吉原 泰介

株式会社ミガロ.

RAD事業部 技術支援課 顧客サポート

[Delphi/400 XE5／XE7]
AndroidアプリケーションのNFC機能活用

略歴
1978 年 3 月 26 日生まれ
2001 年 龍谷大学法学部卒業
2005 年 7 月 株式会社ミガロ. 入社
2005 年 7 月システム事業部配属
2007 年 4 月 RAD 事業部配属

現在の仕事内容
Delphi/400 や JC/400 の 製 品 試験および月 100 件に及ぶ問合せサポートやセミナ講師などを担当している。

1.はじめに

近頃では、物を購入する際に直接お金で支払うのではなく、クレジットカードやおサイフケータイなどを使ってデジタルに決済することが多くなってきた。たとえば、電車などの交通機関では、改札は物理的な切符よりもICカードでの決済のほうが主流になっている。

こうしたICカードの読み取りには、もちろんプログラムによる制御が行われている。この技術は、これまでは限定的な場面や機器で使われていたため、どちらかというと「特殊な技術」と考えられることが多かったが、スマートフォンの急速な普及により、身近なところで活用される機会が増えてきた。また最近ではIoT（Internet of Things）に関する技術が注目されており、NFCはその有効な手段の1つである。

本稿では、こうした背景を踏まえ、ICカードなどの情報をやり取りするためのNFCという通信技術を題材としている。NFCの基本的な情報から、Delphi/400のプログラムで実装する手法までを紹介する。

なお本稿では、スマートデバイスを使ったAndroidの開発が中心となるため、Delphi/400のバージョンはXE5とXE7を対象としている。

2.NFCについて

2-1.NFCとは

ICカードなどを使う際に、多くの場合はNFCと呼ばれる通信技術が使われている。NFCは「Near Field Communication」の略称で、無線通信の国際規格である。十数センチの距離で通信を行える小電力無線通信技術で、最近はスマートフォンやデジタルカメラ、電化製品などで広く採用されている。

使用できる製品には、【図1】のようなロゴマークが付けられていることが多い。NFCでは、ICカードやスマートフォンなどの対応機器を近距離でかざすだけで、電子マネー決済やデータ転送などの情報のやり取りができる。このNFCの特徴は、通信を行う機器間において複雑な操作を必要としない点である。

2-2. 身近で使われるNFC

身近なところでは、先ほど触れた交通機関のICカード（Suica等）などの電子マネーのほかに、おサイフケータイや免許証などにもICチップが搭載され、NFCが活用されている。【図2】

企業では、ビルの入退館管理にICカードを利用したり、工場などでは商品にRFIDなどを付け、製造工程や在庫管理に活用している場合も多い。

2-3. NFCの規格

NFCは、大きく分けて、以下の3つの機能で構成されている。

①リーダライター機能
（NFCタグに読み書きする）
②カードエミュレーション機能
（ICカードで決済する）
③P2P機能
（NFCデバイス同士で通信する）

図1　NFCロゴマーク

図2　NFCが使われるカード等

図3　NFCの規格

図4　Felica対応カード

NFC は、これらの機能に応じて、RFID 関連（①②）と通信関連（③）として規格されている。【図 3】

RFID と は、「Radio Frequency Identification」の略称で、無線通信による認証技術である。IC カードなどは、この RFID の規格に属する。また簡易に読み書きできるものとして、NFC タグと呼ばれる安価なシール形式のタグも普及している。

通信関連は、Bluetooth や Wi-Fi、P2P などの分野で規格されている。本稿では主に RFID を中心に説明していく。

RFID の中にも、いくつかの規格があり、IC カードによっても使われている規格が異なる。

たとえば、TypeA（ISO14443）は、成人識別 IC カード「taspo（タスポ）」で使われていることで有名である。TypeB は、役所関連でよく使用されており、免許証がその代表である。

しかし、日本で一番使われているのは TypeA でも TypeB でもなく、「FeliCa（フェリカ）」である。FeliCa はソニーが開発した独自の無線通信技術規格で、当初 TypeC 認定を目指していたが、認定はまだされていない。

FeliCa は、Suica や ICOCA などの交通カードに代表され、日本の IC カードのほとんどはこの FeliCa が採用されている。【図 4】

FeliCa の特徴は、ソニーが細部まで厳しく規格を決めているため、曖昧な情報が少なく、IC カードへのアクセスが極めて速い点である。

このアクセス速度、反応速度は、NFC の近距離通信では非常に重要である。たとえば、駅の改札で IC カードの反応が悪ければ使い物にならないが、FeliCa の反応速度は 1 秒もかからないため、交通カードのほとんどで FeliCa が採用されている。

2-4. バーコードとの違い

情報を読み取るという技術では、従来から使われているバーコードや QR コードもある。

ここで、NFC とバーコードや QR コードとの違いについて考えてみる。

一番大きな違いとしては、バーコードと QR コードは一方通行の読み取りしかできないのに対して、NFC は IC カードや NFC タグに読み書きができるという点である。

たとえば、バーコードや QR コードで商品を検品する場合、読み取る機器側で情報を収集するが、NFC では読み取りだけでなく商品側の RFID に書き込みもできるので、「検品済み」といったステータスを書き込むこともできる。

またバーコードや QR コードは、一度印刷してしまうと変更できないが、NFC は読み書きできるので、同じ媒体を何度でも使い回せる。読み取り専用の DVD と読み書き可能な DVD の違いと同じである。

操作性では、バーコードや QR コードを使うと読み取り部分にピントを合わせる必要があるが、NFC は近距離でかざすだけで読み取れる。非常に簡単で扱いやすいことも、NFC のメリットの 1 つである。【図 5】

もちろん、すべてがバーコードや QR コードより NFC が優れているわけではない。上記のよう便利さがある半面、バーコードや QR コードは紙媒体に印刷すれば使えるが、NFC は RFID などの IC チップが必要となるため、媒体のコストという点では非常に高くなる。安価な NFC タグであっても 1 枚あたり 100 ～ 200 円かかるため、大量な媒体が必要な場合、読み取りだけならバーコードのほうが採用されやすい。

そのため用途によって、NFC とバーコード、QR コードの使い分けが重要である。

3.NFCの活用

3-1. NFC の活用とアプリケーション

NFC には、バーコードなどと違い、読み書きできる特徴があることを説明したが、実際にどのような用途のアプリケーションに適用できるかを、A・B・C のパターンで分類してみた。

A. IC カードなどの読み取り・書き込み
＜用途例＞

IC カードの情報を読み込んだり、決済ができる。また、前述したように、入退室の認証管理や製品の検品・工程管理といった用途でも使用できる。

＜ NFC を使ったアプリケーション例＞
マルチ残高リーダ【図 6】

・IC カードの電子マネー残高をスマートデバイスでチェックできる。
https：//play.google.com/
（「マルチ残高リーダ」で検索）

B. デバイスの自動設定切替
＜用途例＞

NFC タグを使ってデバイスの設定を自動的に切り替える。機内モードや Wi-Fi の自動設定など、個人用途だけでなく商用施設の入り口などで使用される場合もある。

＜ NFC を使ったアプリケーション例＞
NFC タスクランチャー【図 7】

・自動切替の設定等を NFC タグに保存できる。スマートフォンでその NFC タグをかざせば、設定内容を自動切替する。
https：//play.google.com/
（「NFC タスクランチャー」で検索）

C. アプリやサイトの自動起動・連携
＜用途例＞

NFC タグを使ってアプリケーションや URL を自動起動できる。スマートポスターを使ったマーケティングでは、ポスターに IC チップが埋め込まれており、スマートデバイスをかざすと情報を入手することができる。またアクセス情報は IC チップにも書き込まれ、ポスターの場所ごとに地理的なアクセス情報などの集計・分析に使われる。

またポスター同様に、NFC タグをかざすことで、すぐに twitter と連動するような個人利用のアプリケーションにも使われている。

＜ NFC を使ったソリューション例＞
スマートポスター【図 8】
http：//www.hayato.info/tapee/

＜ NFC を使ったアプリケーション例＞
たっちなう【図 9】

・特定の NFC タグをかざすと twitter を連携できる。
https：//play.google.com/
（「たっちなう」で検索）

図5

QRコード読み込み　　NFCタグ読み込み

図6　マルチ残高リーダー

図7　NFCタスクランチャー

3-2. NFC とスマートデバイス

最近のスマートデバイスでは、NFC は重要な標準機能の 1 つとなっている。スマートデバイスといえば、iPhone ／ iPad の iOS と、Android が主流である。iOS では iPhone 6 で NFC が搭載されるようになったが、残念ながら iOS の NFC 機能は「Apple Pay」に限定されており、ユーザーが開発するアプリケーションで NFC を使う方法は提供されていない。

一方、Android では、NFC が標準搭載された OS 2.3 以降の端末では、ユーザーが NFC を活用したアプリケーションを作成できる。先に取り上げたアプリケーション例も Android である。そのため本稿では、NFC を使う題材として Android を取り上げることにした。

4.Delphi/400からのNFC機能利用

本章では、ここまで説明してきた NFC 機能のうち、IC カードや NFC タグなどの RFID 情報を、Delphi/400 で開発した Android アプリケーションから読み書きする方法を紹介していく。

Delphi/400 で Android の NFC 機能を使う場合は、専用のコンポーネントなどはないため、プログラムで NFC の呼出しを実装する必要がある。下記のサイトでは、Delphi での NFC 実装方法を、XE5 や XE7 のバージョンごとに非常に詳しく説明している。実際に動作するサンプルも用意されている。

Delphi and NFC on Android
（2014 年 9 月 10 日の記事）
http：//blog.blong.com/

ただし、このサイトの記事は英語なので、ここでダウンロードできるサンプルを題材に、NFC の実装に必要なポイントを説明していく。

ダウンロードできるサンプルは、「NFC_Samples_XEX.7z」というファイルである。7z というファイル圧縮形式なので、7-Zip 等の解凍ソフトを準備する必要がある。ファイルを展開すると Delphi のソースが一式揃っており、そのままコンパイルして使うことができる。【図 10A】【図 10B】

4-1. デバイスの設定

本稿では、NFC を標準で実装している Android を対象にするが、NFC を搭載しているからといって無条件に動作するわけではない。機種や OS によって違いはあるが、まずは、デバイスメニューのネットワーク関連の設定で、NFC 通信を有効にしておく必要がある。【図 11】は一例である。これを有効にしておかないと、NFC を搭載していても、RFID などにデバイスが反応しない。

4-2. プロジェクトの設定

次に、プログラムのプロジェクト設定を確認する。

「4-1.」ではデバイス側を設定したが、アプリケーション側も設定が必要となる。Android のプログラム作成時に、［プロジェクト］→［オプション］の［使用する権限］で細かい機能権限を設定できる。

ここで、NFC の機能をチェックして True に設定する【図 12】。これによりアプリケーションで NFC の機能を使用することが許可される。

4-3. XML の定義

NFC のアプリケーションにおいて、プログラムでどのような NFC 情報を扱うかは、XML ファイルを使って定義する。【図 13】

たとえば、スマートフォンに NFC 情報をかざしてアプリケーションを起動するには、対象となる NFC 情報を決めておく必要がある。この情報には AndroidManifest.xml、nfc.tec-discovered.filter.xml という 2 つの XML を使用する。

サンプルプログラムを確認すると、［プロジェクト］→［配置］で AndroidManifest.xml ファイルが組み込まれているのがわかる【図 14】。この xml ファイルはプロジェクトと同じフォルダに存在する。AndroidManifest.xml ファイルを Delphi 上で開くと、中身を確認できる（テキストエディタでも可能）。

ポイントは大きく 2 つある。

1 つは、「android.hardware.nfc」という設定を True で設定しておくことである【図 15】。これは、NFC のデバイス（ハードウェア）機能を要求するという設定になる。

もう 1 つは、「ACTION_NDEF_DISCOVERED」という設定である。これは、アプリケーションで扱いたい NFC のデータタイプを検知したら、アプリケーションを起動できるようにするフィルタ（intent-filter と呼ぶ）の種類の設定である。これによって使用する NFC のタイプを特定し、必要な NFC 情報を受信した場合だけ、アプリケーションでデータを読み込むことができる【図 16】。また、先に触れた NFC の規格にある TypeA や TypeB、FeliCa など対象のタイプを細かく定義することもできる。その場合、res\xml フォルダに「nfc.tec-discovered.filter.xml」という xml を用意し、対象となる規格リストを記述しておく。【図 17】

こうした細かい設定をしておくと、たとえば FeliCa だけに反応するアプリケーションを開発できる。

4-4. JNI の組み込み

実際に、Android プログラムから NFC の機能を使用する場合は、JNI（Java Native Interface）経由で Android SDK ／ API の android.nfc.NfcAdapter を利用する必要がある。

JNI は、Java で作成されたプログラムを他の言語から呼び出すためのインターフェース仕様である。しかし、これは名前の通り、Java モジュールであるため、Delphi には標準で用意されていない。

ただし、今回のサンプルでは、この JNI を利用するためのラッパーソース（Androidapi.JNI.Nfc.pas）が Common フォルダに用意されている。これを利用すると、Delphi から JNI の機能を簡単に呼び出すことができる。【図 18】

また、NFCHelper.pas というユニットも用意されており、NFC を扱う際に非常に便利である。このユニットには、NFC に対する Read や Write などのメソッドなどが定義されている。

4-5. NFC タグの読み込み実装

NFC の読み込みは、NFCHelper の HandleNfcTag を使って取得できる。【図 19】

NFC の読み書きをテストする場合、「2.NFC について」でも説明した、簡易

図8　スマートポスターのマーケティング

図9　たっちなう

な読み書きが可能なNFCタグと呼ばれるタグシールを使うと便利である。サンプルでは、スキャンからの起動時（OnFormActivateイベント）に取得している。【ソース1】

新規のNFCタグを読み込んだ場合はEmptyが表示されるが、書き込み情報がある場合、テキストに出力できる。たとえば免許証などを読み込むとTypeBとして表示されるので、正しく読み込めていることがわかる【図19】。ただし、免許証や交通カードなど秘匿性があるデータは、単純には取得できないようになっている。

4-6. NFCタグの書き込み実装

NFCの書き込みは、HelperのWriteTagTextを使って実装できる【図20】。サンプルでは、「WriteTag」ボタンの押下時（OnClickイベント）に書き込みをしている【ソース2】。このようにNFCHelper.pasを利用すると、NFCに対する実装が非常に簡単に行える。

これで、Delphi/400を使ったNFCに対する基本的な読み書きの実装方法が確認できた。

今回利用したNFCHelper.pasには、StringToJString等のJavaのデータと互換性をもつ関数も用意されているので、一通り機能を確認してから利用するとよいだろう（無駄にプログラムを作成しなくて済む）。

5.Android以外でのNFC利用（補足）

5-1. Windowsでの NFC利用

本稿では、Androidを使ったプログラムを題材に説明してきた。iOSのNFC機能は、「Apple Pay」に限定されていることを先述したが、PCについても調査した。

PC（Windows）ではあまり聞くことがないが、実はWindows 8やWindows 10ではNFC機能に対応している。

PCであまり使われていない理由としては、Windows（OS）が対応していてもハードウェアが対応していないと、NFCを利用できないからである（タッチ画面と同様）。そのため、WindowsOSでNFC機能に対応していても、PC自体がNFCに対応していないと使えないというのが実態である。

現実的には、外付けのNFCアダプターなどを用意することで対応が可能である。またこうしたNFCアダプターには、ハードに合わせたSDK（開発ツール）などが用意されていることが多く、たとえば下記のような製品では、Delphiなどのソースサンプルも提供されている。

＜NFC開発スタートキット101-AB＞
http：//www.orangetags.jp/

こうした外付けのNFCアダプターを利用すれば、バーコードリーダーなどをPC側の入力として利用し、FeliCaなどのICカードを読み取ったデータをアプリケーションの入力として利用できる。【図21】

またSDK（開発ツール）が用意されている場合は、これが本稿でのJNIのようなインターフェースとして使用できるため、NFCアダプターとの連携の開発をしなくともプログラムでは簡単に実装することが可能である。

6.まとめ

本稿ではNFCの基本情報の確認と、Androidを使ったDelphi/400プログラムの実装方法について検証・説明した。NFCを使ったプログラムは、スマートデバイスの普及に伴い、アプリケーションに求められる一般的な機能になると考えられる。なぜなら、より簡単でわかりやすい操作がアプリケーションにとって非常に重要な要素だからである。身近なところで、改札や入退室でのICカード使用の浸透が、それを裏付けている。

今、スマートデバイスは企業でも業務利用が広がり、PCアプリケーションとは異なる新しいユーザーインターフェースが求められている。NFC機能も、スマートデバイスの新しいユーザーインターフェースの1つである。今後は、特にIoTも視野に入れ、新しいユーザーインターフェースを利用する先進のビジネスモデルを考えていくことが必要だろう。そうした際、NFCは不可欠の技術になることを、今回の検証で確信できた。

M

図10A　ダウンロードソース

名前	更新日時	種類
res	2014/09/09 3:31	ファイル フォル...
AndroidManifest.template.xml	2014/09/04 7:16	XML ドキュメント
MainFormU.fmx	2014/09/09 3:32	FireMonkey フ...
MainFormU.pas	2014/09/09 3:33	Delphi ソース フ...
NFC_Sample.deployproj	2014/08/08 6:54	DEPLOYPROJ フ...
NFC_Sample.dpr	2014/09/04 6:31	Delphi プロジェ...
NFC_Sample.dproj	2014/09/09 3:34	Delphi プロジェ...
Readme.txt	2014/08/21 5:16	テキスト文書

図10B　サンプルプログラム

図11　NFCのデバイス設定

図12　NFCの使用権限設定

図13　XMLによる読み込み制御

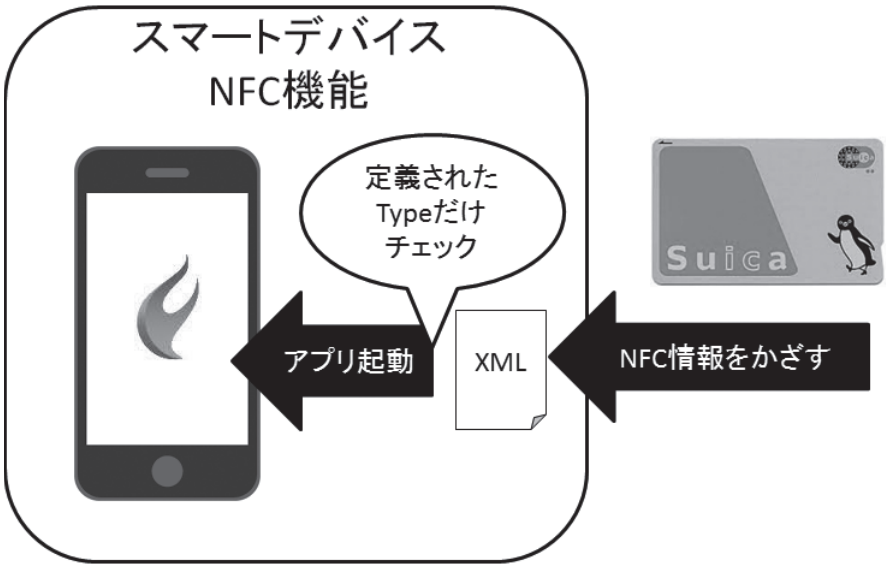


図14　AndroidManifest.xmlの配置

図15　NFC機能の要求

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

図16　Intent-filter

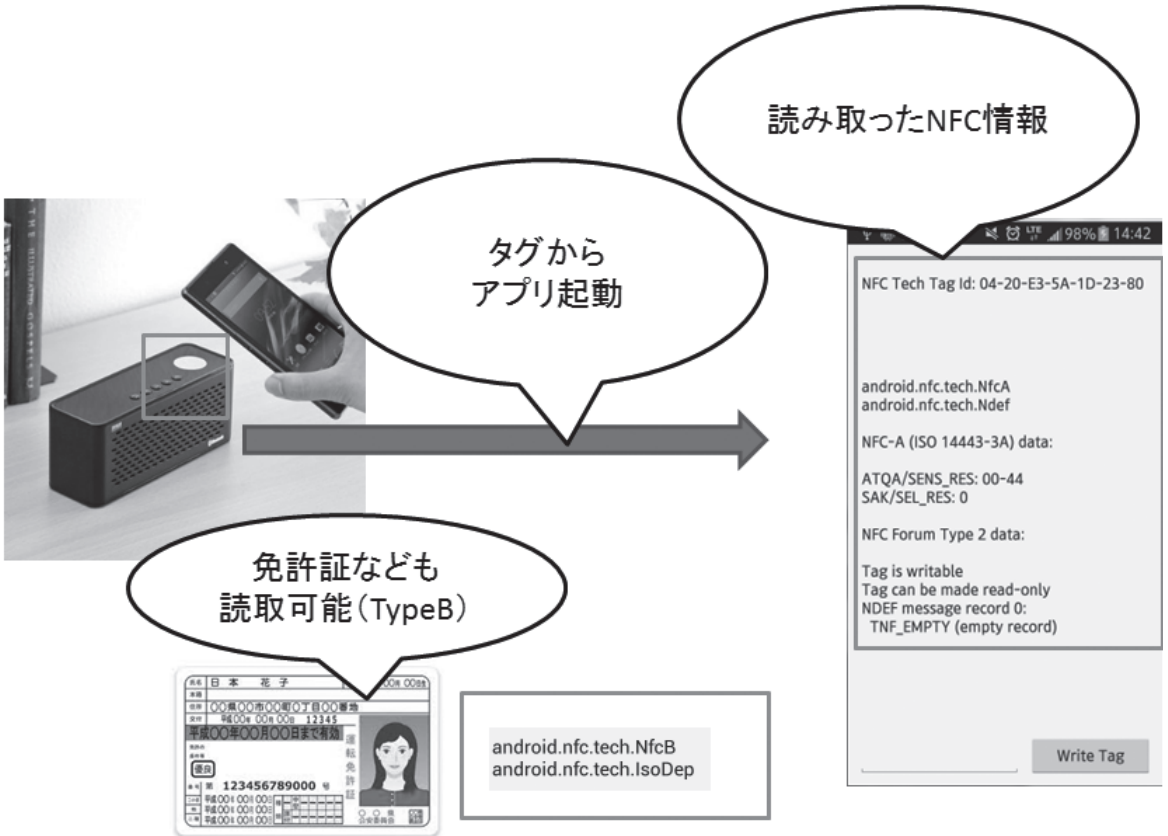
```
<intent-filter>
    <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="text/plain" />
</intent-filter>
```

図17　対象規格のリスト

```
<resources xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2">
    <tech-list>
        <tech>android.nfc.tech.IsoDep</tech>
    </tech-list>
    <tech-list>
        <tech>android.nfc.tech.NfcA</tech>
    </tech-list>
    <tech-list>
        <tech>android.nfc.tech.NfcB</tech>
    </tech-list>
    <tech-list>
        <tech>android.nfc.tech.NfcF</tech>
    </tech-list>
    <tech-list>
        <tech>android.nfc.tech.NfcV</tech>
```

図18　JNIを使うためのソース

図19　NFCタグの読み取り

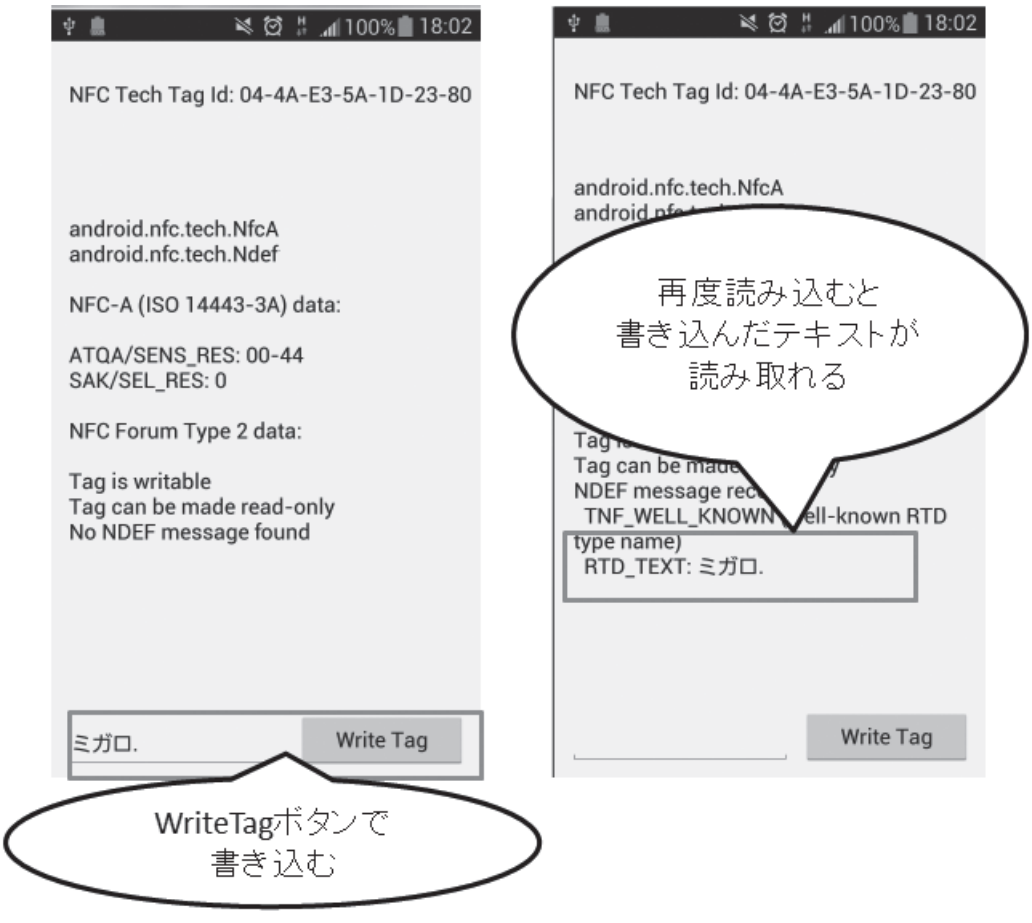


ソース1

NFCタグ読取処理例(Android)

```
procedure TMainForm.FormActivate(Sender:  TObject);
var
  Intent: JIntent;
  TagParcel: JParcelable;
  Tag: JTag;
begin
  Intent := SharedActivity.getIntent;
  //インテントが有効かチェック
  if Intent <> nil then
  begin
    //NFCのアクションを選別
    if TJNfcAdapter.JavaClass.ACTION_NDEF_DISCOVERED.equals(Intent.getAction)  or
       TJNfcAdapter.JavaClass.ACTION_TECH_DISCOVERED.equals(Intent.getAction)  or
       TJNfcAdapter.JavaClass.ACTION_TAG_DISCOVERED.equals(Intent.getAction)  then
    begin
      TagParcel := Intent.getParcelableExtra(TJNfcAdapter.JavaClass.EXTRA_TAG);
      if TagParcel <> nil then
      begin
        Tag := TJTag.Wrap((TagParcel as ILocalObject).GetObjectID);
      end;
      InfoLabel.Text := '';
      //タグの読み込み情報を書き出し
      NFCTagIdLabel.Text  := HandleNfcTag(Tag,
        procedure (const Msg: string)
        begin
          InfoLabel.Text := InfoLabel.Text + Msg + LineFeed;
        end);
    end;
  end;
end;
```

図20　NFCタグの書き込み



ソース2

NFCタグ書込処理例(Android)

```
procedure TMainForm.TagWriteButtonClick(Sender: TObject);
var
  NfcAdapter: JNfcAdapter;
  TagParcel: JParcelable;
  Tag: JTag;
  Intent: JIntent;
begin
  //NFCアダプターをハンドリング
  NfcAdapter := TJNfcAdapter.JavaClass.getDefaultAdapter(SharedActivityContext);
  if (NfcAdapter <> nil) and NfcAdapter.isEnabled then
  begin
    //インテントを取得
    Intent := SharedActivity.getIntent;
    TagParcel := Intent.getParcelableExtra(TJNfcAdapter.JavaClass.EXTRA_TAG);
    if TagParcel <> nil then
    begin
      Tag := TJTag.Wrap((TagParcel as ILocalObject).GetObjectID);
      //タグに情報書き込み
      if not WriteTagText(TagWriteEdit.Text,  Tag) then
        raise Exception.Create('Error connecting to tag');
    end;
  end
  else
    raise Exception.Create('NFC is not available');
end;
```

図21　NFCアダプター例