Como adquirir e publicar conteúdo de/para Tags NFC e pares próximos

Este artigo explica como usar as APIs de proximidade do Windows (Phone) 8 para interagir com tags e dispositivos com Near Field Communication (NFC).

Introdução

A framework de Proximidade na plataforma do Windows 8 é sobre a activação de acções do tipo "Tocar e fazer" . Um dos principais cenários de Near Field Communication (NFC), envolve a aquisição de conteúdo de tags NFC e outros dispositivos dos peers. Sinceramente, você também pode publicar mensagens para outros colegas, ou mesmo escrever tags NFC.

Este artigo dá uma visão geral das possibilidades, bem como trechos de como começar com o desenvolvimento de programas de proximidade. Ele é baseado nos slides do Windows 8 Platform NFC Development lecture, mas acrescenta todas as informações de base necessárias.

Aquisição de conteúdo das Tags NFC e Peers

O conteúdo pode ser adquirido numa infinidade de formas para diferentes cenários. O caso mais simples possível é uma tag que contém um link para o site da empresa. Um tipo mais envolvente é oferecer a um visitante mais informações pelo toque, por exemplo, sobre uma obra de arte em um museu ou comprar um bilhete de concerto tocando no cartaz. Para uma solução de publicidade de maior impacto, também é possível criar extensões de aplicações personalizadas que podem ser activadas pelo toque, por exemplo, para desbloquear um item de bónus num jogo, tocando a tag contida numa revista, ou um brinquedo de pelúcia em forma de um ave adicional.

Vamos dar uma olhada em como adquirir conteúdo do ponto de vista do programador. É necessário, principalmente, para se lembrar de duas classes: `ProximityDevice` e `ProximityMessage`. O primeiro vai criar a conexão com o hardware, detecta dispositivos de extensão próxima e permite que você publique e se subscreva a mensagens. Essas mensagens são então entregues pela classe `ProximityMessage`. No entanto antes que você possa usar essas classes na sua aplicação, você precisa de activar o recurso de "proximidade" no seu arquivo de manifesto.

Subscrever-se a URIs

O caso de uso mais simples para a aquisição de conteúdo é subscrever-se a URIs. Primeiro, activar o dispositivo de proximidade - geralmente, é bom para simplesmente obter o padrão. Estamos a usar aqui a variável cujo nome é `_device` - crie a variável de instância de acordo com a sua classe. Em seguida, escolha para se subscrever para o tipo de mensagem `WindowsUri` e especifique o seu método de manipulador de retorno de chamada. O valor de retorno é uma identificação numérica de inscrição, que pode ser usada para cancelar a subscrição novamente, numa fase posterior, bem como para descobrir qual a assinatura da mensagem que vem no caso de você usar várias assinaturas com o mesmo método do manipulador.

```
_device = ProximityDevice.GetDefault();
_subscribedMessageId = _device.SubscribeForMessage("WindowsUri",
MessageReceivedHandler);
```

Através do segundo parâmetro no método do manipulador (mensagem),o seu código vai receber a mensagem que recebeu - no caso do tipo `WindowsUri`, é diretamente a URI, não importa como ele é realmente armazenado na tag NFC. Você só precisa converter os dados codificados em Unicode da mensagem de proximidade para uma string / URI:

```
private void MessageReceivedHandler(ProximityDevice sender, ProximityMessage message)
{
    var msgArray = message.Data.ToArray();
    var url = Encoding.Unicode.GetString(msgArray, 0, msgArray.Length);
    Debug.WriteLine("URI: " + url);
}
```

Ao ler uma tag NFC, ela provavelmente irá conter uma mensagem NDEF padronizada (mais sobre isso mais tarde), e não um

(C) Copyright Nokia 2014. All rights reserved.

tipo `WindowsUri`. Felizmente, aquando da subscrição a `WindowsUri`, a sua aplicação vai ficar informada sobre a URI contida na tag, não importa se ela é formatada como um registo NDEF URI ou um Smart Poster. Então você pode preparar cada tag padrão com o código acima e não precisa de se preocupar com o seu formato. No entanto, você vai perder informação adicional armazenada no Smart Posters, tal como o texto do título.

Para cancelar a subscrição no caso de você não precisar mais da assinatura de mensagem, basta chamar o método a seguir com o URI que você tem na subscrição (que também a temos armazenado numa variável de membro):

```
_device.StopSubscribingForMessage(_subscribedMessageId);
```

Espalhe a palavra - Publicar & Escrever

Em vez de apenas receber conteúdo, a aplicação também pode querer informar outros peers ou escrever informações para tags. Publicando para outros peers também pode ser usado para empurrar conteúdo para outra instância da sua aplicação em execução noutro dispositivo, através da utilização de um esquema URI personalizada para a qual a sua aplicação é registada (um esquema URI personalizado é, por exemplo skype:username?call em vez de http :/ / www.skype.com/).

Publicando para peers

O envio de uma URI para peers é ainda mais fácil do que subscrever para mensagens, e pode ser feito com apenas uma linha de código (depois de adquirir o dispositivo de proximidade padrão, tal como se mostrou na secção anterior):

```
_publishingMessageId = _device.PublishUriMessage(new Uri("http://nfcinteractor.com"));
```

Note que esta chamada só vai enviar a URI para os dispositivos pares, mas não vai escrever o URI para tags.A comunicação com outros dispositivos é encapsulado no protocolo SNEP padronizada, assim muitos outros dispositivos e telemóveis com NFC são capazes de receber e compreender a mensagem; não tem que ser baseados no Windows.

Especificamente, o seu dispositivo Windows vai realmente enviar duas mensagens NDEF, cada uma contendo um registo:

1. Um registo NDEF URI padronizado. Este pode ser lido por qualquer outro dispositivo e contém apenas o URI especificado no seu código.
2. Um registo NDEF do tipo `windows.com/UriSchemeToAppHint` (tipo do formato do name: Absolute URI), que contém o sistema, a plataforma e a ID da aplicação. Este tipo é, obviamente, específico para o Windows, e ajuda os outros dispositivos do Windows encontrarem uma aplicação que pode lidar com um esquema URI personalizado. Leia mais sobre URIs personalizados e aplicações de lançamento.

Escrever Tags

Se você preferir escrever a mensagem para uma tag, você precisa de chamar um método ligeiramente diferente. A maioria do trabalho, antes desta chamada, é para converter a string URI para um array de bytes no formato little endian com codificação UTF-16.

```
var dataWriter = new Windows.Storage.Streams.DataWriter {
    UnicodeEncoding = Windows.Storage.Streams.UnicodeEncoding.Utf16LE };
dataWriter.WriteString("http://nfcinteractor.com [Este link é para o exterior da TechNet
Wiki. Vai abrir numa nova janela.] ");
var dataBuffer = dataWriter.DetachBuffer();
_device.PublishBinaryMessage("WindowsUri:WriteTag", dataBuffer);
```

Em contraste com o método `PublishUriMessage()`, que envia duas mensagens para os peers, a `PublishBinaryMessage()` com o tipo `WindowsUri:WriteTag` escreve apenas uma única mensagem URI NDEF padronizada para a tag, e não adiciona uma mensagem específica do Windows. Tenha em mente que o tamanho de uma tag NFC típica é limitada a poucos bytes, por isso cada carácter conta e uma segunda mensagem pode até não caber na tag.

(E) Copyright Nokia 2014. All rights reserved.

Tamanho da Tag

Se você quiser descobrir o tamanho da tag gravável antes de tentar armazenar as suas informações nela, você pode subscrever-se a um tipo especial:`WriteableTag`.

```
_device.SubscribeForMessage("WriteableTag", MessageReceivedHandler);
```

Sempre que o `ProximityDevice` vai descobrir uma tag que é adequada para escrever informações, o seu manipulador será chamado, e o `ProximityMessage` contém o tamanho da tag gravável em bytes:

```
var tagSize = BitConverter.ToInt32(message.Data.ToArray(), 0);
Debug.WriteLine("Writeable tag size: " + tagSize);
```

Enquanto estamos no assunto: você também pode encontrar algumas informações sobre o hardware de proximidade - mais útil será a taxa de transferência; as especificações do driver impor um mínimo de 16kB/sec. Disponível também é o tamanho máximo de uma mensagem que você pode publicar (>= 10kB).

```
var bps = _device.BitsPerSecond; // >= 16kB/s
var mmb = _device.MaxMessageBytes; // >= 10kB
```

Manipular o NDEF

Nós já vimos mensagens NDEF e registos, então o que é NDEF? O Fórum NFC padronizou o NFC Data Format Exchange, que define essencialmente como o conteúdo tem que ser formatado no mundo da NFC.

Cada registo NDEF tem um cabeçalho, que contém - entre outros bits de informação - o tipo do registo. Por exemplo, o registo URI tem o nome de "U". A definição registo URI, em seguida, também especifica como a carga do registo tem que ser, por exemplo, a própria URI precisa de ser armazenada em formato UTF-8. Para poupar espaço precioso numa tag, os protocolos comuns URI e prefixos são encurtados. Isto permite que codifica o longo e caro "http://xn--ivg.net/" num único byte no cabeçalho.

O Smart Poster pode, então, ser visto como um meta registo, que envolve uma URI com alguns outros bits de informação, mais comum sendo um ou mais textos do título, em várias línguas.

Uma mensagem NDEF pode então ser pensada como uma caixa que contém vários registos NDEF individuais, para também deixar o leitor saber quando parar a leitura do tag.

Subscrevendo-se para tipo WindowsUri vai poder ler os registos Uri e Smart Posters - e dar-lhe apenas o URI. E se você gostaria de ler as informações restantes de um Smart Poster, ou se subscrever num registo NDEF totalmente diferente? Esta é a chamada da assinatura correspondente:

```
_subscribedMessageId = _device.SubscribeForMessage("NDEF", MessageReceivedHandler);
```

O seu manipulador receberá call-backs para todas as mensagens NDEF descobertas, não importa qual o tipo. O parâmetro irá então conter a carga de matéria-prima da mensagem. Cabe a você descobrir o conteúdo da mensagem, por exemplo, se é um URI simples, você necessita de converter o byte encurtado de volta para o "http://xn--ivg.net/" completo, conforme definido nas especificações do Forum NFC. Isso é um monte de trabalho e exige muito código de baixo nível, trabalhar com bits e bytes. Mesmo a análise de um registo URI simples já é mais do que uma página de código-fonte.

Biblioteca NDEF para APIs de Proximidade

Para o ajudar a não desperdiçar o seu tempo com a implementação das mensagem NDEF analisadas de acordo com as especificação técnicas de documentos brancas, você pode baixar e usar o código-fonte aberto da Biblioteca NDEF.

Escrito em C#, pode converter o array de bytes crus que você adquire das APIs de proximidade numa mensagem NDEF. Classes especializadas para cada tipo de registo permitem que você simplesmente extraia as informações presentes no registo, sem a necessidade de saber exactamente o que está codificado nos dados crus.

(C) Copyright Nokia 2014. All rights reserved.

Além disso, a biblioteca permite criar mensagens NDEF, que podem facilmente ser gravadas numa tag ou publicadas noutro ponto.

A biblioteca é publicada sob a licença LGPL, que é uma das licenças de código aberto mais permissivas. Ela permite o uso da biblioteca em código fechado, aplicações comerciais também. A biblioteca é parcialmente baseada no módulo de conectividade da biblioteca Qt Mobility, que também está licenciada sob a LGPL.

Para ler e analisar um Smart Poster da matriz de bytes crus que você obtém do callback handler das APIs de proximidade, o seguinte trecho de código é suficiente. Claro, você também pode extrair mais informações do `NdefSpRecord`, como títulos adicionais, o tamanho do conteúdo vinculado, etc. - dependendo do que é realmente definido na tag.

```
// Analisar array de bytes cru para mensagem NDEF
var msg = NdefMessage.FromByteArray(rawMsg);

foreach (NdefRecord record in msg)
{
    // Passar por cada registo, verifica se é um Smart Poster
    if (record.CheckSpecializedType() == typeof (NdefSpRecord))
    {
        // Converter e extrair informações sobre Smart Poster
        var spRecord = new NdefSpRecord(record);
        Debug.WriteLine("URI: " + spRecord.Uri);
        Debug.WriteLine("Titles: " + spRecord.TitleCount());
        Debug.WriteLine("1. Title: " + spRecord.Titles[0]);
        Debug.WriteLine("Action set: " + spRecord.ActionInUse());
    }
}
```

Se, por outro lado, você gostaria de escrever um Smart Poster para uma tag, o seguinte trecho de código cria primeiro o registo correspondente e adiciona um único título a ele. Então, ele encapsula o registo numa mensagem NDEF e converte-o para um array de bytes. Este é então publicado com as APIs de proximidade com o tipo `NDEF:WriteTag`, que permite escrever mensagens NDEF de baixo nível.

Note: Tenha cuidado com as matrizes de bytes de baixo nivel - se usa mensagem NDEF contém erros, as APIs podem recusar de os escrever na tag, e não necessariamente informá-lo sobre o assunto. Portanto, é recomendável usar a biblioteca NDEF, que sempre cria mensagens padrão compatíveis.

```
// Inicializar registo Smart Poster com URI, Ação + 1 Title
var spRecord = new NdefSpRecord {
    Uri = "http://nfcinteractor.com",
    NfcAction = NdefSpActRecord.NfcActionType.DoAction };

spRecord.AddTitle(new NdefTextRecord {
    Text = "Nokia", LanguageCode = "en" });

// Adicionar registo para a mensagem NDEF
var msg = new NdefMessage { spRecord };

// Publicar mensagem NDEF para uma tag / telefone
// AsBuffer(): add -> using System.Runtime.InteropServices.WindowsRuntime;
_device.PublishBinaryMessage("NDEF:WriteTag", msg.ToByteArray().AsBuffer());
```

Referências

Este artigo baseia-se na apresentação correspondente por Andreas Jakl e fornece explicações de base da tecnologia NFC que você não vai perceber apenas vendo os slides. Os slides podem ser vistos e descarregados aqui: http://slidesha.re/UJAefK

(3) Copyright Nokia 2014. All rights reserved.

Para saber mais sobre APIs de proximidade, confira a documentação da API.

Para alguns casos, também é útil a especificação do driver de NFC - você não o precisa de saber, mas poderia ajudar com algum conhecimento de base, se você quer ir para níveis mais baixos ou precisar detalhes sobre codificações ou limitações.

A Biblioteca NDEF para APIs de proximidade pode ser descarregada gratuitamente a partir do CodePlex: https://ndef.codeplex.com/

(C) Copyright Nokia 2014. All rights reserved.