ANCS学习、经验及其缺陷注意

---RKB Global Co. Ltd.

**第一章 简介**

ANCS是由Apple公司推出的实现消息推送的非官方（SIG）GATT-Based Services。它的实现依赖于3条Characteristics：Notification Source Characteristic (NS), DS (Data Source Characteristic), Control Point Characteristic (CP)。

Service UUID = 7905F431-B5CE-4E99-A40F-4B1E122D00D0

Notification Source: UUID 9FBF120D-6301-42D9-8C58-25E699A21DBD (Notify)

Control Point: UUID 69D1D8F3-45E1-49A8-9821-9BBDFDAAD9D9 (Reliable Write)

Data Source: UUID 22EAC6E9-24D6-4BB5-BE44-B36ACE7C7BFB (Notify)

上面的16字节UUID、在空气中按照Little Endian传送，比如说Service UUID，“D0”最先传送，“79”最后传送。

ANCS在角色上分为NP（Notification Provider）和NC(Notification Consumer)，ANCS这个Service存在于NP上，因此从BLE Spec角度来说，NP属于GATT Server，而NC则属于GATT Client。

(NP)和(NC)建立连接时，如果是首次连接，则需要配对。至于NP和NC谁充当主设备，谁充当从设备，并无硬性规定。但一般好用户体验一定是NP充当主设备，NC充当从设备。

**第二章 ANCS的工作流程**

NP和NC之间的工作流程大致如下：

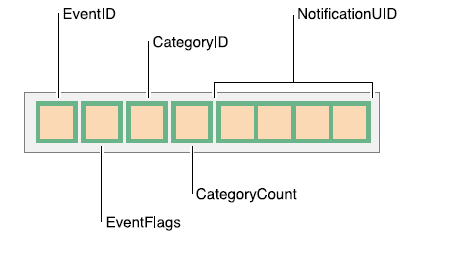
1 NP和NC之间建立连接，如果两个设备是首次建立连接，那么需要配对。

2 NC去发现NP上的ANCS服务，确定是否存在ANCS及确定三条Characteristic的存在性和对应的Characteristic Value和Characteristic Client Configuration的Attribute Handle。

3 NC先使能NP上DS Characteristic的Notify属性（注意，好的经验是先使能它）。

4 NC再使能NP上NS Characteristic的Notify属性。

5 NP上产生一条新的通知，那么会通过NS Characteristic发送Notify到NC，NC收到以后，就可以得知如下信息：



Event ID：这个域有三种定义（**Added=0, Modified=1, Removed=2**）。对于Added和Removed比较好理解，Added即表示新产生了一条通知，Removed即表示在手机上通知已经被查看。而对于Modified，Apple给出的ANCS文档中并没有给出什么情况下会推送该通知。因此，建议外围设备实现ANCS应用时不要理睬Modified类型的通知。

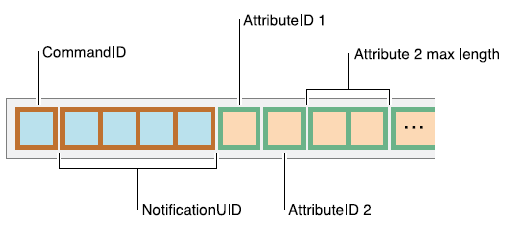
Event Flags：是一个位定义域(Silent=1, Important=2)，即**不重要**或者**重要，不重要**和**重要**都是分类Apple自己定义的，个人认为意义不大，可以不需理睬。

Category ID：这个域的含义比较明确，目前来说最新的ANCS规范1.0中提到了12大类Other=0, IncomingCall=1, MissedCall=2, Voicemail=3, Social=4, Schedule=5, Email=6, News=7, HealthAndFitness=8, BusinessAndFinance=9, Location=10, Entertainment=11。这12大类里面需要解释一下的可能就只有Social，因为Social指的是社交类通讯软件的通知，比如说QQ,、SMS、微信等等都是通过先告诉NC这是一条Social类别的通知。可想而知，光从NS的Notify想区分出来SMS、QQ是不可能的。因此要区分的话NC即必须要请求更多的内容才能得知。

Category Count：指的是该类别下，当前有多少条通知没有在NP（手机上）查看。

Notification UID：是一个唯一的识别号。用来标识一次新的推送。如果NP上先产生一条QQ消息，再产生一条短信，那么这两条推送的Notification UID是不同的。NC就是利用这个ID号去请求具体的通知对应的内容。有了这个ID号，NC和NP都能将不同通知区分开来。

6 NC收到第5个步骤中的NS的Notify后，生成一个请求报文，以Reliable Write的方法写到CP Characteristic。请求报文满足如下



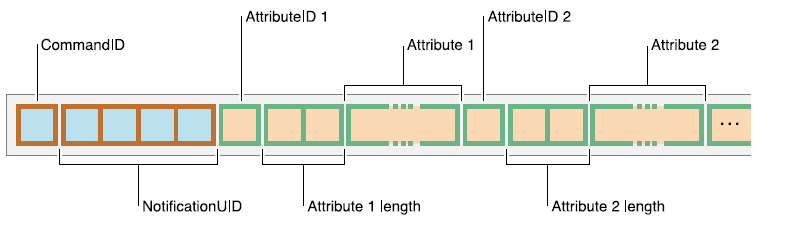
Command ID永远都等于0。

Notification UID来自于NS Notify。

Attribute ID x：该报文中从前往后第x个Attribute ID。Attribute ID有如下几个AppIdentifier=0，Title=1， Subtitle=2， Message=3， MessageSize=4，Date=5。其中Title、SubTitle和Message在NP上可能会出现很长的字符串，因此考虑到NC的缓冲能力，允许NC指定只请求指定大小的内容，即这三条Attribute ID，其后面需要出现Attribute x max Length域。注意x表示，这一个请求报文可以携带多个要请求的信息，比如说一次性可以请求App Identifier、Title、SubTitle、Date等等。

Attribute x max Length：指NC想要从NP上请求多少字节的信息。

7 NP收到CP上的请求后，会将按照请求中Attribute ID出现的顺序、应答回应报文，应答报文格式内容如下：



CommandID：永远为0。

Notification UID：请求的是哪一次通知（前面已经提到NP会为每一次通知都分配一个唯一识别号用来标识当前通知有别于其他通知）。

Attribute ID x：和请求报文中的Attribute ID x出现的顺序一致。

Attribute x length：表示当前请求报文中对应的消息长度。这里的length和请求报文出现的max length域还是有一点关系的。即如果Attribute ID后要求跟max length域，那么应该Value(length field in rsp) ≤ Value(max length in req)。

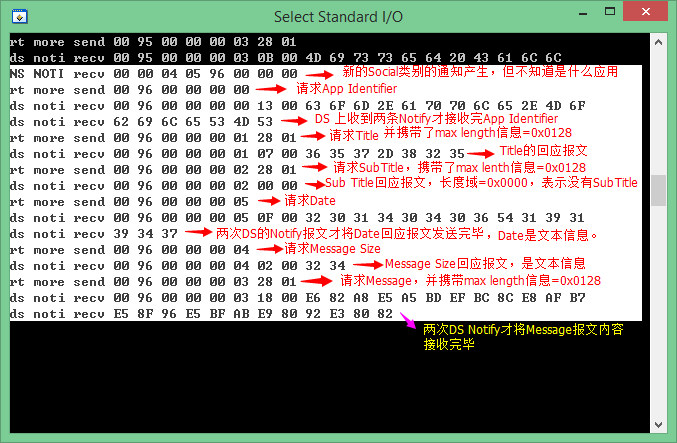
**第三章 实例分析**

下图中一条短信的通知产生时，底层的工作工作过程。

“NS NOTI recv”：表示NP（手机）发送NS Characteristic的Notify给NC（外围设备。

“rt more send”：表示NC将Retrieve More Request发送给了NP。

“ds noti recv”：表示NP发送DS Characteristic的Notify给NC。



值得注意的时，对于DS的Notify，NP常常两条一起连续发送给NC，这是由于回复报文太长不足以在一次Notify报文中装载。并且第二条Notify往往不再携带报头信息，而是可以直接拼接上一条Notify而形成一个完整的回应报文，那么这里在编程时就要注意区分了。

上图中的报文内容解析如下：



**第四章 缺陷**

缺陷是无法改进的，我们只能适应它。

缺陷1 使用原有App Identifier定义来在ANCS中来标识应用，这个是最失败的设计。

众所周知，App Identifier是字符串格式的，下面是几个例子。

com.apple.MobileSMS （短信的App Identifier 19字节）。

com.apple.mobilephone （来电和未接来电的App Identifier 21字节）。

com.tencent.mqq （QQ的App Identifier 15字节）。

这个缺陷使得ANCS不得不定义一个包含分类信息的NS Characteristic，它的Notify可以先将通知分为一个大类，但是NS Notify给的信息不够直接，比如说Social里面可能包含多个我们常常需要关心的App，比如说短信、QQ等等、无法及时区分。

另外字符串会加大NC设备的运算量，如果苹果后台有一个数据库，每一款应用有对应一个唯一的ID号，哪怕是16字节整数的，都是可以很方便的标识一个应用的。

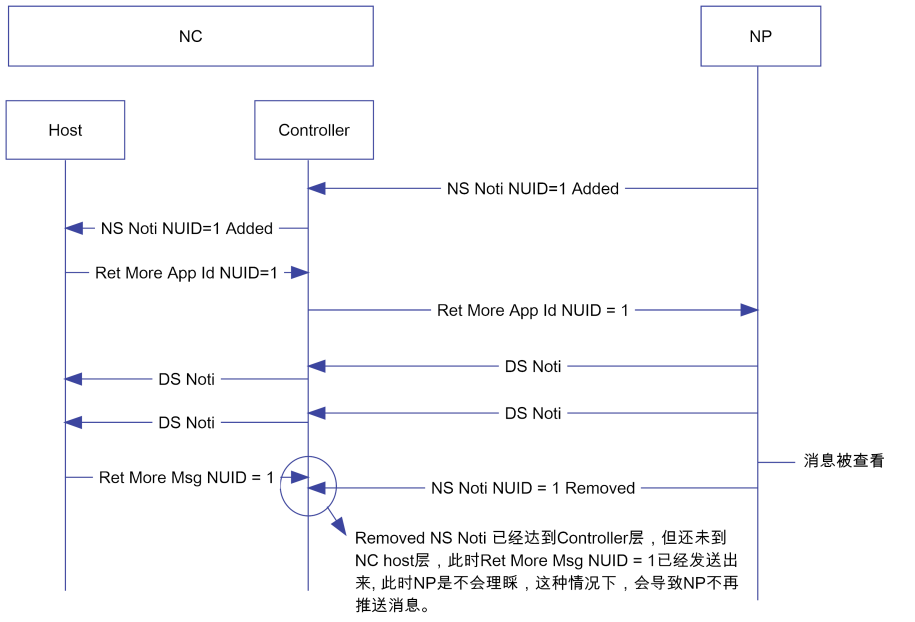
缺陷2，请求更多信息的报文中Attribute ID后面有的需要跟一个max length域，有的不需要跟。这个也是一个缺陷。这种定义会使得ANCS的请求更多信息失去了良好的可扩展性。

**第五章 易出现问题分析**

1 Incoming Call消息Added时，无法请求通知产生时的Date，请求时会得到空返回。

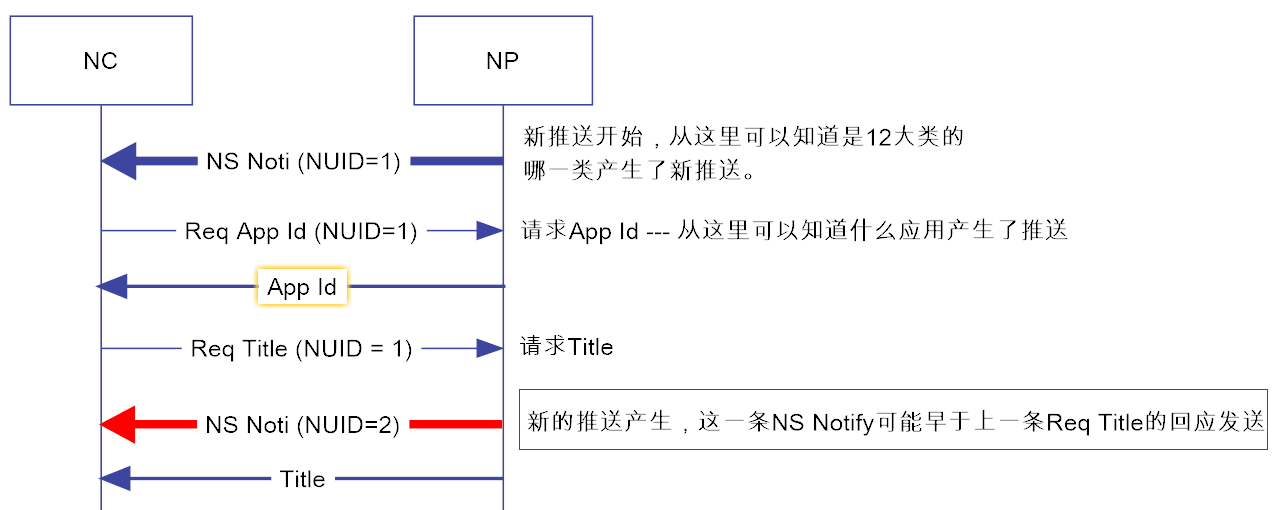
但是奇怪的是在接电话时，如果有另外一个电话来电时，会请求出Date信息。

2 如果对Removed的消息进行请求更多信息的操作，那么ANCS会出现NP不再推送消息给NC，需要断开连接再重新建立连接才有效。这一点比较容易发生，见下图：



这种问题最容易出现，因此NC端应用程序必须防范这一招。很有必要的是，当每一次Retrieve More Request发出来后，NC必须建立一个超时机制，如果在超时后任然没有得到回应，就应该立即断开连接，并重新广播。

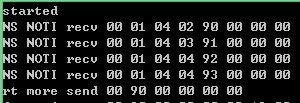
3 ANCS的底层DS和NS两个Characteristic的Notify是具有并发性的，即如果上一次请求更多消息的过程还在进行中，很有可能NP上会来一个电话或者收到一条短信，此时NS上很有可能会发出新的Notify出来。因此NC的程序设计时必须注意这个并发性。请参考第六章中编程建议。下面是一个示例，红颜色的箭头即表示了并发性的存在。



**第六章 NC端编程建议**

第一章到第五章对ANCS的基本工作流程、示例以及易出现的问题进行了分析，本章主要介绍ANCS的编程建议。

1 NC端应建立对NS Notify的队列，队列尽量要长。NS Notify只有8个字节。下图是一个建立NS Notify队列的主要原因，“started”是上电就打印出来的，上电后“NS NOTI recv”打印了一连串4个。连接刚刚建立时，有可能在连接还没有建立的时候出NP上产生了很多通知。因此，连接刚刚建立，就会有一连串的NS Notify发送到NC。NC如果不缓冲，会出乱子。



2 NC发送Retrieve More Information时，建立超时保护，超时时间设置3秒为适宜。超时发生，立刻断开连接并重新广播。

3 Incoming Call不会推送Date，因此对于这种即时消息，如果需要知道时间戳，那么本地最好建立利用TIME Profile，先从NP端同步时间，再打开NS，DS的Notify属性，Incoming Call发生时，从本地取发生时间。

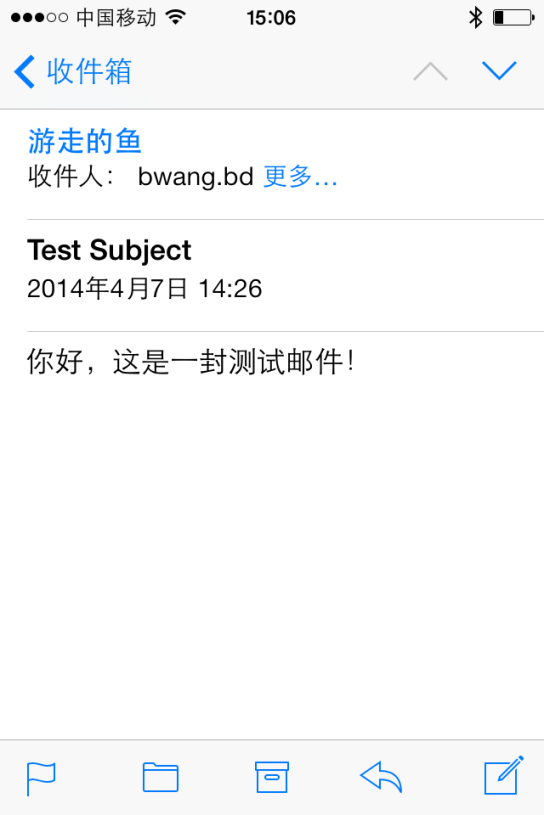
4 对Retrieve More Information尽量使用Reliable Write方法，方便以后扩展，这个也是协议要求。

6 断开连接后，一定要清空NS队列。否则前一次连接的请求不会得到任何回应。

7 如果NS Notify中携带的推送是Incoming Call和Missed Call，那么不需要请求App Identifier，只需要请求Title就可以获取到来电人的名字或者号码。

8 如果NS Notify中携带的是Email，那么可以直接请求Title（发件人）、Sub Title、Message

Title对应的是发件人（昵称或者邮箱地址）、Sub Title对应的邮件标题，Message对应的邮件内容，见下图，左边是iPhone上。



**第七章 各种规律**

1 Incoming Call相关

1.1 Added和Modified基本上同时到来

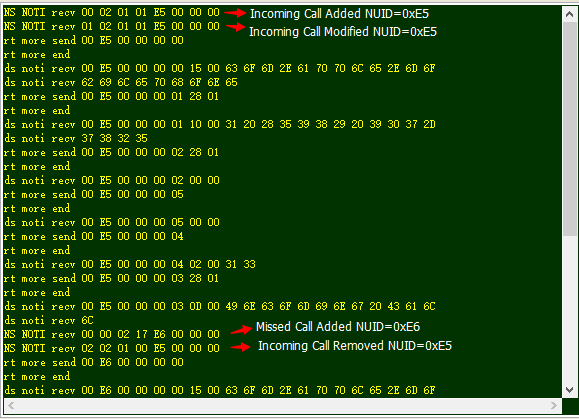
当来电时，会连续推送Added和Modified信息过来，而Added和Modified携带相同的Notification UID，因此请求更多信息时得到的是相同的消息。见1.2节中截图。

1.2 Removed何时到来？

情况一：当电话在拨打方挂断时，会先从NP端推送一个Missed Call Added通知（Notification UID有别于Incoming Call Added及Modified），再紧跟着一个Incoming Call Removed (Notification UID相同于先前的Incoming Call Added和Modified)。

情况二：电话拨打方不主动挂电话、接收方（NP）也一直不接听电话，一段时间后，电话运营商会认为电话接收方（NP）无人接听而终止本次呼叫。此时会出现与情况一相同的现象。

见下图log：



1.3 当接收方NP主动拒接电话时现象

此时可能NP可能会推送一连串的Incoming Call Removed的NS Notify给NC。所以一定要注意。请参考下面图。



2 国内使用美版iPhone时(iPhone带卡贴)

在手机上首先要在通讯录里面加上+86前缀才能在来电时，在手机上显示对方的在通讯录中保存的名称。如果不加+86，来电时NP上显示的就是对方的号码。而这种手机在ANCS工作时，请求Title时，NP推送给NC的永远是号码（不管通讯中是否保存、怎么保存）见下图：

猜测其他一切未解锁版本（国内、国外）的iPhone都会有这样的现象。



**第八章 常见的App对应的Identifier**

1腾讯公司

1.1 QQ：com.tencent.mqq，Social

Title：QQ；

SubTitle：空

Message：好友在QQ中的昵称:信息

1.2 QQ Mail： com.tencent.qqmail，Other

Title: QQ邮箱

SubTitle:空

Message:邮箱昵称+回车符+标题

1.3 微信：com.tencent.xin，Social

Title:微信

Message:昵称:+信息内容

1.4 腾讯视频：com.tencent.live4ohpone，Entertainment

Title: 腾讯视频

SubTitle:空

Message：广告的文本内容

2 iPhone自带应用

2.1来电： com.apple.mobilephone，Incoming Call

Title：来电人的用户姓名或者号码

SubTitle：空

Message：Incoming Call

请求Date时为空

2.2未接来电：com.apple.mobilephone，Missed Call

Title：来电人的用户姓名或者号码

SubTitle：空

Message：Missed Call

2.3 手机短信：com.apple.MobileSMS，Social

Title：发件人的用户姓名（电话簿中的用户姓名）或者号码

SubTitle：空

Message：短信内容

2.4 邮件：com.apple.mobilemail，Email

Title：邮箱昵称

SubTitle：邮件标题

Message：邮件内容

3 优酷

优酷视频：com.youku.YouKu，Entertainment