

How to Create Cross-Platform LaunchApp NFC Tags

This article explains how to create NFC tags that launch an app and work on other platforms too.

Introduction



For an app developer, one of the most important use cases of NFC is to let users easily discover and launch your app:

- **Discover:** attract new potential customers – they see a poster, tap it with their phone, and get to download your app.
- **Explore & Benefit:** your app might associate functionality with tapping tags – e.g., to unlock further levels in a game, to solve a puzzle in a treasure hunt game, or to get localized & personalized bus schedules from the exact bus stop where you tap.
- **Remember:** even if users already have your app installed, they can remember your app and its associated actions when they see a tag. A good example is a check-in with FourSquare, done by tapping in a restaurant.

These scenarios require three individual components:

- **Download new app from the store:** if the user doesn't already have the app installed, the device should directly point the user to the marketplace to let him download the app with minimum effort.
- **Launch already installed app:** in case the app is already installed, it should be immediately launched upon touching the tag
- **App-specific data:** if required, the app should be able to identify the tag, e.g., to know the name / ID of the restaurant to enable a direct check-in.

Unfortunately, this scenario isn't standardized by the [NFC Forum](#) – therefore, many companies come up with their own approaches. This article provides more background information on how to create a tag that is directly associated with your specific app and works on as many platforms as possible.

Alternatives: On the Windows platform, you also have the option to use a custom URI protocol – see the article [How to Launch Apps via Proximity APIs \(NFC\)](#) for more information. Custom URI schemes have the advantage that they're easier for cross-platform scenarios; the downside is that a custom URI scheme is not unique to your app; any other competitor can register its app for the same URI scheme, causing the user to find multiple apps in the store when they tap your tag.

Platform Approaches to Launching Apps

This section gives an overview of the different approaches on how to create a LaunchApp tag, followed by a section on how to combine the individual records to a single tag.

Windows Platform

Microsoft defined the [LaunchApp NDEF record](#) (Type: "windows.com/LaunchApp", Type Name Format: Absolute URI).

It contains launch parameters that will be passed to the application, and allows specifying any number of platforms and the respective app IDs (used to find the app in the store, as well as to launch it). A major limitation is that this records needs to be the first record in the NDEF message; otherwise, it will not be handled by the Windows OS. You can find more information in [this article](#).



While this format would allow to be extended with other platforms, no current other operating system is handling these records.

Symbian

Symbian allows registering an app for a custom record type (Type: custom – e.g., "nokia.com:nfcinteractor", Type Name Format: External RTD). Custom data for the application can be stored as the payload of the custom record.

This record doesn't enable linking to the store for downloading the app; the custom record is only recognized if an app registered for a specific custom type during its installation. Therefore, you would typically put two records on the tag: the first one being the custom record that is linked to your app, plus a second record a standard URL / Smart Poster record that links to your homepage or the store. If the phone already has the app installed, it will recognize the first, custom record and launch your app. If the app isn't installed, the phone will skip the (unknown) first record, proceed to the second record (the homepage / store URL) and open the browser / store client to let the user download the app.

More information on registering for custom record types: [App Autostart on NFC Tag Touch](#) (Nokia Developer Blogs).

Android

Starting with Android 4.x, Google introduced the [Android Application Record \(AAR\)](#) format (Type: "android.com:pkg", Type Name Format: External RTD). The payload of the record contains the name of the package, which is used by the operating system to launch the already installed app, or to search for it in the Play Store.

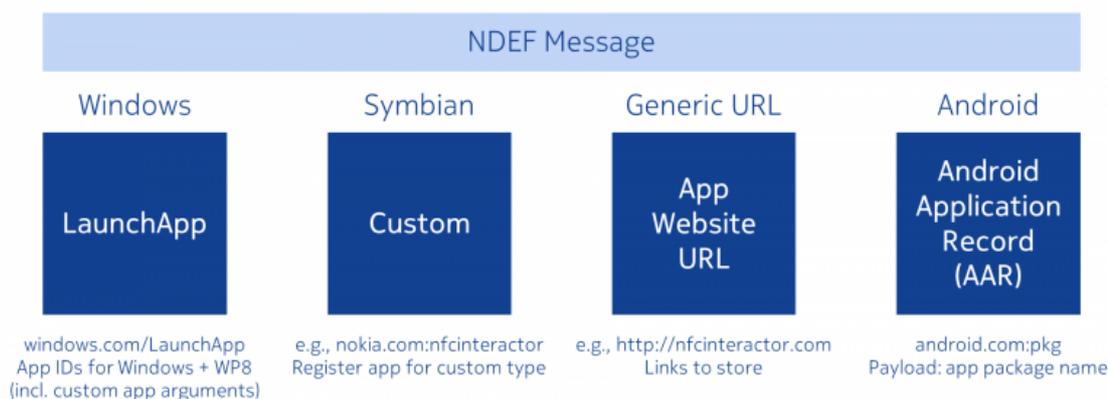
Google recommends to put this record last in the message, to enable compatibility for phones that do not understand this record type and start parsing from the beginning. So you would typically add a URL record that links to your homepage for phones that do not understand the AAR, and/or a custom record that contains data that will be read by your application.

Cross-Platform LaunchApp Tag

As described, various operating systems follow different approaches for directly launching apps from a tag. Fortunately, there is a way to combine all different records into a single tag, to create a cross-platform tag that directly launches your app on Windows, Symbian and Android, plus it contains a URI to link to your website for phones that do not understand one of the other schemes present on your tag.

For this tag to work, the specific order of the records in the message is of importance. By combining 4 records into a single message, the size requirements are considerable; in typical situations, the message will require at least a writable space on the tag of 250+ bytes.

This schematic visualizes how to store the records on the tag:



- **Windows LaunchApp record:** includes Windows and Windows Phone app IDs. Also include the custom app launch parameters here – if your app is already installed & running on another platform, it can read & parse the parameters out of any record; therefore, you only need to include the custom data once on the tag, saving important writable tag memory.
- **Custom record:** for Symbian. If the app is already installed & registered for the custom record on those platforms, the phone will automatically launch the app. Otherwise, the phone will continue parsing the record, until it finds a record they understand.
- **URI record:** used for Symbian if your app is not yet installed, or any other phone that doesn't know any of the other records and skipped the first two records that it didn't understand. Points to your website, where you can provide download links for any kind of platform. Can't be used to directly launch your application.
- **Android Application Record:** place this record at the end, as Android will also find it there and it won't interfere with the other records present on the tag.

How to write such a tag? Windows Proximity APIs can write a LaunchApp tag, but don't support including this record into a NDEF message that additionally contains other records.

The [NDEF library for Proximity APIs](#) includes a class that can create a LaunchApp record and embed it into a bigger NDEF message. Therefore, use the library in order to create a cross-platform LaunchApp tag – it also includes support for creating custom, URI and Android Application Records.

References

For an overview of the Proximity APIs and how to read and write tags, read the article [How to Acquire and Publish Content from / to NFC Tags and Proximity Peers](#).

To find out more about Proximity APIs, check out the [API documentation](#). More information about how LaunchApp tags are actually formatted as the raw tag contents, is contained in the [specification for Proximity driver implementers](#).

If you'd like to create an NDEF message & LaunchApp record with the Proximity APIs without having to read all the [NDEF specifications](#), you can simply use the [NDEF Library for Proximity APIs](#), which include ready-made convenience classes for creating all kinds of records and messages.

