How to launch apps available in the store using NFC LaunchApp tag

This article explains how to launch any given application available from Windows Phone Store via NFC tag by using `LaunchApp` protocol provided in `Windows.Networking.Proximity.ProximityDevice` from Windows Phone 8.

Introduction

Microsoft introduced the proximity framework with Windows Phone 8 SDK. The proximity framework provides the developers access to features enabled by Near field communication (NFC).

One of the essential features that NFC provides is launch an application when the end user tap his/her mobile device to an external tag. This feature is fundamental to many potential use cases. For example, in the car driving scenario, the end user puts the mobile device to a phone holder in the car. If the phone holder has a NFC tag on it, then once the end user fixes his mobile device to the phone holder, the mobile device will automatically launch navigation software e.g. Nokia Drive/Maps. Another user case could be for sleeping use case. The end user puts the mobile devices to his bed desk. If the desk has a NFC tag on it, the mobile device could launch an application automatically according to user's preference, such as opening a music player to play audio book, or simply switch to silent mode. Similar use cases could be for scenarios like shopping, hotel check-in, travel guide, restaurant review, etc. Launching a specific application on mobile device by tapping a tag is an essential enabler for those innovative use cases. This article will explain in details how to implement such feature on Windows Phone 8 devices.

Although Microsoft has provided some documentation about LaunchApp protocol [1][2], the quality of current version of the documentation still has room to improve. For example, examples and explanations are only for windows 8 and not yet covering windows phone 8. Therefore, it is better to have an article to explain those details for windows phone 8. We will firstly introduce what is LaunchApp protocol, then explain how to form the LaunchApp protocol for a specific app on windows phone 8, and thirdly we will describe how to write the LaunchApp content into a NFC tag, and at the end we will illustrate how the device behaves when tapping the LaunchApp tag in different scenarios.

What is the LaunchApp protocol

The definition of LaunchApp protocol can be found from the API documentation for ProximityDevice.PublishBinaryMessage. The protocol is designed to provide support for multiple platform using the same tag content, such as both for windows 8 and Windows Phone 8.

According to that, the LaunchApp content should follow the syntax below and you must specify at least one app platform and app name.

```
<launch arguments>[tab]<app platform 1>[tab]<app name 1>...[tab]<app platform>
```

An example is given for windows 8: "user=default\tWindows\tExample.Proximity.JS_8wekyb3d8bbwe!Proximity.App", where the <launch arguments> is "user=default", the <app platform> is "Windows", and the app name is in the format of <package family name>!<app Id>, which is "Example.Proximity.JS_8wekyb3d8bbwe!Proximity.App" in the example.

For windows phone 8, the syntax of LaunchApp still applies. The <launch arguments> is "user=default" and the <app platform> is "WindowsPhone". However, the format of <app name> is different than windows 8. The <app name> is {<app id>}, where app id is in the format of <8digi hex>-<4digi hex>-<4digi hex>-<12digi hex>. For example Nokia MixRadio's app id is f5874252-1f04-4c3f-a335-4fa3b7b85329. Then the content of LaunchApp for Nokia MixRadio should be "user=default\tWindowsPhone\t{f5874252-1f04-4c3f-a335-4fa3b7b85329}". Please note that the **minimum size** of the LaunchApp tag for windows phone 8 is **128 bytes**.

How to form LaunchApp content for a specific application on Windows Phone 8

As explained in the previous section, the key to form the LaunchApp content for Windows Phone 8 is to obtain the app id of the application. Depending on the scenarios, there are different ways to get the app id of the desired application

How to get the app id of the current running application

If you want to get the app id of the current running application in C# code, you can simply use the Windows.ApplicationModel.Store.CurrentApp.appId to do so, which documentation can be found from here

(C) Copyright Nokia 2014. All rights reserved.

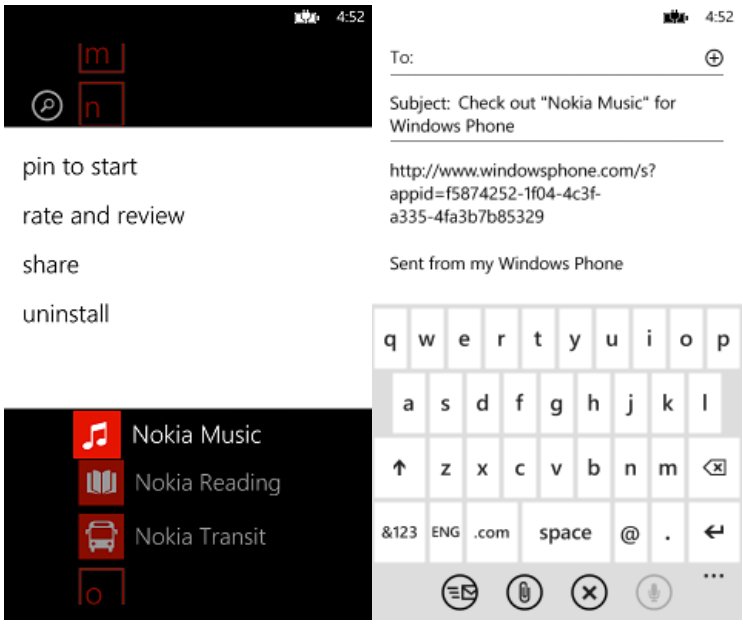
```
var appId = Windows.ApplicationModel.Store.CurrentApp.AppId;
```

Or you can use the System.Xml.Linq.XDocument to pass WMAppManifest.xml to get product ID (same as app id once the app is published)

```
Guid appId = Guid.Empty;
var productId =
XDocument.Load("WMAppManifest.xml").Root.Element("App").Attribute("ProductID");
if (productId != null && !string.IsNullOrEmpty(productId.Value))
{
    Guid.TryParse(productId.Value, out appId);
}
```

How to get the app id of an application installed on the device

If you want to get the app id of an application installed on the device, you can go to application list and long press the app you want, click "share", choose any email account, then copy the string after "?appid=" from the URL.



How to get the app id of an application available in the store

If you want to get the app id of an application available in the store, you can simply visit http://www.windowsphone.com/en-us/store or its localized sites and search for the app. Once you find the link of the app, then pay attention to the URL of the app, the app id is at the end of the URL. For example, the Nokia MixRadio's URL is http://www.windowsphone.com/en-us/store/app/nokia-music/f5874252-1f04-4c3f-a335-4fa3b7b85329, where its app id is f5874252-1f04-4c3f-a335-4fa3b7b85329

How to write LaunchApp content into a NFC tag

To write the LaunchApp content into a NFC tag, you need to do the following:

- Get an instance of Windows.Networking.Proximity.ProximityDevice

```
var device = Windows.Networking.Proximity.ProximityDevice.GetDefault();
```

- Check if the target tag has enough payload for the LaunchApp content. If the LaunchApp has only windows phone 8 entry, then the required payload size is 128 bytes. WriteableTag protocol is defined in ProximityDevice.PublishBinaryMessage

```
var writableTagId = device.SubscribeForMessage("WriteableTag", checkWritableTagSize);
```


(C) Copyright Nokia 2014. All rights reserved.

- Publish the LaunchApp content using Windows.Networking.Proximity.ProximityDevice.PublishBinaryMessage with "LaunchApp:WriteTag" as the type of message

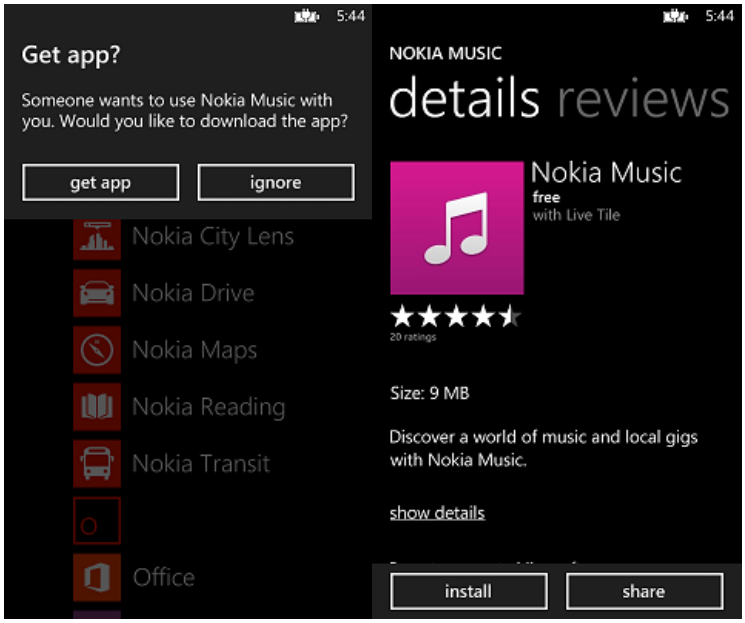
```
msgId = device.PublishBinaryMessage("LaunchApp:WriteTag",
GetBufferFromString(launchAppString), messageTransmittedHandler);
```

How the windows phone device behaves when tapping a tag with LaunchApp content

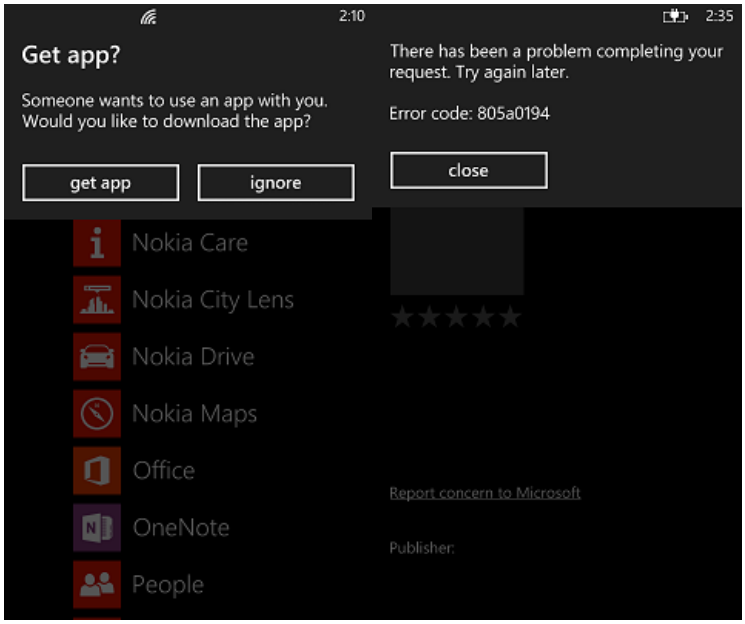
When the end user taps his/her windows phone 8 device to the tap with LaunchApp tag, the device behaves differently depending on whether the app has been already installed on the device or not. Here we use Nokia MixRadio as an example app for LaunchApp tag with windows phone 8 device.

How the windows phone 8 behaves if the app the tag linked to is not installed

If the application is not yet installed on the windows phone 8 device when tapping the tag, then the device will ask if the end user wants to install the app. If the end user selects "get app", then the device will launch store to install the app.



If the application is not installed and is not yet published to store, then the device will display general info asking to install the app. Then if you try to install the app by clicking "get app", the system will show an error message as the app cannot be found from the store.

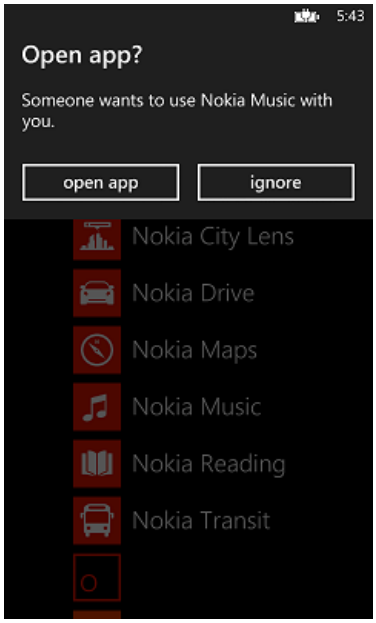


How the windows phone 8 device behaves if the app the tag linked to is

(C) Copyright Nokia 2014. All rights reserved.

installed

If the application is already installed on the windows phone 8 device, then the device will ask if the end user wants to launch the app or not.



Version Hint

Windows Phone: [[Category:Windows Phone]]
[[Category:Windows Phone 7.5]]
[[Category:Windows Phone 8]]

Nokia Asha: [[Category:Nokia Asha]]
[[Category:Nokia Asha Platform 1.0]]

Series 40: [[Category:Series 40]]
[[Category:Series 40 1st Edition]] [[Category:Series 40 2nd Edition]]
[[Category:Series 40 3rd Edition (initial release)]] [[Category:Series 40 3rd Edition FP1]] [[Category:Series 40 3rd Edition FP2]]
[[Category:Series 40 5th Edition (initial release)]] [[Category:Series 40 5th Edition FP1]]
[[Category:Series 40 6th Edition (initial release)]] [[Category:Series 40 6th Edition FP1]] [[Category:Series 40 Developer Platform 1.0]] [[Category:Series 40 Developer Platform 1.1]] [[Category:Series 40 Developer Platform 2.0]]

Symbian: [[Category:Symbian]]
[[Category:S60 1st Edition]] [[Category:S60 2nd Edition (initial release)]] [[Category:S60 2nd Edition FP1]] [[Category:S60 2nd Edition FP2]] [[Category:S60 2nd Edition FP3]]
[[Category:S60 3rd Edition (initial release)]] [[Category:S60 3rd Edition FP1]] [[Category:S60 3rd Edition FP2]]
[[Category:S60 5th Edition]]
[[Category:Symbian^3]] [[Category:Symbian Anna]] [[Category:Nokia Belle]]

(C) Copyright Nokia 2014. All rights reserved.

 (C) Copyright Nokia 2014. All rights reserved.