NFC近场通信

[android](http://www.oschina.net/search?scope=blog&q=android) [NFC](http://www.oschina.net/search?scope=blog&q=NFC)

目录[-]

 [NFC基础](http://my.oschina.net/wolfcs/blog/178618#OSC_h1_1)

 [标签Dispatch系统](http://my.oschina.net/wolfcs/blog/178618#OSC_h2_2)

 [NFC标签是如何映射到MIME type和URI的](http://my.oschina.net/wolfcs/blog/178618#OSC_h3_3)

 [NFC标签如何被dispatched到应用程序](http://my.oschina.net/wolfcs/blog/178618#OSC_h3_4)

 [在Android Manifest中请求NFC访问权限](http://my.oschina.net/wolfcs/blog/178618#OSC_h2_5)

 [过滤NFC Intents](http://my.oschina.net/wolfcs/blog/178618#OSC_h2_6)

 [ACTION\_NDEF\_DISCOVERED](http://my.oschina.net/wolfcs/blog/178618#OSC_h3_7)

 [ACTION\_TECH\_DISCOVERED](http://my.oschina.net/wolfcs/blog/178618#OSC_h3_8)

 [ACTION\_TAG\_DISCOVERED](http://my.oschina.net/wolfcs/blog/178618#OSC_h3_9)

 [从intents中获取信息](http://my.oschina.net/wolfcs/blog/178618#OSC_h3_10)

 [创建NDEF记录的通用类型](http://my.oschina.net/wolfcs/blog/178618#OSC_h2_11)

 [TNF\_ABSOLUTE\_URI](http://my.oschina.net/wolfcs/blog/178618#OSC_h3_12)

 [TNF\_MIME\_MEDIA](http://my.oschina.net/wolfcs/blog/178618#OSC_h3_13)

 [TNF\_WELL\_KNOWN with RTD\_TEXT](http://my.oschina.net/wolfcs/blog/178618#OSC_h3_14)

 [TNF\_WELL\_KNOWN with RTD\_URI](http://my.oschina.net/wolfcs/blog/178618#OSC_h3_15)

 [TNF\_EXTERNAL\_TYPE](http://my.oschina.net/wolfcs/blog/178618#OSC_h3_16)

 [Android应用程序记录](http://my.oschina.net/wolfcs/blog/178618#OSC_h3_17)

 [Beaming NDEF 消息到其他的设备](http://my.oschina.net/wolfcs/blog/178618#OSC_h2_18)

近场通信(NFC)是一组短距离的无线技术，典型地，需要距离为4cm或更短来初始化一个连接。NFC允许你在一个NFC标签和一个Android设备之间，或在两个Android设备之间共享有效载荷很小的数据。

标签的范围可能非常复杂。简单的标签只提供读和写的语义，有时会通过单次可编程区域来使卡变得只读，更复杂的标签提供了数学操作，并具有加密硬件来验证对于一个扇区的访问。最精致的标签包含操作环境，通过标签上执行的代码而允许复杂的交互。标签中存储的数据也可以以各种各样的格式写入，但Android framework APIs中的大多数是基于并围绕着一种称为NDEF(NFC Data Exchange Format)的[NFC Forum](http://www.nfc-forum.org/)标准的。

具有NFC的Android设备同时支持三种主要的操作模式：

1. **读/写模式**，允许NFC设备读取和/或写入被动式的NFC标签。
2. **P2P模式**，允许NFC设备与其他的NFC设备交换数据；这种操作模式被用于Android Beam。
3. **卡模拟模式**，允许NFC设备本身就像一个NFC卡一样运行。然后被模拟的NFC卡可以由一个外部的NFC读取器来访问，比如一个NFC销售点的终端。

[**NFC基础**](http://developer.android.com/guide/topics/connectivity/nfc/nfc.html)

这篇文档描述了Android是如何处理发现的NFC标签的，以及它是如何将与特定应用相关的数据同志给该应用的。这份文档也会说明如何在你的应用中使用NDEF数据，及支持Android基本NFC功能集合的framework APIs的一个概览。

[**高级NFC**](http://developer.android.com/guide/topics/connectivity/nfc/advanced-nfc.html)

这篇文档描述了启用Android支持的各种标签技术的APIs。当你不使用NDEF数据时，或当你在使用的NDEF数据Android不能完全理解时，你不得不用你自己的协议栈以原始字节的形式手动地读取或写入标签。在这些情况中，Android提供了一些支持，来探测某中tag技术，并使用你自己的协议栈来打开与标签的通信。

[**Host-based Card Emulation**](http://developer.android.com/guide/topics/connectivity/nfc/hce.html)

这份文档描述了Android设备如何能够不使用一种安全元素而能像NFC卡一样操作，从而使任何Android应用程序来模拟一张卡，并与NFC读取器直接交流。

**NFC基础**

这份文档描述了你可以在Android上执行的基本的NFC任务。它解释了如何以NDEF消息的形式发送和接收NFC数据，并描述了支持这些功能的Android framework APIs。关于更高级的主题，其中包含了对于使用非NDEF数据的讨论，请参见[**高级NFC**](http://developer.android.com/guide/topics/connectivity/nfc/advanced-nfc.html)。

当应用NDEF数据和Android时，有两个主要的用例：

* 从一个NFC标签中读取NDEF数据。
* 通过Android Beam，从一个设备beaming NDEF消息到另一个。

从一个NFC标签中读取NDEF数据由标签dispatch系统来处理，该系统会分析发现的NFC标签，适当地分类数据，并启动一个对分类的数据感兴趣的应用程序。想要处理扫描的NFC标签的应用程序可以声明一个intent filter来请求处理该数据。

Android Beam功能允许一个设备通过物理地轻敲一个在一起的设备来推送一个NDEF消息到另一个设备。相较于向Bluetooth等其他无线技术，这种交互提供了一种更简单的方式来发送数据，因为通过NFC，而不需要手动的设备搜索或配对。当两个设备距离在一定范围时，连接会自动启动。通过一些NFC APIs可以来应用Android Beam，因而，任何应用程序可以在设备间传递信息。比如，Contacts，Browser，和YouTube应用程序使用Android Beam来与其他设备共享联系人，网页和视频。

**标签Dispatch系统**

当屏幕解锁时，Android设备通常都会不停地寻找NFC 标签，除非在设备的Settings菜单中禁用了NFC。当一个Android设备发现一个NFC标签时，想要的行为为使得最适当的activity处理intent，而不需要询问用户要使用什么应用程序。由于设备扫描标签是在一个非常短的距离进行的，要求用户手动选择一个activity很可能会强制他们将设备移离标签而中断连接。你应该开发你的activity来只处理你的activity关心的NFC标签，以避免Activity Chooser的出现。

为了帮助你实现这个目标，Android提供了一个特殊的标签dispatch系统，它可以分析扫描到的NFC标签，解析它们，并试图定位对扫描到的数据感兴趣的应用程序。它通过如下方法来做到这一点：

解析NFC标签，并计算出标签中标识数据载荷的MIME type或这一个URI。

将MIME type或URI及载荷封装进一个intent。开始的这两个步骤在  **NFC标签是如何映射到MIME type和URI的**  部分描述。

依据intent来启动一个activity。这些在 **NFC标签如何被dispatched给应用程序** 中描述。

**NFC标签是如何映射到MIME type和URI的**

在你开始编写你的NFC应用程序之前，理解不同的NFC标签类型，tag dispatch系统是如何解析NFC标签的，及当标签dispatch系统探测到一个NDEF消息时它所做的特殊工作是很重要的。NFC标签可以以多种技术生产，并且也可以以许多种不同的方式来将数据写入它们。Android具有着最多的支持的是NDEF标准，它是由[NFC Forum](http://www.nfc-forum.org/)定义的。

NDEF数据被封装在一个消息(NdefMessage)里面，其中包含了一个或多个记录(NdefRecord)。每个NDEF记录必须是符合你想要创建的记录的类型的规范的。Android也支持其它的不包含NDEF数据的标签类型，你可以通过使用android.nfc.tech包中的类来应用它们。为了学习更多关于这些技术的东西，请参见[**高级NFC**](http://developer.android.com/guide/topics/connectivity/nfc/advanced-nfc.html)主题。应用这些其它的标签类型包括编写你自己的协议栈来与标签通信，因而我们建议你为了开发的简单及最大化的支持Android设备而在可能的时候使用NDEF。

**注意：**想要下载完整的NDEF规范的话，可以到[NFC Forum Specification Download](http://www.nfc-forum.org/specs/spec_license)站点，并参考[创建通用的NDEF记录类型](http://developer.android.com/guide/topics/connectivity/nfc/nfc.html#creating-records)的示例来了解如何构建NDEF记录。

现在你已经有了一些关于NFC标签的背景知识了，接下来的几节将描述关于Android如何处理NDEF格式化的标签的更具体的信息。当一个Android设备扫描到一个包含了NDEF格式化的数据的NFC标签时，它会解析消息，并尝试去及算出数据的MIME类型或标识URI。为了做到这一点，系统会读取[NdefMessage](http://developer.android.com/reference/android/nfc/NdefMessage.html)里的第一条[NdefRecord](http://developer.android.com/reference/android/nfc/NdefRecord.html)来决定如何解释完整的NDEF消息(一个NDEF消息可以具有多个NDEF记录)。在一个符合规范的NDEF消息中，第一个 [NdefRecord](http://developer.android.com/reference/android/nfc/NdefRecord.html)包含如下的域：

**3-bit TNF (Type Name Format，类型名称格式)**

表示如何解释 **变量长度类型域**。有效的值在[Table 1](http://developer.android.com/guide/topics/connectivity/nfc/nfc.html#table1)

**变量长度类型域**

描述了记录的类型。如果使用[TNF\_WELL\_KNOWN](http://developer.android.com/reference/android/nfc/NdefRecord.html#TNF_WELL_KNOWN)，则使用这个域来描述Record Type Definition (RTD)。有效的RTD值在[Table 2](http://developer.android.com/guide/topics/connectivity/nfc/nfc.html#table2)中描述。

**变量长度ID**

记录唯一的一个标识符。这个域通常不使用，但如果你需要唯一的标识一个标签，你可以为它创建一个ID。

**变量长度载荷**

你想要读取或写入的实际数据载荷。一个NDEF消息可以包含多个NDEF记录，因而，不要假设完整的载荷都在NDEF消息的第一个NDEF记录中。

tag dispatch系统使用TNF和类型域来试着将一个MIME type或者URI映射到NDEF消息。如果成功，它会将那些信息封装在一个包含有实际的载荷的[ACTION\_NDEF\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED) intent中。然而，有些情况下，基于第一个NDEF记录，标签dispatch系统无法决定数据的类型。当NDEF数据无法被映射到一个MIME type或者URI时，或者当NFC标签不包含起始的NDEF数据时，这种情况就会发生。在这些情况中，一个包含有标签的技术信息及载荷的[Tag](http://developer.android.com/reference/android/nfc/Tag.html)对象则会被封装进一个[ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED) intent中。

[Table 1](http://developer.android.com/guide/topics/connectivity/nfc/nfc.html#table1)描述了标签dispatch系统是如何将TNF和类型域映射MIME type或URI的。它也描述了那些TNFs不能被映射到一个MIME type或URI。在这些情况中，标签dispatch系统使用[ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED)。

比如，如果标签dispatch系统遇到了一个类型为[TNF\_ABSOLUTE\_URI](http://developer.android.com/reference/android/nfc/NdefRecord.html#TNF_ABSOLUTE_URI)的记录，它会将那个记录的变量长度类型域映射到一个URI。标签dispatch系统会将那个URI与关于标签的其他信息，比如载荷，一起封装进一个[ACTION\_NDEF\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED) intent的数据成员中。另一方面，如果它遇到了一个类型为[TNF\_UNKNOWN](http://developer.android.com/reference/android/nfc/NdefRecord.html#TNF_UNKNOWN)的记录，则它创建一个封装了标签的技术信息的intent来替代。

**Table 1.** 支持的TNFs及他们的映射

|  |  |
| --- | --- |
| Type Name Format (TNF) | Mapping |
| [TNF\_ABSOLUTE\_URI](http://developer.android.com/reference/android/nfc/NdefRecord.html#TNF_ABSOLUTE_URI) | URI based on the type field. |
| [TNF\_EMPTY](http://developer.android.com/reference/android/nfc/NdefRecord.html#TNF_EMPTY) | Falls back to [ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED). |
| [TNF\_EXTERNAL\_TYPE](http://developer.android.com/reference/android/nfc/NdefRecord.html#TNF_EXTERNAL_TYPE) | URI based on the URN in the type field. The URN is encoded into the NDEF type field in a shortened form: <domain\_name>:<service\_name>. Android maps this to a URI in the form: vnd.android.nfc://ext/<domain\_name>:<service\_name>. |
| [TNF\_MIME\_MEDIA](http://developer.android.com/reference/android/nfc/NdefRecord.html#TNF_MIME_MEDIA) | MIME type based on the type field. |
| [TNF\_UNCHANGED](http://developer.android.com/reference/android/nfc/NdefRecord.html#TNF_UNCHANGED) | Invalid in the first record, so falls back to [ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED). |
| [TNF\_UNKNOWN](http://developer.android.com/reference/android/nfc/NdefRecord.html#TNF_UNKNOWN) | Falls back to [ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED). |
| [TNF\_WELL\_KNOWN](http://developer.android.com/reference/android/nfc/NdefRecord.html#TNF_WELL_KNOWN) | MIME type or URI depending on the Record Type Definition (RTD), which you set in the type field. See [Table 2.](http://developer.android.com/guide/topics/connectivity/nfc/nfc.html#well_known) for more information on available RTDs and their mappings. |

**Table 2.** TNF\_WELL\_KNOWN的支持的RTDs及他们的映射

|  |  |
| --- | --- |
| Record Type Definition (RTD) | Mapping |
| [RTD\_ALTERNATIVE\_CARRIER](http://developer.android.com/reference/android/nfc/NdefRecord.html#RTD_ALTERNATIVE_CARRIER) | Falls back to [ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED). |
| [RTD\_HANDOVER\_CARRIER](http://developer.android.com/reference/android/nfc/NdefRecord.html#RTD_HANDOVER_CARRIER) | Falls back to [ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED). |
| [RTD\_HANDOVER\_REQUEST](http://developer.android.com/reference/android/nfc/NdefRecord.html#RTD_HANDOVER_REQUEST) | Falls back to [ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED). |
| [RTD\_HANDOVER\_SELECT](http://developer.android.com/reference/android/nfc/NdefRecord.html#RTD_HANDOVER_SELECT) | Falls back to [ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED). |
| [RTD\_SMART\_POSTER](http://developer.android.com/reference/android/nfc/NdefRecord.html#RTD_SMART_POSTER) | URI based on parsing the payload. |
| [RTD\_TEXT](http://developer.android.com/reference/android/nfc/NdefRecord.html#RTD_TEXT) | MIME type of text/plain. |
| [RTD\_URI](http://developer.android.com/reference/android/nfc/NdefRecord.html#RTD_URI) | URI based on payload. |

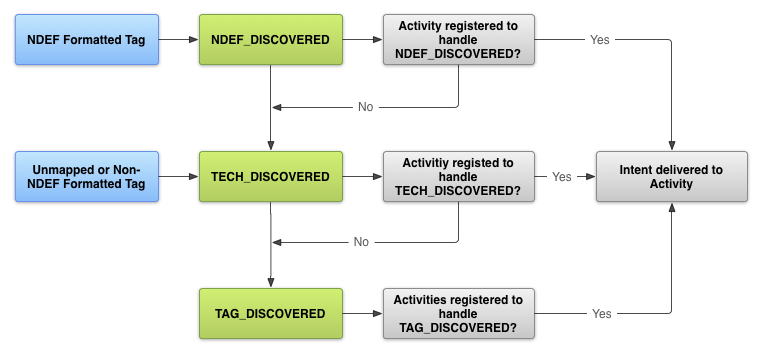
**NFC标签如何被dispatched到应用程序**

当标签dispatch系统完成了创建一个封装了NFC标签及它的标识信息的intent时，它会将那个intent发送给一个过滤了那个intent的对那个inent感兴趣的应用程序。如果有多于一个应用程序可以处理那个intent，则Activity Chooser会出现，以使用户可以选择Activity。标签dispatch系统定义了三种intents，按优先级从高到低排列如下：

1. [ACTION\_NDEF\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED)：当一个包含了一个NDEF载荷的标签被扫描到时，这个intent用于启动一个Activity，它属于一个可识别类型。这是最高优先级的intent，只要可能，标签dispatch系统会在任何其他的intent之前，试着以这种intent来启动一个Activity。
2. [ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED)：如果没有activities注册了要去处理[ACTION\_NDEF\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED) intent，则标签dispatch系统会试着以这种intent启动一个应用程序。如果标签被扫描到包含了NDEF数据但无法被映射到一个MIME type或URI，或者标签不包含NDEF数据但属于一个已知的标签技术，则会直接以这种intent启动(不会先启动[ACTION\_NDEF\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED))。
3. [ACTION\_TAG\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TAG_DISCOVERED)：如果没有activities处理[ACTION\_NDEF\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED)或者[ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED) intents。

标签系统工作的基本流程如下：

1. 试着以标签dispatch系统解析NFC标签时创建的intent来启动一个Activity ([ACTION\_NDEF\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED)或[ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED))。
2. 如果没有activities过滤那样的intent，则试着以更低优先级的intent ([ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED)或[ACTION\_TAG\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TAG_DISCOVERED))来启动一个Activity，直到一个应用程序过滤了那个intent或直到标签dispatch系统试遍了所有可能的intents。
3. 如果没有应用程序过滤任何一种intent，则什么都不做。

[](http://static.oschina.net/uploads/img/201311/24171458_R5Gb.png)

**Figure 1.**Tag Dispatch System

只要有可能，请使用NDEF消息和[ACTION\_NDEF\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED) intent，因为它是三种中最规范的一种。相对于另外两种intents，这种intent允许你在一个更合适的时间点来启动你的应用程序，并给用户一个更好的体验。

**在Android Manifest中请求NFC访问权限**

在你访问一个设备的NFC硬件并适当的处理NFC intents之前，在你的AndroidManifest.xml文件中声明如下的项目：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 1 | <uses-permission android:name="android.permission.NFC" /> |

你的应用程序能支持的最小的SDK版本。API level 9只能通过[ACTION\_TAG\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TAG_DISCOVERED)有限地支持标签dispatch，并只能通过[EXTRA\_NDEF\_MESSAGES](http://developer.android.com/reference/android/nfc/NfcAdapter.html#EXTRA_NDEF_MESSAGES) extra来访问NDEF消息。没有其他的标签属性或I/O操作可用。API level 10包含了广泛的读者/写者支持及前景NDEF推送，而API level 14，通过Android Beam和额外的一些用于创建NDEF记录的方便的方法，提供了一种更简单的方式来推送NDEF消息到其他的设备。

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 1 | <uses-sdk android:minSdkVersion="10"/> |

uses-feature元素使你的应用程序在Google Play中只会显示给那些具有NFC硬件的用户：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 1 | <uses-feature android:name="android.hardware.nfc" android:required="true" /> |

如果你的应用程序使用了NFC功能，但那个功能对于你的应用程序来说不是决定性的，则你可以省略uses-feature元素，而通过检查[getDefaultAdapter()](http://developer.android.com/reference/android/nfc/NfcAdapter.html#getDefaultAdapter(android.content.Context))返回值是否为null在运行时检查NFC的可用性。

**过滤NFC Intents**

为了在你想要处理的NFC标签被扫描到时启动你的应用程序，你的应用程序可以在Android manifest中过滤一个，两个或所有的三个NFC intents。然而，为了当你的应用程序启动时能有最大化的控制权，你通常想要过滤[ACTION\_NDEF\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED)。[ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED) intent是[ACTION\_NDEF\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED)的替补，当没有应用程序过滤[ACTION\_NDEF\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED)或当载荷不是NDEF时使用。过滤[ACTION\_TAG\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TAG_DISCOVERED)通常是一个很泛泛的过滤种类。许多应用程序在过滤 [ACTION\_TAG\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TAG_DISCOVERED)之前，将会先过滤[ACTION\_NDEF\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED)或[ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED)，因而你的应用程序将会有很低的被启动的可能性。只要当没有安装其他的应用程序来处理[ACTION\_NDEF\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED)或[ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED) intent的情况下，才会使用[ACTION\_TAG\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TAG_DISCOVERED)来作为最后的过滤应用程序的依据。

由于NFC标签部署变化无常，而且大多不在你的控制之下，则这不总是可能的，这也是为什么在需要的时候你可以使用其他两种intents的替补。当你可以控制标签的类型及写入的数据时，则建议你使用NDEF来格式化你的标签。下面的几节描述如果过滤每种类型的intent。

**ACTION\_NDEF\_DISCOVERED**

为了过滤[ACTION\_NDEF\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED) intents，则声明包含有你想要过滤的数据的类型的intent filter。下面的例子过滤具有一个值为text/plain的MIME type的[ACTION\_NDEF\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED) intent。：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 1 | <intent-filter> |
| 2 | <action android:name="android.nfc.action.NDEF\_DISCOVERED"/> | |

|  |  |  |
| --- | --- | --- |
| 3 | <category android:name="android.intent.category.DEFAULT"/> | |
| 4 | <data android:mimeType="text/plain" /> |

|  |  |
| --- | --- |
| 5 | </intent-filter> |

下面的例子过滤一个形为http://developer.android.com/index.html的URI：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 1 | <intent-filter> |
| 2 | <action android:name="android.nfc.action.NDEF\_DISCOVERED"/> | |

|  |  |  |
| --- | --- | --- |
| 3 | <category android:name="android.intent.category.DEFAULT"/> | |
| 4 | <data android:scheme="http" |

|  |  |  |
| --- | --- | --- |
| 5 | android:host="developer.android.com" | |
| 6 | android:pathPrefix="/index.html" /> |

|  |  |
| --- | --- |
| 7 | </intent-filter> |

**ACTION\_TECH\_DISCOVERED**

如果你的activity过滤 [ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED) intent，则你必须创建一个XML资源文件在一个tech-list集合中描述你的activity支持的技术。如果一个tech-list集合是标签支持的技术的子集，则你的activity将被认为是一个匹配，你可以通过调用[getTechList()](http://developer.android.com/reference/android/nfc/Tag.html#getTechList())来获取标签支持的技术。

比如，如果被扫描到的标签支持MifareClassic，NdefFormatable，和NfcA，则你的tech-list集合必须描述所有的三种、两种或一种技术(并且没有其它)，以使得你的activity能够被匹配上。

下面的例子定义了所有的技术。你可以移除你不需要的那些。将这个文件(你可以以任何你喜欢的名字命名)保存在<project-root>/res/xml目录下。

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |  |
| --- | --- | --- |
| 01 | <resources xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2"> | |
| 02 | <tech-list> |

|  |  |  |
| --- | --- | --- |
| 03 | <tech>android.nfc.tech.IsoDep</tech> | |
| 04 | <tech>android.nfc.tech.NfcA</tech> |

|  |  |
| --- | --- |
| 05 | <tech>android.nfc.tech.NfcB</tech> |
| 06 | <tech>android.nfc.tech.NfcF</tech> |

|  |  |
| --- | --- |
| 07 | <tech>android.nfc.tech.NfcV</tech> |
| 08 | <tech>android.nfc.tech.Ndef</tech> |

|  |  |  |
| --- | --- | --- |
| 09 | <tech>android.nfc.tech.NdefFormatable</tech> | |
| 10 | <tech>android.nfc.tech.MifareClassic</tech> |

|  |  |  |
| --- | --- | --- |
| 11 | <tech>android.nfc.tech.MifareUltralight</tech> | |
| 12 | </tech-list> |

|  |  |
| --- | --- |
| 13 | </resources> |

你也可以指定多个tech-list集合。每个tech-list集合都被认为是独立的，而且只要任何一个tech-list集合是[getTechList()](http://developer.android.com/reference/android/nfc/Tag.html#getTechList())所返回的技术的子集，则你的activity都会被认为是匹配的。这为技术的匹配提供了AND和OR语义。下面的例子匹配可以支持NfcA和Ndef技术或可以支持NfcB和Ndef技术的标签：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |  |
| --- | --- | --- |
| 01 | <resources xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2"> | |
| 02 | <tech-list> |

|  |  |
| --- | --- |
| 03 | <tech>android.nfc.tech.NfcA</tech> |
| 04 | <tech>android.nfc.tech.Ndef</tech> |

|  |  |  |
| --- | --- | --- |
| 05 | </tech-list> | |
| 06 | </resources> |

|  |  |
| --- | --- |
| 07 |  |
| 08 | <resources xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2"> | |

|  |  |
| --- | --- |
| 09 | <tech-list> |
| 10 | <tech>android.nfc.tech.NfcB</tech> | |

|  |  |  |
| --- | --- | --- |
| 11 | <tech>android.nfc.tech.Ndef</tech> | |
| 12 | </tech-list> |

|  |  |
| --- | --- |
| 13 | </resources> |

在你的AndroidManifest.xml文件中，像下面的例子那样，在<activity>元素里的<meta-data>元素中指定你刚刚创建的资源文件：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |  |
| --- | --- | --- |
| 01 | <activity> | |
| 02 | ... |

|  |  |
| --- | --- |
| 03 | <intent-filter> |
| 04 | <action android:name="android.nfc.action.TECH\_DISCOVERED"/> | |

|  |  |  |
| --- | --- | --- |
| 05 | </intent-filter> | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | <meta-data android:name="android.nfc.action.TECH\_DISCOVERED" | |
| 08 | android:resource="@xml/nfc\_tech\_filter" /> |

|  |  |
| --- | --- |
| 09 | ... |
| 10 | </activity> | |

关于使用tag技术及[ACTION\_TECH\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TECH_DISCOVERED) intent的更多信息，请参考**高级NFC**文档中的[Working with Supported Tag Technologies](http://developer.android.com/guide/topics/connectivity/nfc/advanced-nfc.html#tag-tech)。

**ACTION\_TAG\_DISCOVERED**

为了过滤[ACTION\_TAG\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_TAG_DISCOVERED)，使用如下的intent filter：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 1 | <intent-filter> |
| 2 | <action android:name="android.nfc.action.TAG\_DISCOVERED"/> | |

|  |  |
| --- | --- |
| 3 | </intent-filter> |

**从intents中获取信息**

如果一个activity是由于一个NFC intent而被启动的，则你可以从intent中获取关于所扫描到的NFC标签的信息。依赖于所扫描到的标签，intents可能包含如下的extras：

* [EXTRA\_TAG](http://developer.android.com/reference/android/nfc/NfcAdapter.html#EXTRA_TAG) (必需的)：一个Tag对象表示被扫描的标签。
* [EXTRA\_NDEF\_MESSAGES](http://developer.android.com/reference/android/nfc/NfcAdapter.html#EXTRA_NDEF_MESSAGES) (optional): An array of NDEF messages parsed from the tag. This extra is mandatory on [intents.](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED)
* [EXTRA\_NDEF\_MESSAGES](http://developer.android.com/reference/android/nfc/NfcAdapter.html#EXTRA_NDEF_MESSAGES) (可选的)：由标签解析来的一个NDEF消息的数组。这个extra在intents上是强制的。
* [{@link android.nfc.NfcAdapter#EXTRA\_ID](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED) (可选的)：tag的底层的ID。

为了获取这些extras，可以去检查你的activity是否是由一个NFC intents启动起来的，以确保扫描到了一个标签，然后从intent中获取extras。下面的例子检查[ACTION\_NDEF\_DISCOVERED](http://developer.android.com/reference/android/nfc/NfcAdapter.html#ACTION_NDEF_DISCOVERED) intent，并从一个intent extra中获取NDEF消息。

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |  |
| --- | --- | --- |
| 01 | public void onResume() { | |
| 02 | super.onResume(); |

|  |  |  |
| --- | --- | --- |
| 03 | | ... |
| 04 | if (NfcAdapter.ACTION\_NDEF\_DISCOVERED.equals(getIntent().getAction())) { | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 05 | Parcelable[] rawMsgs = intent.getParcelableArrayExtra(NfcAdapter.EXTRA\_NDEF\_MESSAGES); | | |
| 06 | | if (rawMsgs != null) { |

|  |  |
| --- | --- |
| 07 | msgs = new NdefMessage[rawMsgs.length]; |
| 08 | for (int i = 0; i < rawMsgs.length; i++) { | |

|  |  |  |
| --- | --- | --- |
| 09 | msgs[i] = (NdefMessage) rawMsgs[i]; | |
| 10 | } |

|  |  |  |
| --- | --- | --- |
| 11 | } | |
| 12 | } |

|  |  |  |
| --- | --- | --- |
| 13 | //process the msgs array | |
| 14 | } |

作为一种选择，你可以从intent中获取一个[Tag](http://developer.android.com/reference/android/nfc/Tag.html)对象，其中将包含载荷，并使你可以迭代标签的拘束：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 1 | Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA\_TAG); |

**创建NDEF记录的通用类型**

这一节将描述如何创建NDEF记录的通用类型，而使你能够在向NFC标签中写入或通过Android Beam发送数据时得到帮助。从Android 4.0 (API level 14)开始,[createUri()](http://developer.android.com/reference/android/nfc/NdefRecord.html#createUri(android.net.Uri))方法就可以用来自动地创建URI记录。[createExternal()](http://developer.android.com/reference/android/nfc/NdefRecord.html#createExternal(java.lang.String, java.lang.String, byte[]))和[createMime()](http://developer.android.com/reference/android/nfc/NdefRecord.html#createMime(java.lang.String, byte[]))可以用来帮助你创建MIME和外部类型的NDEF记录。只要可能，就使用这些辅助方法以避免手动创建NDEF记录时可能出现的错误。

这一节也将描述如何创建对应于那样的记录的intent filter。所有的这些NDEF记录示例应该在你写入一个标签或beaming的NDEF消息的第一个NDEF记录里。

**TNF\_ABSOLUTE\_URI**

**注意：我们建议你使用**[**RTD\_URI**](http://developer.android.com/guide/topics/connectivity/nfc/nfc.html#well-known-uri)**类型，而不是**[**TNF\_ABSOLUTE\_URI**](http://developer.android.com/reference/android/nfc/NdefRecord.html#TNF_ABSOLUTE_URI)**，因为它更有效。**

你可以用下面的方法创建一个 [TNF\_ABSOLUTE\_URI](http://developer.android.com/reference/android/nfc/NdefRecord.html#TNF_ABSOLUTE_URI)  NDEF记录：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |  |
| --- | --- | --- |
| 1 | NdefRecord uriRecord = new NdefRecord( | |
| 2 | NdefRecord.TNF\_ABSOLUTE\_URI , |

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | "http://developer.android.com/index.html".getBytes(Charset.forName("US-ASCII")), | | |
| 4 | | new byte[0], new byte[0]); |

对应于前面的NDEF记录的intent filter看起来将像下面这样：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 1 | <intent-filter> |
| 2 | <action android:name="android.nfc.action.NDEF\_DISCOVERED" /> | |

|  |  |  |
| --- | --- | --- |
| 3 | <category android:name="android.intent.category.DEFAULT" /> | |
| 4 | <data android:scheme="http" |

|  |  |  |
| --- | --- | --- |
| 5 | android:host="developer.android.com" | |
| 6 | android:pathPrefix="/index.html" /> |

|  |  |
| --- | --- |
| 7 | </intent-filter> |

**TNF\_MIME\_MEDIA**

你可以用如下方法创建一个[TNF\_MIME\_MEDIA](http://developer.android.com/reference/android/nfc/NdefRecord.html#TNF_MIME_MEDIA) NDEF记录。使用[createMime()](http://developer.android.com/reference/android/nfc/NdefRecord.html#createMime(java.lang.String, byte[]))方法：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |  |
| --- | --- | --- |
| 1 | NdefRecord mimeRecord = NdefRecord.createMime("application/vnd.com.example.android.beam", | |
| 2 | | "Beam me up, Android".getBytes(Charset.forName("US-ASCII"))); |

手动地创建 [NdefRecord](http://developer.android.com/reference/android/nfc/NdefRecord.html) ：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |  |
| --- | --- | --- |
| 1 | NdefRecord mimeRecord = new NdefRecord( | |
| 2 | NdefRecord.TNF\_MIME\_MEDIA , |

|  |  |  |
| --- | --- | --- |
| 3 | "application/vnd.com.example.android.beam".getBytes(Charset.forName("US-ASCII")), | |
| 4 | | new byte[0], "Beam me up, Android!".getBytes(Charset.forName("US-ASCII"))); |

对应于前面的NDEF记录的intent filter看起来则将像下面这样：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 1 | <intent-filter> |
| 2 | <action android:name="android.nfc.action.NDEF\_DISCOVERED" /> | |

|  |  |
| --- | --- |
| 3 | <category android:name="android.intent.category.DEFAULT" /> |
| 4 | <data android:mimeType="application/vnd.com.example.android.beam" /> |

|  |  |
| --- | --- |
| 5 | </intent-filter> |

**TNF\_WELL\_KNOWN with RTD\_TEXT**

你可以用如下方法创建一个[TNF\_WELL\_KNOWN](http://developer.android.com/reference/android/nfc/NdefRecord.html#TNF_WELL_KNOWN) NDEF记录：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 01 | public NdefRecord createTextRecord(String payload, Locale locale, boolean encodeInUtf8) { |
| 02 | byte[] langBytes = locale.getLanguage().getBytes(Charset.forName("US-ASCII")); |

|  |  |  |
| --- | --- | --- |
| 03 | Charset utfEncoding = encodeInUtf8 ? Charset.forName("UTF-8") : Charset.forName("UTF-16"); | |
| 04 | byte[] textBytes = payload.getBytes(utfEncoding); |

|  |  |
| --- | --- |
| 05 | int utfBit = encodeInUtf8 ? 0 : (1 << 7); |
| 06 | char status = (char) (utfBit + langBytes.length); | |

|  |  |  |
| --- | --- | --- |
| 07 | byte[] data = new byte[1 + langBytes.length + textBytes.length]; | |
| 08 | data[0] = (byte) status; |

|  |  |
| --- | --- |
| 09 | System.arraycopy(langBytes, 0, data, 1, langBytes.length); |
| 10 | System.arraycopy(textBytes, 0, data, 1 + langBytes.length, textBytes.length); |

|  |  |  |
| --- | --- | --- |
| 11 | NdefRecord record = new NdefRecord(NdefRecord.TNF\_WELL\_KNOWN, | |
| 12 | NdefRecord.RTD\_TEXT, new byte[0], data); |

|  |  |  |
| --- | --- | --- |
| 13 | return record; | |
| 14 | } |

intent filter则将为：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 1 | <intent-filter> |
| 2 | <action android:name="android.nfc.action.NDEF\_DISCOVERED" /> | |

|  |  |  |
| --- | --- | --- |
| 3 | <category android:name="android.intent.category.DEFAULT" /> | |
| 4 | <data android:mimeType="text/plain" /> |

|  |  |
| --- | --- |
| 5 | </intent-filter> |

**TNF\_WELL\_KNOWN with RTD\_URI**

你可以用如下方法创建一个[TNF\_WELL\_KNOWN](http://developer.android.com/reference/android/nfc/NdefRecord.html#TNF_WELL_KNOWN) NDEF记录。使用[createUri(String)](http://developer.android.com/reference/android/nfc/NdefRecord.html#createUri(java.lang.String))方法：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 1 | NdefRecord rtdUriRecord1 = NdefRecord.createUri("http://example.com"); |

使用 [createUri(Uri)](http://developer.android.com/reference/android/nfc/NdefRecord.html#createUri(android.net.Uri)) 方法：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 1 | Uri uri = new Uri("http://example.com"); |
| 2 | NdefRecord rtdUriRecord2 = NdefRecord.createUri(uri); | |

手动地创建 [NdefRecord](http://developer.android.com/reference/android/nfc/NdefRecord.html)：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 1 | <SPAN></SPAN>byte[] uriField = "example.com".getBytes(Charset.forName("US-ASCII")); |
| 2 | byte[] payload = new byte[uriField.length + 1];              //add 1 for the URI Prefix |

|  |  |
| --- | --- |
| 3 | byte payload[0] = 0x01;                                      //prefixes http://www. to the URI |
| 4 | System.arraycopy(uriField, 0, payload, 1, uriField.length);  //appends URI to payload |

|  |  |
| --- | --- |
| 5 | NdefRecord rtdUriRecord = new NdefRecord( |
| 6 | NdefRecord.TNF\_WELL\_KNOWN, NdefRecord.RTD\_URI, new byte[0], payload); | |

对应于前面的NDEF记录的intent filter看起来则将像下面这样：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 1 | <intent-filter> |
| 2 | <action android:name="android.nfc.action.NDEF\_DISCOVERED" /> | |

|  |  |  |
| --- | --- | --- |
| 3 | <category android:name="android.intent.category.DEFAULT" /> | |
| 4 | <data android:scheme="http" |

|  |  |  |
| --- | --- | --- |
| 5 | android:host="example.com" | |
| 6 | android:pathPrefix="" /> |

|  |  |
| --- | --- |
| 7 | </intent-filter> |

**TNF\_EXTERNAL\_TYPE**

你可以用如下的方法创建一个[TNF\_EXTERNAL\_TYPE](http://developer.android.com/reference/android/nfc/NdefRecord.html#TNF_EXTERNAL_TYPE) NDEF记录。使用[createExternal()](http://developer.android.com/reference/android/nfc/NdefRecord.html#createExternal(java.lang.String, java.lang.String, byte[]))方法：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 1 | byte[] payload; //assign to your data |
| 2 | String domain = "com.example"; //usually your app's package name | |

|  |  |
| --- | --- |
| 3 | String type = "externalType"; |
| 4 | NdefRecord extRecord = NdefRecord.createExternal(domain, type, payload); | |

手动地创建   [NdefRecord](http://developer.android.com/reference/android/nfc/NdefRecord.html) ：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |  |
| --- | --- | --- |
| 1 | byte[] payload; | |
| 2 | ... |

|  |  |
| --- | --- |
| 3 | NdefRecord extRecord = new NdefRecord( |
| 4 | NdefRecord.TNF\_EXTERNAL\_TYPE, "com.example:externalType", new byte[0], payload); | |

对应于前面的NDEF记录的intent filter看起来则将像下面这样：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 1 | <intent-filter> |
| 2 | <action android:name="android.nfc.action.NDEF\_DISCOVERED" /> | |

|  |  |  |
| --- | --- | --- |
| 3 | <category android:name="android.intent.category.DEFAULT" /> | |
| 4 | <data android:scheme="vnd.android.nfc" |

|  |  |
| --- | --- |
| 5 | android:host="ext" |
| 6 | android:pathPrefix="/com.example:externalType"/> | |

|  |  |
| --- | --- |
| 7 | </intent-filter> |

为了更通用的NFC标 签部署以更好地同时支持Android和非Android设备，则使用 TNF\_EXTERNAL\_TYPE。

注意：[TNF\_EXTERNAL\_TYPE](http://developer.android.com/reference/android/nfc/NdefRecord.html#TNF_EXTERNAL_TYPE)的URNs具有一个标准的格式：

urn:nfc:ext:example.com:externalType，然而NFC Forum RTD规范中声明了urn:nfc:ext: URN的部分必须被从NDEF记录中省略。因而，你需要做的全部就是提供域(例子中的example.com)及类型(例子中的externalType)，并用冒号分开。当dispatchingTNF\_EXTERNAL\_TYPE是，Android将urn:nfc:ext:example.com:externalTypeURN转换为一个vnd.android.nfc://ext/example.com:externalType URI，即例子的intent filter中声明的那个。

**Android应用程序记录**

在Android 4.0(API level 14)中引入，在一个NFC标签被扫面到时你的activity被启动，一个Android Application Record (AAR)提供了一种更强的确定性。一个AAR在一个NDEF记录中嵌入了一个应用程序的包名。因为Android会在整个的NDEF消息中搜索AARs，你可以给你的NDEF消息的任何NDEF记录添加一个AAR。如果它找到一个AAR，它会根据AAR中的包名，启动那个应用程序。如果设备中没有那个应用程序，则Google Play会被启动来下载那个应用程序。

如果你想要阻止其他的应用程序过滤相同的intent，并可能处理你所部署的特定的标签，则AARs是很有用的。AARs只在应用程序层面支持，由于包名的限制，而不再Activity的层面，即intent过滤的层面。如果你想要在Activity的层面处理一个intent，则[使用intent filters](http://developer.android.com/guide/topics/connectivity/nfc/nfc.html#filtering-intents)。

如果一个标签包含一个AAR，则标签dispatch系统将以如下的方式dispatch：

1. 试着正常地使用一个intent filter启动一个Activity。如果那个Activity匹配intent同时也匹配AAR，则启动那个Activity。
2. 如果过滤了那个intent的Activity与AAR不匹配，如果有多个Activities可以处理那个intent，或者没有Activity处理那个intent，则启动AAR描述的那个应用程序。
3. 如果没有应用程序可以以那个AAR启动，则根据AAR进入Google Play来下载那个应用。

**注意：**你可以通过[前景dispatch系统](http://developer.android.com/guide/topics/connectivity/nfc/advanced-nfc.html#foreground-dispatch)覆写AARs和intent dispatch系统，这将使得当一个NFC标签被发现时，一个前景activity具有一定的优先权。通过这种方法，activity必须在前景来覆写AARs和intent dispatch系统。

如果你仍然想要过滤所扫描的不包含AAR的标签，你可以像通常那样声明intent filters。如果你的应用程序还对其他的不包含一个AAR的标签感兴趣时，这一点很有用。比如，也许你想要保证你的应用程序掌握所有的你部署的标签，及由第三方所部属的普通的标签。请记住，AARs只在Android 4.0设备或更新的版本中在可用，因而当部署标签时，你很可能想要使用AARs和MIME types/URI的结合来支持最广泛范围的设备。此外，当你部署NFC标签是，请考虑你想要如何写入你的NFC标签来启用对于大多数设备的支持(Android 和其他设备)。你可以通过定义一个相对唯一的MIME type或URI来使应用程序更易于区分来做到这一点。

Android提供了一个简单的API来创建一个AAR，[createApplicationRecord()](http://developer.android.com/reference/android/nfc/NdefRecord.html#createApplicationRecord(java.lang.String))。你要做的就只是把这个AAR嵌入到你的[NdefMessage](http://developer.android.com/reference/android/nfc/NdefMessage.html)的什么地方。你不要使用你的[NdefMessage](http://developer.android.com/reference/android/nfc/NdefMessage.html)的第一个记录，除非AAR是[NdefMessage](http://developer.android.com/reference/android/nfc/NdefMessage.html)中的唯一的记录。这是因为Android系统检查一个[NdefMessage](http://developer.android.com/reference/android/nfc/NdefMessage.html)的第一个记录来确定标签的MIME type或URI，他们会被用来为应用程序创建一个intent以过滤。下面的代码显示了你要如何创建一个AAR：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |  |
| --- | --- | --- |
| 1 | NdefMessage msg = new NdefMessage( | |
| 2 | new NdefRecord[] { |

|  |  |  |
| --- | --- | --- |
| 3 | | ..., |
| 4 | NdefRecord.createApplicationRecord("com.example.android.beam")} | | |

**Beaming NDEF 消息到其他的设备**

Android Beam允许在两个Android设备之间进行简单的点到点数据交换。想要beam数据到另外的设备的应用程序必须在前台，而接收数据的设备必须是没有锁屏的。当beaming设备与接收设备靠得足够近时则会进行联系，beaming设备显示“Touch to Beam” UI。然后用户选择是否要beam消息到接收设备。

**注意：**前台NDEF推送在API level 10是可用的，其提供了与Android Beam类似的功能。这些API已经被废弃了，但仍然是可用的，以支持较老的设备。请参考[enableForegroundNdefPush()](http://developer.android.com/reference/android/nfc/NfcAdapter.html#enableForegroundNdefPush(android.app.Activity, android.nfc.NdefMessage))来获取更多的信息。

你可以通过调用两个方法中的一个来为你的应用程序启用Android Beam：

* [setNdefPushMessage()](http://developer.android.com/reference/android/nfc/NfcAdapter.html#setNdefPushMessage(android.nfc.NdefMessage, android.app.Activity, android.app.Activity...))：接收一个[NdefMessage](http://developer.android.com/reference/android/nfc/NdefMessage.html)作为要beam的消息。当两个设备靠得足够近时自动地beam消息。
* [setNdefPushMessageCallback()](http://developer.android.com/reference/android/nfc/NfcAdapter.html#setNdefPushMessageCallback(android.nfc.NfcAdapter.CreateNdefMessageCallback, android.app.Activity, android.app.Activity...))：接收一个回调，其中包含有一个[createNdefMessage()](http://developer.android.com/reference/android/nfc/NfcAdapter.CreateNdefMessageCallback.html#createNdefMessage(android.nfc.NfcEvent))，当有设备在beam数据的范围时回调被调用。回调能够使你只在需要的时候才创建NDEF消息。

一个activity在某一时刻只能推送一条NDEF消息，因而如果两者都被设置了，则[setNdefPushMessageCallback()](http://developer.android.com/reference/android/nfc/NfcAdapter.html#setNdefPushMessageCallback(android.nfc.NfcAdapter.CreateNdefMessageCallback, android.app.Activity, android.app.Activity...))优先于[setNdefPushMessage()](http://developer.android.com/reference/android/nfc/NfcAdapter.html#setNdefPushMessage(android.nfc.NdefMessage, android.app.Activity, android.app.Activity...))。为了使用Android Beam，则必须满足下面的大体的guidelines：

* beaming数据的activity必须在前台。两台设备必须都是屏幕解锁的。
* 你必须把你要beaming的数据封装进一个[NdefMessage](http://developer.android.com/reference/android/nfc/NdefMessage.html)对象。
* 接收被beamed的数据的NFC设备必须支持com.android.npp NDEF推送协议或NFC Forum的SNEP (Simple NDEF Exchange Protocol)。在API level 9 (Android 2.3)到API level 13 (Android 3.2)上设备需要com.android.npp协议。在API level 14 (Android 4.0)及之后，则com.android.npp和SNEP都需要。

**注意：**如果你的activity启用了Android Beam，并且在前台，则标准的intent dispatch系统会被禁用。然而如果你的activity也启用了[前台dispatching](http://developer.android.com/guide/topics/connectivity/nfc/advanced-nfc.html#foreground-dispatch)，则它仍然可以扫描匹配了在前台dispatching中设置的intent filters的标签。

启用Android Beam：

1. 创建一个[NdefMessage](http://developer.android.com/reference/android/nfc/NdefMessage.html)，其中包含了你想要推送到另外的设备的[NdefRecord](http://developer.android.com/reference/android/nfc/NdefRecord.html)s。
2. 以一个[NdefMessage](http://developer.android.com/reference/android/nfc/NdefMessage.html)为参数调用[setNdefPushMessage()](http://developer.android.com/reference/android/nfc/NfcAdapter.html#setNdefPushMessage(android.nfc.NdefMessage, android.app.Activity, android.app.Activity...))，或在你的activity的onCreate()方法中传递一个[NfcAdapter.CreateNdefMessageCallback](http://developer.android.com/reference/android/nfc/NfcAdapter.CreateNdefMessageCallback.html)对象来调用[setNdefPushMessageCallback](http://developer.android.com/reference/android/nfc/NfcAdapter.html#setNdefPushMessageCallback(android.nfc.NfcAdapter.CreateNdefMessageCallback, android.app.Activity, android.app.Activity...))。这些方法至少需要一个你想用通过Android Beam启用的activity，同时其他的想要激活的activity则是可选的。通常，如果你的Activity只需要在所有时候都推送相同的NDEF消息，则正常地使用[setNdefPushMessage()](http://developer.android.com/reference/android/nfc/NfcAdapter.html#setNdefPushMessage(android.nfc.NdefMessage, android.app.Activity, android.app.Activity...))即可，当两个设备距离足够近时通信。当你的应用程序关心应用程序当前的上下文时，或者想要推送的一个NDEF消息依赖于你的应用程序中用户正在做的事时，则使用[setNdefPushMessageCallback](http://developer.android.com/reference/android/nfc/NfcAdapter.html#setNdefPushMessageCallback(android.nfc.NfcAdapter.CreateNdefMessageCallback, android.app.Activity, android.app.Activity...))。

下面的例子显示了一个简单的activity如何在activity的onCreate()方法中调用[NfcAdapter.CreateNdefMessageCallback](http://developer.android.com/reference/android/nfc/NfcAdapter.CreateNdefMessageCallback.html)。(参考[AndroidBeamDemo](http://developer.android.com/resources/samples/AndroidBeamDemo/index.html)来获取完整的示例)。这个例子也具有帮助创建MIME记录的方法：

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |  |
| --- | --- | --- |
| 01 | package com.example.android.beam; | |
| 02 |  |

|  |  |
| --- | --- |
| 03 | import android.app.Activity; |
| 04 | import android.content.Intent; | |

|  |  |  |
| --- | --- | --- |
| 05 | import android.nfc.NdefMessage; | |
| 06 | import android.nfc.NdefRecord; |

|  |  |
| --- | --- |
| 07 | import android.nfc.NfcAdapter; |
| 08 | import android.nfc.NfcAdapter.CreateNdefMessageCallback; | |

|  |  |  |
| --- | --- | --- |
| 09 | import android.nfc.NfcEvent; | |
| 10 | import android.os.Bundle; |

|  |  |
| --- | --- |
| 11 | import android.os.Parcelable; |
| 12 | import android.widget.TextView; | |

|  |  |
| --- | --- |
| 13 | import android.widget.Toast; |
| 14 | import java.nio.charset.Charset; | |

|  |  |
| --- | --- |
| 15 |  |
| 16 |  |

|  |  |  |
| --- | --- | --- |
| 17 | public class Beam extends Activity implements CreateNdefMessageCallback { | |
| 18 | NfcAdapter mNfcAdapter; |

|  |  |  |
| --- | --- | --- |
| 19 | TextView textView; | |
| 20 |  |

|  |  |
| --- | --- |
| 21 | @Override |
| 22 | public void onCreate(Bundle savedInstanceState) { | |

|  |  |  |
| --- | --- | --- |
| 23 | super.onCreate(savedInstanceState); | |
| 24 | setContentView(R.layout.main); |

|  |  |  |
| --- | --- | --- |
| 25 | TextView textView = (TextView) findViewById(R.id.textView); | |
| 26 | // Check for available NFC Adapter |

|  |  |  |
| --- | --- | --- |
| 27 | mNfcAdapter = NfcAdapter.getDefaultAdapter(this); | |
| 28 | if (mNfcAdapter == null) { |

|  |  |  |
| --- | --- | --- |
| 29 | Toast.makeText(this, "NFC is not available", Toast.LENGTH\_LONG).show(); | |
| 30 | finish(); |

|  |  |  |
| --- | --- | --- |
| 31 | return; | |
| 32 | } |

|  |  |
| --- | --- |
| 33 | // Register callback |
| 34 | mNfcAdapter.setNdefPushMessageCallback(this, this); | |

|  |  |  |
| --- | --- | --- |
| 35 | } | |
| 36 |  |

|  |  |
| --- | --- |
| 37 | @Override |
| 38 | public NdefMessage createNdefMessage(NfcEvent event) { | |

|  |  |
| --- | --- |
| 39 | String text = ("Beam me up, Android!\n\n" + |
| 40 | "Beam Time: " + System.currentTimeMillis()); | |

|  |  |
| --- | --- |
| 41 | NdefMessage msg = new NdefMessage( |
| 42 | new NdefRecord[] { createMime( | |

|  |  |  |  |
| --- | --- | --- | --- |
| 43 | "application/vnd.com.example.android.beam", text.getBytes()) | | |
| 44 | | /\*\* |

|  |  |
| --- | --- |
| 45 | \* The Android Application Record (AAR) is commented out. When a device |
| 46 | \* receives a push with an AAR in it, the application specified in the AAR |

|  |  |  |
| --- | --- | --- |
| 47 | \* is guaranteed to run. The AAR overrides the tag dispatch system. | |
| 48 | \* You can add it back in to guarantee that this |

|  |  |  |
| --- | --- | --- |
| 49 | \* activity starts when receiving a beamed message. For now, this code | |
| 50 | \* uses the tag dispatch system. |

|  |  |  |
| --- | --- | --- |
| 51 | | \*/ |
| 52 | //,NdefRecord.createApplicationRecord("com.example.android.beam") | | |

|  |  |
| --- | --- |
| 53 | }); |
| 54 | return msg; | |

|  |  |  |
| --- | --- | --- |
| 55 | } | |
| 56 |  |

|  |  |
| --- | --- |
| 57 | @Override |
| 58 | public void onResume() { | |

|  |  |
| --- | --- |
| 59 | super.onResume(); |
| 60 | // Check to see that the Activity started due to an Android Beam | |

|  |  |  |  |
| --- | --- | --- | --- |
| 61 | if (NfcAdapter.ACTION\_NDEF\_DISCOVERED.equals(getIntent().getAction())) { | | |
| 62 | | processIntent(getIntent()); |

|  |  |  |
| --- | --- | --- |
| 63 | } | |
| 64 | } |

|  |  |
| --- | --- |
| 65 |  |
| 66 | @Override | |

|  |  |
| --- | --- |
| 67 | public void onNewIntent(Intent intent) { |
| 68 | // onResume gets called after this to handle the intent | |

|  |  |  |
| --- | --- | --- |
| 69 | setIntent(intent); | |
| 70 | } |

|  |  |
| --- | --- |
| 71 |  |
| 72 | /\*\* | |

|  |  |  |
| --- | --- | --- |
| 73 | \* Parses the NDEF Message from the intent and prints to the TextView | |
| 74 | \*/ |

|  |  |
| --- | --- |
| 75 | void processIntent(Intent intent) { |
| 76 | textView = (TextView) findViewById(R.id.textView); | |

|  |  |  |
| --- | --- | --- |
| 77 | Parcelable[] rawMsgs = intent.getParcelableArrayExtra( | |
| 78 | NfcAdapter.EXTRA\_NDEF\_MESSAGES); |

|  |  |
| --- | --- |
| 79 | // only one message sent during the beam |
| 80 | NdefMessage msg = (NdefMessage) rawMsgs[0]; | |

|  |  |
| --- | --- |
| 81 | // record 0 contains the MIME type, record 1 is the AAR, if present |
| 82 | textView.setText(new String(msg.getRecords()[0].getPayload())); |

|  |  |  |
| --- | --- | --- |
| 83 | } | |
| 84 | } |

注意这段代码注释掉了一个AAR，你也可以移除它。如果你启用了AAR，则AAR中描述的应用程序总是接收Android Beam消息。如果应用程序不存在，则Google Play会被启动起来来下载那个应用。因此，对于Andriod 4.0或之后的版本，如果使用了AAR，则在技术上下面的intent filter不是必须的。

[view source](http://my.oschina.net/wolfcs/blog/178618#viewSource)



[print](http://my.oschina.net/wolfcs/blog/178618#printSource)[?](http://my.oschina.net/wolfcs/blog/178618#about)

|  |  |
| --- | --- |
| 1 | <intent-filter> |
| 2 | <action android:name="android.nfc.action.NDEF\_DISCOVERED"/> | |

|  |  |
| --- | --- |
| 3 | <category android:name="android.intent.category.DEFAULT"/> |
| 4 | <data android:mimeType="application/vnd.com.example.android.beam"/> | |

|  |  |
| --- | --- |
| 5 | </intent-filter> |

通过这个intent filter，com.example.android.beam应用现在可以在它扫描到一个NFC标签或接收到具有一个类型为com.example.android.beam的AAR的Android Beam时，或者当一个NDEF格式化的消息包含了一个类型application/vnd.com.example.android.beam的MIME记录时，被启动了。

即使AARs保证一个应用程序被启动或下载，仍然建议使用intent filters，因为他们使你启动你的应用中你选择的Activity，而不是总启动由一个AAR描述的包内的主Activity。同时，由于一些Android设备不支持AARs，你也应当在你的NDEF消息的第一个NDEF记录中嵌入标识信息，以防万一，并过滤它们。参考[创建NDEF记录的通用类型](http://developer.android.com/guide/topics/connectivity/nfc/nfc.html#creating-records)来获取更多如何创建记录的信息。

Done.