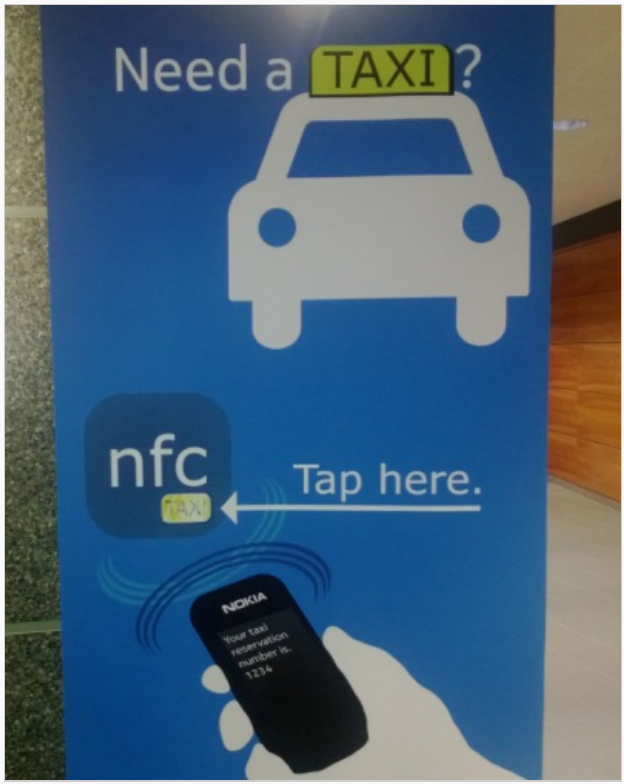
Link2SMS - a protocol handling example

This article explains how to create a protocol handler for Windows Phone 8, using the particular example of handling *sms:* links (as found, for example, in NFC tags).

Introduction

One popular use case for NFC is the implementation of the "tap to call a taxi" scenario. A predefined phone number and message is encoded in an NFC tag attached to a poster that advertises the service. All a user has to do is tap the poster and then confirm that they want to send the message - one tap *of* the phone and one tap *on* the phone, and your taxi order is placed.



Tap to call a taxi poster.

The problem with the above use case is that it relies on the *sms:* protocol being handled on the device and Windows Phone does not support it at time of writing. Luckily on Windows Phone 8, third-party developers can add their own protocol handler applications and this article will show how easy this *sms:* handling gap can be filled.

Protocol specification

We start by creating a simple Windows Phone application. We then add the following protocol specification to **WMAppManifest.xml** (note that the `<Extensions>` must follow the `<Tokens>`):

```
...
</Tokens>
<Extensions>
  <Protocol Name="sms" NavUriFragment="encodedLaunchUri=%s" TaskID="_default" />
</Extensions>
<ScreenResolutions>
...
```

Now the application will be recognized as a `sms:` protocol handler and will be called each time a link using this protocol is activated, be it from an other app, from a browser or from an NFC tag.

Tip: If you're writing your own protocol handler, the only customization required is changing the `Protocol Name` value to your own protocol text.

(C) Copyright Nokia 2014. All rights reserved.

Handling the sms protocol URI

When our application is started, its navigation context will contain the fragment `encodedLaunchUri` with the actual link URL encoded as its value.

Since the app would probably want to handle this special start mode in a different way than when the user starts it from the menu or tile, we implement an URIMapper as follows:

```
namespace LinkToSMS
{

        class LinkToSMSUriMapper : UriMapperBase
        {
            private string tempUri;

            public override Uri MapUri(Uri uri)
            {
                tempUri = System.Net.HttpUtility.UrlDecode(uri.ToString());

                // URI association launch for sms.
                if (tempUri.Contains("sms:"))
                {
                    int phoneNoIndex = tempUri.IndexOf("sms:") + 4;

                    // url encode back the payload string before passing it to the
navigation framework
                    string payload =
System.Net.HttpUtility.UrlEncode(tempUri.Substring(phoneNoIndex));

                    // Redirect this text message send request to the view specialized
in handling such request
                    return new Uri("/MainPage.xaml?SMS=" + payload, UriKind.Relative);
                }

                // Otherwise perform normal launch.
                return uri;
            }
        }


        public App()
        {
            // Global handler for uncaught exceptions.
            UnhandledException += Application_UnhandledException;

            // Standard XAML initialization
            InitializeComponent();

            // Phone-specific initialization
            InitializePhoneApplication();

            // Assign the URI-mapper class to the application frame.
            RootFrame.UriMapper = new LinkToSMSUriMapper();
            ...
```


(C) Copyright Nokia 2014. All rights reserved.

In the `MapUri()` method the string received is first URL decoded then parsed to identify the protocol. If the SMS protocol is found its parameters are passed forward to one of the XAML pages, in this case the **MainPage.xaml**, for handling. Note that the parameters passed on to the page are again URL encoded so that no information is lost during the automatic parsing done in the navigation context.

In **MainPage.xaml.cs** the SMS can be handled by simply calling the SmsComposeTask using the phone number and message retrieved from the link:

```
        // Constructor
        public MainPage()
        {
            InitializeComponent();

            // Sample code to localize the ApplicationBar
            //BuildLocalizedApplicationBar();

            Loaded += MainPage_Loaded;
        }

        void MainPage_Loaded(object sender, RoutedEventArgs e)
        {
            if (this.NavigationContext.QueryString.ContainsKey("SMS"))
            {
                string token = this.NavigationContext.QueryString["SMS"];

                int bodyPos = token.IndexOf("?body=");
                string phoneNo = token.Substring(0, bodyPos);
                string body = token.Substring(bodyPos + 6);

                SmsComposeTask smsComposeTask = new SmsComposeTask();
                smsComposeTask.To = phoneNo;
                smsComposeTask.Body = body;
                smsComposeTask.Show();
            }

            //else
            // The view was entered without an SMS to handle, so likely the app was
started from the app list.
        }
```

As simple as that. Now you can use the NFC Interactor to write something like "sms:+5551234?body=This is a sample text message." on an NFC tag then tap it with your phone to have the message compose and sent away after user's confirmation. In our Taxi example, this sample message might include the poster location (or the location might be retrieved from the device).

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.