iOS 11 迟来的 NFC

前言

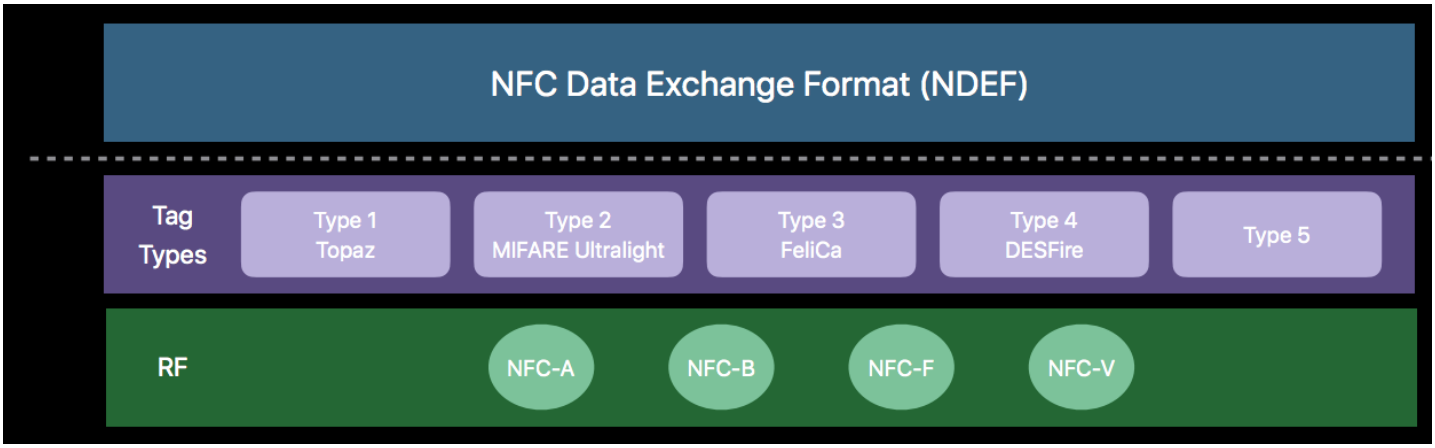
NFC这个词相信大家现在都已经不陌生了，各大城市的地铁、商场等等支持NFC支付一度成为头条的热点。其实很早之前就已经有二维码和NFC的诞生了，但是由于二维码成本低廉，技术门槛相对较低，因此，二维码迅速抢占了移动支付的市场，但是，与此同时NFC的发展并未因此停止。其实在安卓端的NFC发展已经非常迅猛了，只是我们的苹果爸爸迟迟不肯带我们一起玩NFC。终于，在去年的WWDC上，苹果宣布**~~“开放”~~**其NFC接口，这为以后NFC的应用开发提供了更多的可能。

关于NFC

NFC（Near Field Communication）近场通信，当两个设备相互靠近时能进行信息交流的一个技术。使用了NFC技术的设备（比如手机）可以在彼此靠近的情况下进行数据交换，是由非接触式射频识别(RFID)及互连互通技术整合演变而来，通过在单一芯片上集成感应式读卡器、感应式卡片和点对点通信的功能，利用移动终端实现移动支付、电子票务、门禁、移动身份识别、防伪等应用。目前，苹果的CoreNFC对NFC的格式支持有限，暂时仅支持NDEF格式。

关于NDEF

NDEF（NFC Data Exchange Format）是一种能够在NFC设备或者标签之间进行信息交换的数据格式。NDEF格式由各种 `NDEF Messages` 和 `NDEF Records` 组成。NDEF格式使用了一种容易理解的格式来存储和交换信息，如：URI、纯文本等等。NFC标签，像 `Mifare Classic` 卡片可以配置为NDEF标签,通过一个NFC设备写入的数据可以被其他NDEF兼容的设备访问。NDEF消息还可以用于两个活跃的NFC设备之间“点对点”模式交换数据。



NDEF Messages

`NDEF Messages` 是 `NDEF Records` 交换机制的基础，每一个message包含一个或多个records。

NDEF Records

`NDEF Records` 包含一个特定的 `payload` ，并且有以下结构来标识内容和记录大小:

```
Bit 7    6      5      4      3      2      1      0
------ ------ ------ ------ ------ ------ ------ ------
[ MB ] [ ME ] [ CF ] [ SR ] [ IL ] [       TNF        ]
[                   TYPE LENGTH                      ]
[                 PAYLOAD LENGTH                     ]
[                   ID LENGTH                        ]
[                  RECORD TYPE                       ]
[                      ID                            ]
[                   PAYLOAD                          ]
```

Record Header（记录头）

记录头包含了很多重要的信息，它占用3个位来标识遵循 `TNF` 协议的记录的类型，讲人话就是这3个位用来表示这条记录的类型的。

TNF: Type Name Format 字段

一条NDEF记录的类型名称是一个3个位的数值，用来描述这条记录的类型，并且可以用来设置对该记录中其它的结构和内容的期望。简单的说就是这3个位不仅可以表示该条记录的类型，也可以在一定程度上决定了该条记录接下来的数据结构。可能的记录名称如下表：

TNF Value	Record Type
0x00	Empty Record 表明这条记录没有类型、id或有效payload。这个记录类型一般用于新格式化的NDEF卡上，因为NDEF标签必须有至少一个NDEF纪录。
0x01	Well-Known Record 表明记录类型字段使用RTD类型名称格式。这种类型名称用一个Record Type Definition (RTD)来存储任何指定的类型，例如：存储RTD文本、RTD URIs等等。同时，这是一种比较常用的也比较有用的记录类型。
0x02	MIME Media Record 表明payload是这条NDEF记录分块的中间或者最后一块。
0x03	Absolute URI Record 表明这条记录的类型字段一定包含一个URI字段。
0x04	External Record 表明这条记录的类型字段包含一个RTD格式的外部字段。
0x05	Unknown Record 表明payload的类型未知。
0x06	Unchanged Record 未发生变化的记录类型，释同MIME Media Record。

IL：ID Length 字段

`IL` 是ID长度的标志位，用来表示下面的 `ID Length` 字段是否省略。如果这个位设置为0，则该条记录中省略 `ID Length` 。

SR：Short Record 位

短记录标志位，如果下面的 `PAYLOAD LENGTH` 字段小于等于一个字节，则该位设置为1 。

CF：Chunk Flag

块标识位，用于标识当前块是第一个记录块还是中间的记录块。

ME：Message End

结束标志位，用于标识当前记录是否是当前Message的最后一条记录。

MB：Message Begin

起始标志位，用于标识当前记录是否是当前Message的第一条记录。

Type Length

表示类型字段的长度。对于上面 `TNF` 字段描述中的某些值，该字段一直为0 。

Payload Length

表示该条记录的payload字段长度。如果上面的 `SR` 字段设置为1，则该字段占用1个字节的长度，但是如果 `SR` 设置为0，则该字段将有32个位，占用4个字节的长度。

ID Length

表示该条记录的ID的长度。只有当上面的 `IL` 位置1时该字段才会被省略。

Record Type

表示记录的类型，这个字段的值必须根据 `TNF` 位的设置确定。

Record ID

表示该条记录的ID。当 `IL` 位为0时，该字段省略。

Payload

表示该条记录的payload，该字段的长度务必与上面的 `Payload Length` 字段值一致。

关于 Well-Known Records 和 URI Records

首先要说的就是这两个概念的区别，**Well-Known Records（TNF Record Type 0x01）** 是最常用也是最有用的NFC记录类型，它是写在上面说到的TNF字段的三个位里的，它描述的是当前这条NDEF记录的整体类型，相当于一个总的架构决策。而 **URI Records（0x55/‘U’）** 是比较有用的数据类型，它是写在上面 `Recode Type` 字段的一个字节（8个位）里的，它描述的是这条NDEF记录携带的数据信息类型，简单的说就是这条记录携带了什么样的信息。

这里，关于这个URI Records我要多说几句，这个类型可以用来存储例如电话号码、网站地址以及各种协议的链接等等很多有用的信息，它的结构定义如下：

Name	Offset	Size	Description
Identifier Code	0	1 byte	See table below
URI Field	1	N bytes	The rest of the URI (depending on byte 0 above)

第一个字节表示该类型的识别码，这个识别码的主要是用于缩短URI的长度，它的有效值详见下表：

Value	Protocol
0x00	No prepending is done ... the entire URI is contained in the URI Field
0x01	http://www.
0x02	https://www.
0x03	http://
0x04	https://
0x05	tel:
0x06	mailto:
0x07	ftp://anonymous:anonymous@
0x08	ftp://ftp.
0x09	ftps://
0x0A	sftp://
0x0B	smb://
0x0C	nfs://
0x0D	ftp://
0x0E	dav://
0x0F	news:
0x10	telnet://
0x11	imap:
0x12	rtsp://
0x13	urn:
0x14	pop:
0x15	sip:
0x16	sips:
0x17	tftp:
0x18	btspp://
0x19	btl2cap://
0x1A	btgoep://
0x1B	tcpobex://
0x1C	irdaobex://
0x1D	file://
0x1E	urn:epc:id:
0x1F	urn:epc:tag:
0x20	urn:epc:pat:
0x21	urn:epc:raw:
0x22	urn:epc:
0x23	urn:nfc:

后面的N个字节就是用来表示一个URI去掉前面识别码之后剩余的部分，举个例子：例如我们

要将 `https://www.mob.com` 写入，则在第一个字节里我们要写入的是 `0x02`，表示 `https://www.`，接下来要连续写入的就是 `0x6D 0x6F 0x62 0x2E 0x63 0x6F 0x6D`（详细请参考：ASCII码对照表）

以上所有内容就是关于NDEF数据格式的详细说明了，那么这里也很不幸的告诉大家，我们平时直接从某宝、某东或者某某某上买来的NFC卡片（俗称：白卡）都不会是NDEF格式的，所以。。。

将 Mifare Classic Cards 用作 NDEF 标签

Mifare Classic 1K和4K的卡可以被初始化为NFC的NDEF格式标签。关于Mifare的详细介绍请各位老板参见：Mifare维基百科
Mifare Classic 可以配置为NFC论坛兼容的NDEF标签，但必须以某种特定的方式组织它里面的数据才可以。具体要求可以参考如下资料：

AN1304 – NFC Type MIFARE Classic Tag Operation

上面的是使用手册的权威来源，下面将快速的介绍一下将 Mifare Classic Cards 用作 NDEF 标签所涉及的关键概念。

1. Mifare Application Directory (MAD)

Mifare应用程序的目录，为了在Mifare Classic卡片的扇区内存与单个NDEF记录之间建立关系而存在。MAD表明了哪个扇区包含着哪个NDEF记录。关于Mifare应用程序目录的权威信息来源如下：

AN10787 – MIFARE Application Directory (MAD)

> *为了兼容性考虑，官方根据卡片内存的大小定义了两种不同类型的MAD，简单说明一下区别，MAD1可以用在任何卡片上，而MAD2只能用在内存大于1K字节的卡片上。*

2. 存储 NDEF Messages

为了在Mifare Classic卡上存储NDEF消息，消息需要被封装在一个叫做 `TLV 块` 的东西里面。关于 `TLV 块` 的基本结构描述如下：

> *TLV 是三个不同维度的缩写：*
> *T：Tag Field 标签*
> *L：Length Field 长度*
> *V：Value Field 数值*

一个 `TLV 块` 由一个或多个字节组成，这取决于上面三个维度的存在情况，但至少有一个字节，因为 T 字段在每种情况下都是强制性的。

Tag Field

标签字段是唯一必填的字段，使用单个字节来标识 `TLV 块` 的类型，有效值如下：

```
TLV Block Types

  Block Type    Value  Description
  ------------ -----  ------------------------------------
  NULL          0x00   These blocks should be ignored
  NDEF Message  0x03   Block contains an NDEF message
  Proprietary   0xFD   Block contains proprietary information
  Terminator    0xFE   Last TLV block in the data area
```

Length Field

长度字段包含了数值字段的长度（字节），它可以使用一个或三个字节两种不同的方式表示：
一个字节格式就是简单0x00～0xFF的一个字节数值；
三个字节格式的组成如下：

```
  Byte 0:       Always 0xFF to indicate that we are using the three byte format
  Byte 1..2:    Can be a value between 0x00FF and 0xFFFE
```

Value Field

数值字段仅在长度字段存在且不等于0x00时才存在，这个字段就是 `payload` 的存储位置。

Terminator TLV

终止符，是数据区域中的最后一个TLV 块，固定的单个字节：0xFE，这个TLV 块也是是强制性的。

3. 一个带有NDEF记录的Mifare Classic 卡片的内存示例

```
[                     Start of Memory Dump                  ]
-----------------------Sector 0--------------------------
Block 0  3E 39 AB 7F D3 88 04 00 47 41 16 57 4D 10 34 08
Block 1  14 01 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1
```

```
Block 2  03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1
Block 3  00 00 00 00 00 00 78 77 88 C1 00 00 00 00 00 00
------------------------Sector 1-------------------------
Block 4  00 00 03 11 D1 01 0D 55 01 61 64 61 66 72 75 69
Block 5  74 2E 63 6F 6D FE 00 00 00 00 00 00 00 00 00 00
Block 6  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 7  00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00
------------------------Sector 2-------------------------
Block 8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 9  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 11 00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00
------------------------Sector 3-------------------------
Block 12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 13 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 15 00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00
------------------------Sector 4-------------------------
Block 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 17 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 18 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 19 00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00
------------------------Sector 5-------------------------
Block 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 21 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 22 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 23 00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00
------------------------Sector 6-------------------------
Block 24 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 25 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 26 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 27 00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00
------------------------Sector 7-------------------------
Block 28 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 29 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 31 00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00
------------------------Sector 8-------------------------
```

```
Block 32 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 34 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 35 00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00
------------------------Sector 9-------------------------
Block 36 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 37 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 38 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 39 00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00
------------------------Sector 10-------------------------
Block 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 42 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 43 00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00
------------------------Sector 11-------------------------
Block 44 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 45 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 46 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 47 00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00
------------------------Sector 12-------------------------
Block 48 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 49 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 51 00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00
------------------------Sector 13-------------------------
Block 52 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 53 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 54 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 55 00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00
------------------------Sector 14-------------------------
Block 56 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 57 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 58 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 59 00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00
------------------------Sector 15-------------------------
Block 60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 61 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 62 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 63 00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00
```

```
[                    End of Memory Dump                    ]
```

上图示例中在扇区1包含了两个NDEF记录：

第一个记录在扇区1块4的前两个字节：根据上面的描述，每个记录都以TLV 块开头，并且TLV 块的第一个字节（值0x00）指示这是一个空块类型，第二个字节是长度字段，并且也是0x00，因此该记录没有payload，TLV 块的值字段不存在。当Mifare Classic卡首次格式化以确保至少有一条记录存在时，一般会插入此记录。

第二个记录从扇区1块4的第3个字节开始，到块5的第6个字节结束：

```
Block  00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
-----  -----------------------------------------------
04           03 11 D1 01 0D 55 01 61 64 61 66 72 75 69
05     74 2E 63 6F 6D
```

同样，对于前两个字节，根据TLV 块的描述，第一个字节 0x03 表示这是一个NDEF Message类型，第二个字节 0x11 表示该块的数据长度是17个字节。对于接下来17个字节的分析如下表：

Byte(s)	Value	Description
04:04	0xD1	NDEF记录的记录头，详细解释参考上面的NDEF记录描述部分。位分配如下： TNF = 0x01 表示这是一个Well-Known类型的记录 IL = 0 表示没有ID字段 SR = 1 表示这是一个短记录 CF = 0 表示这个块不是第一块 ME = 1 表示当前记录为最后一条记录 MB = 1 表示Message的开始
04:05	0x01	NDEF记录类型的长度，1个字节，因为下面的记录类型值是0x55
04:06	0x0D	表示payload的长度，13个字节
04:07	0x55	NDEF记录的类型，0x55表示URI类型
04:08	0x01	表示payload开始，后面的将是payload中的内容
04:09..05:04	...	payload内容的16进制数据

最后一个字节，0xFE 是TLV 块的终止符，表示这个块的结束，这个没什么好说的了～～

> *相关参考链接：*
>
> - *NXP官方网站*
> - *NFC论坛*
> - *Mifare Classic S50 技术详解*

iOS CoreNFC

下面将通过一个简单的示例来演示怎么使用苹果爸爸推出的CoreNFC，该示例仅可以用来读取存储在卡片上的NDEF格式的信息。

> *为此，我使用了* `STM32F407` *单片机与* `Adafruit PN532 Shield` *NFC读写模块配对，将信息写入NDEF格式的卡片上。本文中，我不会记录如何将普通的* `Mifare Classic` *卡片格式化成NDEF格式的卡片以及如何将数据写入到NDEF卡片中（iOS的CoreNFC暂时不支持写入）。*

我们的app要使用NFC必须要进行应用授权：首先要创建一个支持NFC的证书，并且开启 `NFC Tag Reading` ，如下图：

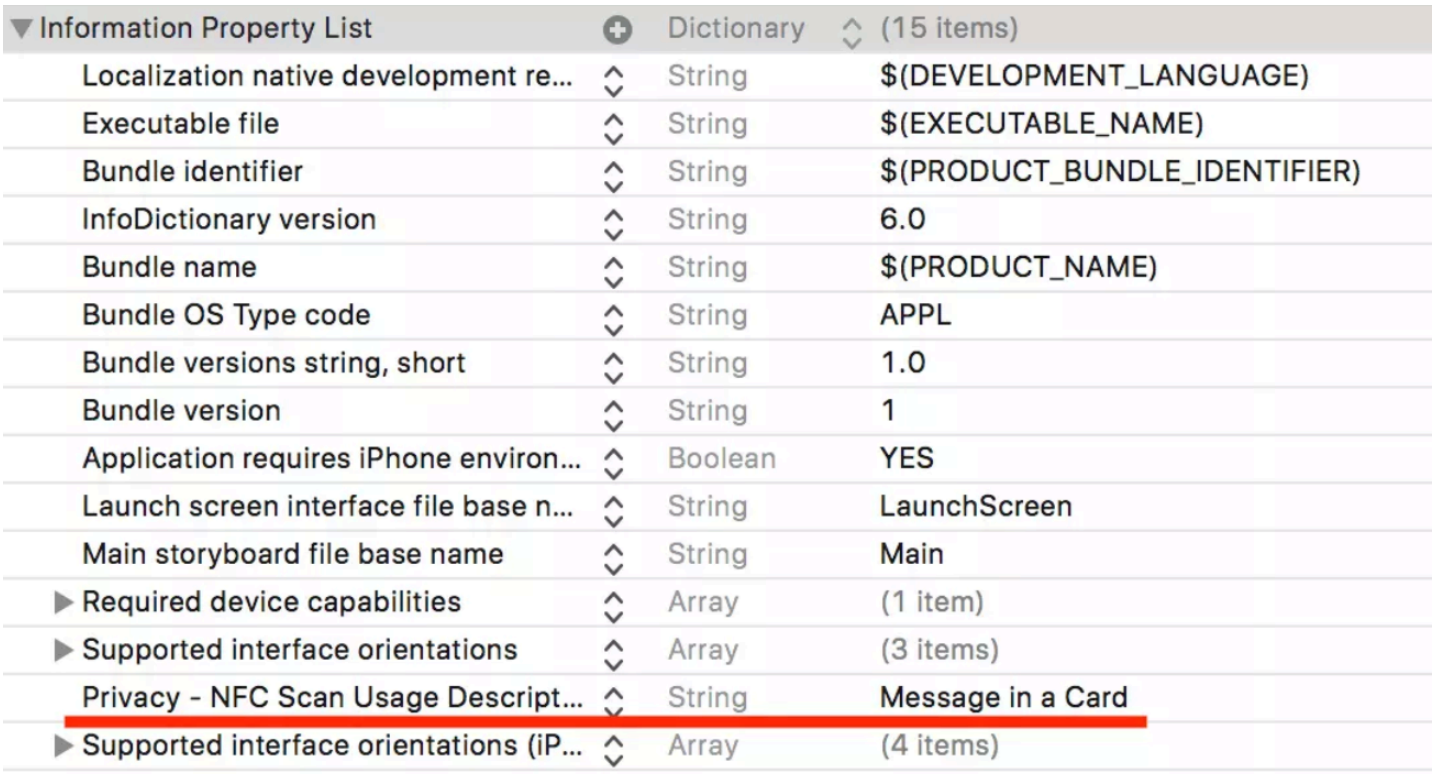
Confirm your App ID.

To complete the registration of this App ID, make sure your App ID information is correct, and click the submit button.

App ID Description:	**nfc**
Identifier:	
App Groups:	Disabled
Apple Pay:	Disabled
Associated Domains:	Disabled
Data Protection:	Disabled
Game Center:	Disabled
HealthKit:	Disabled
HomeKit:	Disabled
Hotspot:	Disabled
iCloud:	Disabled
In-App Purchase:	Disabled
Inter-App Audio:	Disabled
Multipath:	Disabled
Network Extensions:	Disabled
NFC Tag Reading:	**Enabled**
Personal VPN:	Disabled
Push Notifications:	Disabled
SiriKit:	Disabled
Wallet:	Disabled
Wireless Accessory Configuration:	Disabled

导入证书之后，我们需要进行Info.plist配置 `Privacy - NFC Scan Usage Description` 权

限，如下图：



要实现NFC功能，我们得先导入 `CoreNFC.framework` ，并导入其头文件 `#import <CoreNFC/CoreNFC.h>` 目前为止，iOS模拟器还不支持CoreNFC，只能使用真机调试。

1. 初始化Session

与二维码扫描等类似，NFC 也具备一个用于信息交互的Session，并且这个Session要在使用期间一直持有，所以初始化Session代码如下：

```
@interface NFCTableViewController ()<NFCNDEFReaderSessionDelegate>

/**
 NFC Session
 */
@property (nonatomic, strong) NFCNDEFReaderSession *nfcSession;

/**
 founded NFC Messages
 */
@property (nonatomic, strong) NSMutableArray<NSArray<NFCNDEFMessage *> *> *
nfcMessages;

@end
```

```
@implementation NFCTableViewController

#pragma mark - initializeNFC

- (void)initializeNFCSession {
    // 创建 Session
    self.nfcSession = [[NFCNDEFReaderSession alloc] initWithDelegate:self q
ueue:dispatch_get_main_queue() invalidateAfterFirstRead:NO];
    // 设置 Session 的提示信息
    self.nfcSession.alertMessage = @"You can scan NFC-tags by holding them
behind the top of your iPhone.";
}

@end
```

2. 启动Session

经过上面的初始化之后，有了Session就可以开启监听了，启动监听很简单，示例代码如下：

```
- (IBAction)startSearchBtnClick:(UIBarButtonItem *)sender {
    NSLog(@"%s", __func__);
    // 启动Session
    [self.nfcSession beginSession];
}
```

Session启动时会自动调出系统的扫描面板，如下图：

3. 监听代理回调

通过上面的方式把Session启动之后，设备就会自动开始扫描NDEF格式的标签信息，当扫描到标签信息，或者发生任何异常时都会通过代理方法回调，所以我们需要监听Session的回调信息，示例如下：

```
#pragma mark - NFCNDEFReaderSessionDelegate

/*!
 * @method readerSession:didInvalidateWithError:
 *
 * @param session   The session object that is invalidated.
 * @param error     The error indicates the invalidation reason.
 *
```

```
 * @discussion      Gets called when a session becomes invalid.  At this po
int the client is expected to discard
 *                  the returned session object.
 */

/**
 NFC 读取的session发生错误或session过期时回调，此时客户端应当丢弃返回的session，即此s
ession不可重用。

 @param session 发生错误的session
 @param error 错误信息
 */
- (void)readerSession:(NFCNDEFReaderSession *)session didInvalidateWithErro
r:(NSError *)error {
    if (error)
    {
        NSLog(@"NFC-Session invalidated: %@", error);
    }
    if (self.nfcSession)
    {
        // 关闭当前session
        [self.nfcSession invalidateSession];
        self.nfcSession = nil;
    }
    // 重新初始化一个新的session
    [self initializeNFCSession];
}


/*!
 * @method readerSession:didDetectNDEFs:
 *
 * @param session   The session object used for tag detection.
 * @param messages  Array of @link NFCNDEFMessage @link/ objects. The order
 of the discovery on the tag is maintained.
 *
 * @discussion      Gets called when the reader detects NFC tag(s) with NDE
F messages in the polling sequence.  Polling
 *                  is automatically restarted once the detected tag is rem
oved from the reader's read range.
 */

/**
 当NFC Session在轮询队列中读取到NDEF信息时回调，此时轮询会自动重启一次再次检测NFC标签是
否离开了读取范围。（原理类似于机械按键的防抖动）
```

```
 @param session 读取Session
 @param messages 读取到的信息
 */
- (void)readerSession:(NFCNDEFReaderSession *)session didDetectNDEFs:(NSArray<NFCNDEFMessage *> *)messages {
    NSLog(@"New NFC Messages %zd detected:", messages.count);
    for (NFCNDEFMessage *message in messages) {
        NSLog(@"- %zd Records:", message.records.count);
        for (NFCNDEFPayload *record in message.records) {
            NSLog(@"\t- TNF(TypeNameFormat): %@", [self formattedTypeNameFormat:record.typeNameFormat]);
            NSLog(@"\t- Payload: %@", [[NSString alloc] initWithData:record.payload encoding:NSUTF8StringEncoding]);
            NSLog(@"\t- Type: %@", [[NSString alloc] initWithData:record.type encoding:NSUTF8StringEncoding]);
            NSLog(@"\t- Identifier: %@", [[NSString alloc] initWithData:record.identifier encoding:NSUTF8StringEncoding]);
        }
    }

    [self.nfcMessages addObject:messages];

    dispatch_async(dispatch_get_main_queue(), ^{
        [self.tableView reloadData];
    });
}
```