应的Attribute Handle, 若无或Opcode不支持, 则使用0x0000   
Error Code                : 为下图值之一, 若Client无法理解(如ErrorCode为更新版本中定义), 则认为是未知原因



**4.2 MTU Exchange**

**4.2.1 Exchange MTU Request**

Client使用MTU Exchange Request来告知所支持的最大接收MTU size   
同时请求Server回应Server所支持的最大接收MTU size

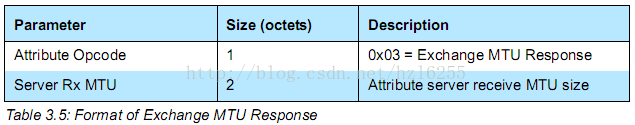


Client Rx MTU>= default ATT\_MTU

该请求在一个连接中仅发送一次，Client Rx MTU应当设置为Client所能接收ATT PDU的最大值

**4.2.2 Exchange MTU Response**

Server使用Exchange MTU Response来回应来自Client的Exchange MTU Request



Server Rx MTU>= default ATT\_MTU

Server Rx MTU应当设置为Server所能接收ATT PDU的最大值

当完成Req-Rsp后，Server和Client将ATT\_MTU均设置为Client Rx MTU和Server Rx MTU中的较小值

ATT\_MTU生效时机:   
- Server : Response发送后、其他ATT PDU发送前   
- Client : Response接收后、其他ATT PDU发送前

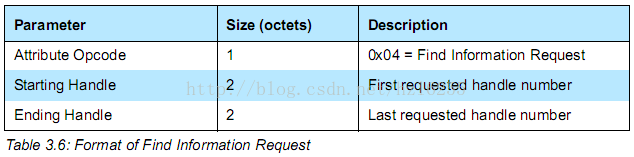
当其中任一值小于默认ATT\_MTU大小时认为该值不正确，此时应当将ATT\_MTU设置为默认值

当一个设备同时作为Client和Server，不同角色时ATT\_MTU值应当相同(详细规则可参看规范)

**4.3 Find Information**

**4.3.1 Find Information Request**

Find Information Request被用来获取与Attribute Handles相关联的类型   
Client使用该请求来发现Server上的Attribute-Type列表

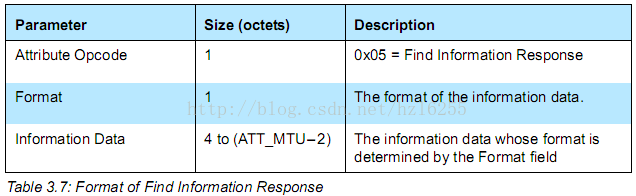


读取所有Attributes   
- Starting Handle : 0x0001   
- Ending Handle   : 0xFFFF

规则   
- Starting Handle <= Ending Handle : Response PDU或Error Response<Attribute Not Found>   
- Starting Handle  > Ending Handle : Server回应Error Response<Invalid Handle>   
- Starting Handle  = 0x0000        : Server回应Error Response<Invalid Handle>

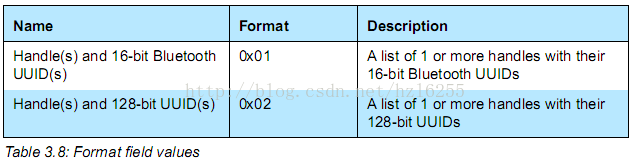
Server不应该回应以下Error Code的Error Response   
- <Insufficient Authentication>   
- <Insufficient Authorization>   
- <Insufficient Encryption Key Size>   
- <Application Error>

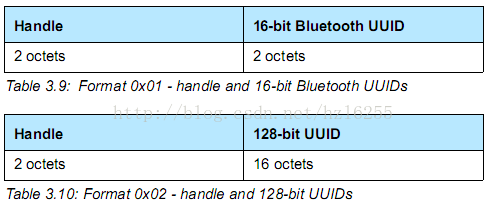
**4.3.2 Find Information Response**



Response PDU中应当包含完整的Handle-UUID对   
这意味着Handle-UUID对应该在单一的回应报文中   
同时Response PDU按照Handle的升序回应

Format参数有两个可能值

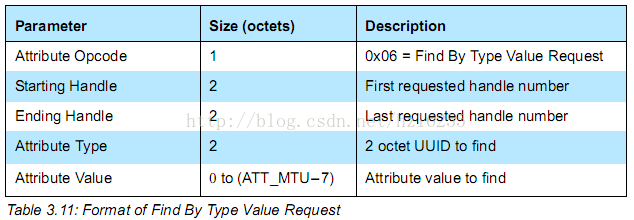




当一个Response PDU无法装下所有的Handle-UUID对时   
Client以新的Starting Handle发起另一个Find Information Request来获取未回应的信息

**4.3.3 Find By Type Value Request**

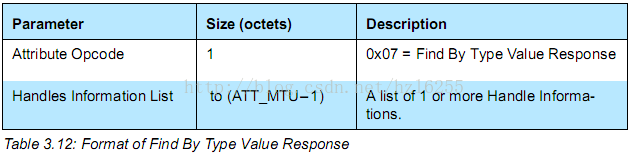
该Req用来获取指定16-bit UUID Attribute Type和Attribute Value的Attribute Handles



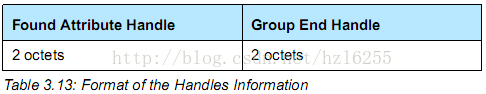
读取所有Attributes   
- Starting Handle : 0x0001   
- Ending Handle   : 0xFFFF

其他规则与Find Information Request相似

**4.3.4 Find by Type Value Response**



Handles Information List包含一个或多个Handle Information列表

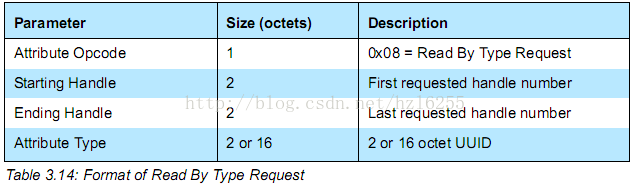


规则与Find Information Response相似

**4.4 Reading Attributes**

**4.4.1 Read By Type Request**

Read By Type Request用来获取Attribute Values(Client已知Attribute Type但不知Attribute Handle)



查找所有Attributes   
- Starting Handle : 0x0001   
- Ending Handle   : 0xFFFF

TIP : 所有的Attribute Type通过128-bit UUID进行比较，即使提供的是16-bit UUID

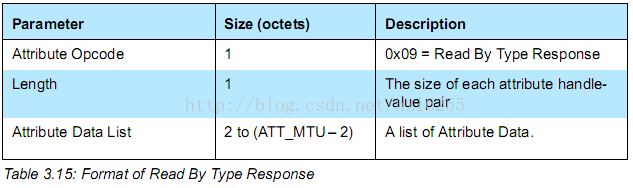
规则   
- Starting Handle <= Ending Handle : Response PDU或Error Response<Attribute Not Found>   
- Starting Handle  > Ending Handle : Server回应Error Response<Invalid Handle>   
- Starting Handle  = 0x0000        : Server回应Error Response<Invalid Handle>

该报文中Attribute Type的长度应当相同，否则需要重新发起另一次请求

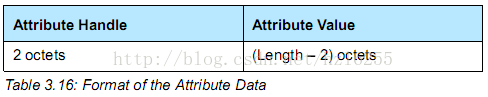
当Client因为安全原因无法访问时，Server应该回应Error Response   
- <Insufficient Authentication>   
- <Insufficient Authorization>   
- <Insufficient Encryption Key Size>   
- <Insufficient Encryption>

一些其他权限原因导致无法读时，则回应   
- <Read Not Permitted>

**4.4.2 Read By Type Response**



Attribute Data Field字段是Attribute Handle-Value列表



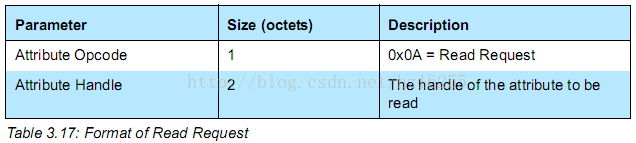
Read By Type Response包含完整的Handle-Value对   
这意味着Handle-Value对应该在单一的回应报文中   
同时Response PDU按照Handle的升序回应

Handle-Value对的最大长度为255 octets，即Length参数   
故最大的Attribute Value为(Length-2)=253 octets

在回应的Handle-Value列表中   
- Attribute Value  > (ATT\_MTU-4)/253 octets部分 : Client重新请求Read Blob Request来获取   
- Attribute Value <= (ATT\_MTU-4)/253 octets部分 : Rsp PDU中回应

**4.4.3 Read Request**

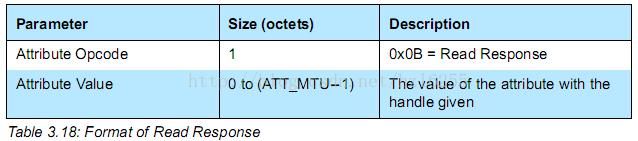
Read Request用来请求Attribute Value



Attribute Handle应该为一个有效的Handle   
否则回应Error Response<Invalid Handle>

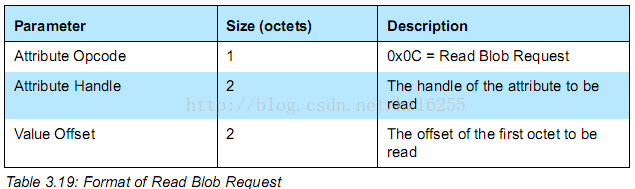
因安全问题无法访问的回应同Read By Type Request

**4.4.4 Read Response**



当Attribute Value长度超过(ATT\_MTU-1)时，前(ATT\_MTU-1) octets应该被回应   
Client发起Read Blob Request来获取剩下数据

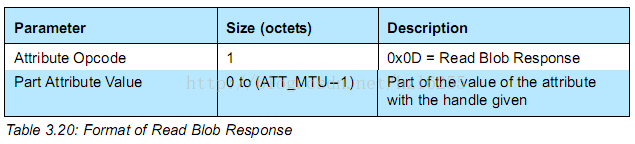
**4.4.5 Read Blob Request**



用来获取Handle指定Offset的Attribute Value

*NOTICE: 内容太多，笔者无法一一细述，仅给出概要，详细信息请参考规范，下同*

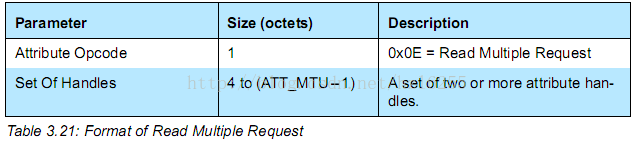
**4.4.6 Read Blob Response**



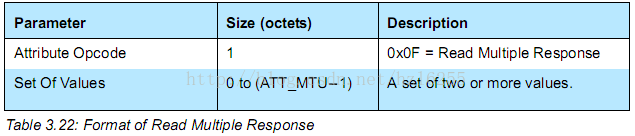
当Value Offset等于Attribute Value的长度时，回应0   
当Attribute Value大于(Value Offset + ATT\_MTU-1)时   
回应Attribute Value从Value Offset开始的(ATT\_MTU-1)个octets

**4.4.7 Read Multiple Request**

用来请求两个或更多的属性集的值

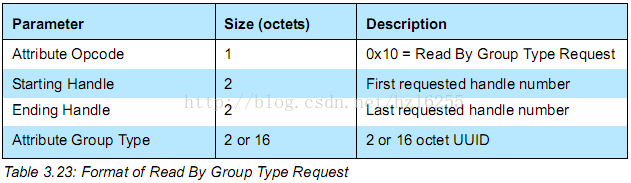


**4.4.8 Read Multiple Response**

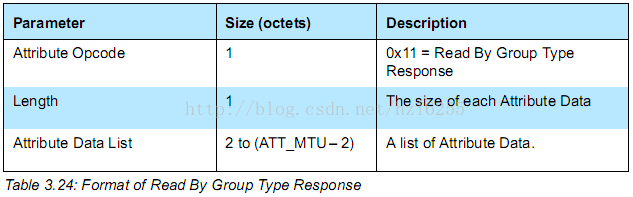


**4.4.9  Read by Group Type Request**

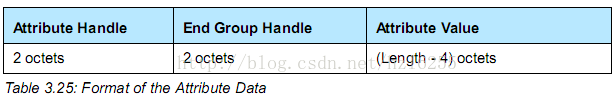
用来请求已知Attribute Type的属性值



**4.4.10  Read by Group Type Response**



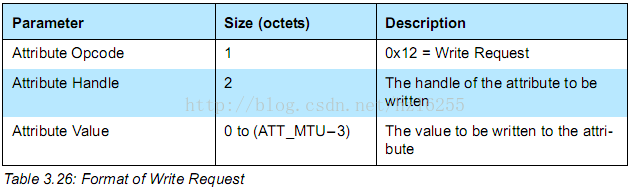
Attribute Data格式如下



**4.5 Writing Attributes**

**4.5.1 Write Request**

用来写属性值



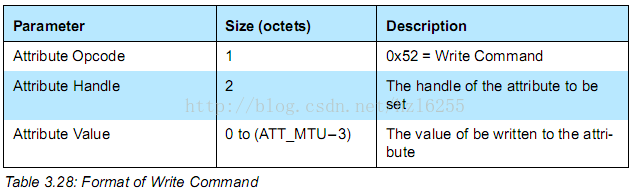
**4.5.2 Write Response**

用来通知Client属性值已成功写入



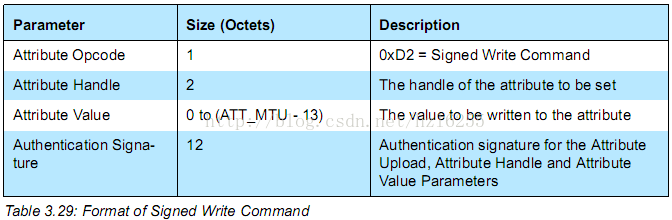
**4.5.3 Write Command**

用来写属性值，通常为Control-Point Attribute



**4.5.4  Signed Write Command**

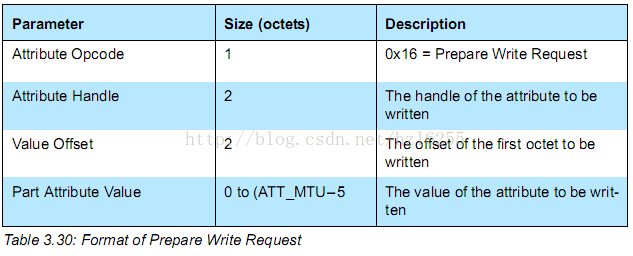
用来写属性值，包含一个Authentication Signature，通常为Control-Point Attribute



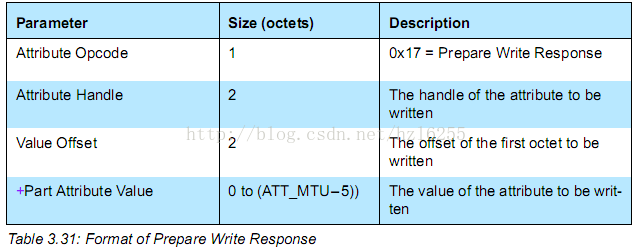
**4.6 Queued Writes**

将多个属性值写操作以FIFO方式入队，然后再一个原子操作中进行

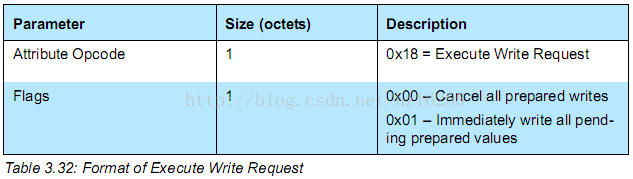
**4.6.1  Prepare Write Request**



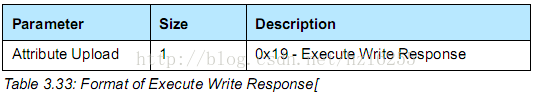
**4.6.2 Prepare Write Response**



**4.6.3  Execute Write Request**



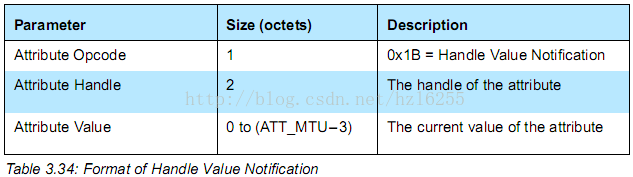
**4.6.4 Execute Write Response**



**4.7 Server Initiated**

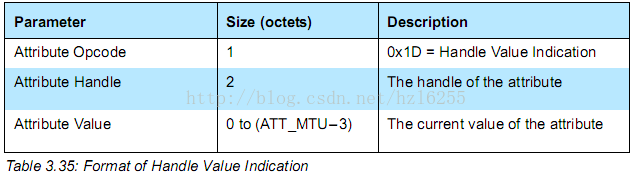
**4.7.1  Handle Value Notification**

Server可以在任何时候向Client发送Attribute Value的通知



**4.7.2  Handle Value Indication**

Server可以发送Attribute Value的指示(Indication)



**4.7.3  Handle Value Confirmation**

用来回应Handle Value Indication，确认Ind已接收

