吉原 泰介

株式会社ミガロ.

RAD事業部 技術支援課

[Delphi/400] アプリケーションテザリングを利用したPC & モバイルアプリケーション連携

●はじめに
●アプリケーションテザリング技術
●アプリケーションテザリングを活用した連携開発
●おわりに

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を中心に製品試験および月100件に及ぶ問い合わせサポートやセミナー講師などを担当している。

1.はじめに

スマートフォンやタブレットなどのモバイル端末の普及が始まって数年が経ち、現在では個人はもちろん、企業内での活用も一般的になってきた。こうしたモバイル端末は、名前のとおり、持ち歩いて使用できる特性を活かした場面で利用されることが多い。

企業でモバイル端末が使われる主な用途としては、ブラウザ、電話、カメラ（写真撮影）などが挙げられる。

これらの機能は、モバイル端末での特殊な機能というよりも、基本的には既存のデバイス（たとえばPCやデジタルカメラ）で使用していた機能をモバイル端末で代用していることがわかる。業務によってはタブレットがPC自体の代用となり、置き換わっていることも多い。

つまりモバイル端末のアプリケーションでは、特殊な機能だけが求められるわけではなく、従来の業務で使っている機能をモバイル端末で利用できるだけでも、大きな価値や利便性が得られる。

本稿では、モバイルアプリケーション開発を難しくとらえず、従来の機能を代替えするという観点で「アプリケーションテザリング」という技術の活用方法について考察する。

2.アプリケーションテザリング技術

2-1. アプリケーションテザリングとは

最近スマートフォンによく搭載されている「テザリング」という機能がある。これはスマートフォンがWi-Fiルータの役割を果たし、スマートフォン経由でPCなどのインターネット通信を共有する機能である。

Delphi/400ではXE7より、「アプリケーションテザリング」という技術が実装されている。これは前述したインターネット共有とは若干異なるが、同じネットワーク（あるいはBluetooth接続された）アプリケーション同士でデータや処理を共有する機能である。アップテザリングと呼ばれることもある。

このアプリケーションテザリングは、モバイルアプリケーション開発のFireMonkeyでも、PCアプリケーション開発のVCLでも、同じコンポーネントを使って実装可能である。

つまり従来のPCアプリケーションと新規開発したモバイルアプリケーションを容易に連携できるわけだ。たとえばモバイルアプリケーションからPCアプリケーションに処理結果を連携して、PCアプリケーション側でIBM i に更新するといった仕組みも構築できる。

本稿ではこうしたアプリケーションテザリングの技術に着目し、PCアプリケーションとモバイルアプリケーションを連携する実装方法をまとめている。なお、開発環境の対象バージョンはDelphi/400 XE7以降を想定している。

2-2. アプリケーションテザリングでモバイル端末を活用

アプリケーションテザリングでモバイルアプリケーションを活用する実装例として、従来のシステムでPCと連携して

図1

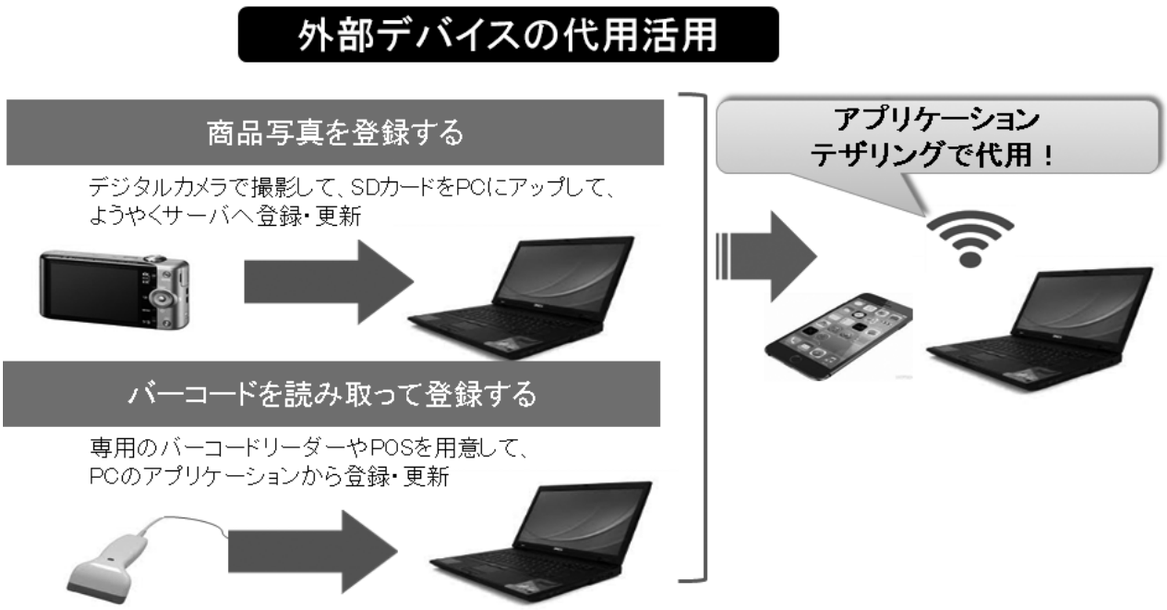


図2

いる外部デバイスを検討した。モバイル端末を活かせるものとして、次の2機能を題材とする。【図1】

①デジタルカメラで商品写真を撮り、画像データとしてシステムに取り込む機能
②バーコードリーダーやPOSでバーコードやQRコードをシステムに取り込む機能

いずれの機能も業務で使われることが多く、別デバイスで運用されているケースもよく見られる。また機能的にもモバイルアプリケーションに適しており、実装が容易である。

3.アプリケーションテザリングを活用した連携開発

3-1. アプリケーションテザリング用コンポーネント

アプリケーションテザリング機能を使用するためのコンポーネントを確認する。専用のコンポーネントとして、TTetheringManagerとTTetheringAppProfileの2つのコンポーネントが用意されている。それぞれの役割は、次のとおりである。【図2】

TTetheringManagerコンポーネント

同一ネットワークあるいはBluetoothの通信上で、アプリケーション同士の接続を管理する

TTetheringAppProfileコンポーネント

テザリングで接続したアプリケーション間で共有するリソースを制御する

この2つのコンポーネントは基本セットで使用し、TTetheringManagerで接続してTTetheringAppProfileで共有リソースを送受信する。たとえばコードや文字列、画像などのファイルをStream形式で扱うこともできる。【図3】

本稿では、モバイルアプリケーションで写真撮影やバーコード読み取りの機能を実装し、それをPCアプリケーションで受信して利用するアプリケーションを作成する。【図4】

3-2. モバイルアプリケーションの開発

まず、モバイル端末側のアプリケーションを作成する。

使用コンポーネントは写真撮影用にTActionList、TImageコンポーネント、バーコード読み取り用にTTMSFMXZBarReaderコンポーネント、それぞれの実行用にTButtonを2つ配置する。またアプリケーションテザリング用には、前述したTTetheringManagerコンポーネントとTTetheringAppProfileを配置しておく。【図5】

ここで使用するTTMSFMXZBarReaderコンポーネントは標準コンポーネントではないが、TMSSoftware社のHPから無償で使用できるiOS用バーコード読み取りコンポーネントである（http://www.tmssoftware.com/）。Androidを使う場合は、拡張したフリーソースのTTKRBarCodeScannerが使いやすい。

次にコンポーネントを設定する。テザリングの設定は、TTetheringAppProfileのGroupプロパティに通信するアプリケーションで共通の値を設定しておく。グループ名のようなものだと考えるとわかりやすい。【図6】

そしてアプリケーション間で共有するデータについては、Resourcesプロパティにアイテムを、次のように2つ追加して用意する。

1つ目は写真転送用で、アイテムのNameプロパティには"Camera"、ResTypeプロパティには写真（画像ファイル）を扱うために"Stream"を設定する。

2つ目はバーコード転送用で、アイテムのNameプロパティには"Barcode"、ResTypeプロパティにはバーコード値を扱うために"Data"を設定する。いずれもKindプロパティはデフォルト値"Shared"のままにしておく。【図6】

設定が完了したら、配置したコンポーネントにプログラム動作をコーディングしていく。撮影用のボタンには、カメラ操作の標準アクションであるTTakePhotoFromCameraActionを割り当て、OnDidFinishTakingイベントを作成する。【図7】

写真撮影終了時に発生するOnDidFinishTakingイベントには、【ソース1】のようなコードを記述する。

撮影した写真（画像ファイル）をTTetheringAppProfileのアイテムに代入することで、テザリングされた別のアプリケーションでアイテム情報を共有できる。

バーコード読み取り用のボタンには、OnClickイベントでTTMSFMXZBarReaderのShowメソッドを実行することで、バーコードスキャン画面を起動。バーコードスキャンされた際にOnGetResultイベントで取得したバーコード値を、TTetheringAppProfileのアイテムに代入するだけで実装できる。【ソース2】

最後に、フォームのOnCreateイベントでTTetheringManagerを接続するようにコーディングしておくことで、起動直後にPCアプリケーションとテザリング通信状態になる。

これだけの開発でモバイル端末の写真撮影とバーコード読み取りを実行し、PCへデータを転送（共有）するアプリケーションが完成である。

3-3. PCアプリケーションの開発（拡張実装）

次に、PC側のアプリケーションを実装する。PCアプリケーションは、写真やコードを扱う既存アプリケーションがすでに導入されていることを想定する。本稿では、飲料水の商品一覧・明細を扱う単純なアプリケーション例を題材に拡張実装している。

まずアプリケーションテザリング用として、モバイルアプリケーションと同様にTTetheringManagerコンポーネントとTTetheringAppProfileを追加で配置する。【図8】

プロパティの設定もモバイルアプリケーションと同様ではあるが、TTetheringAppProfileのアイテムでKindプロパティには"Mirror"を設定する。これは送られてきた共有アイテムを受信する側の設定である。

そして、それぞれのアイテムにはOnResourceReceivedのイベントがあるので、ここに転送されてきた情報を受信した際の処理をコーディングする。写真の受信については、【ソース3】のように、送られてきたStreamをLoadFromStreamに読み込むことで画面のTImageへ写真画像を反映できる。

図3

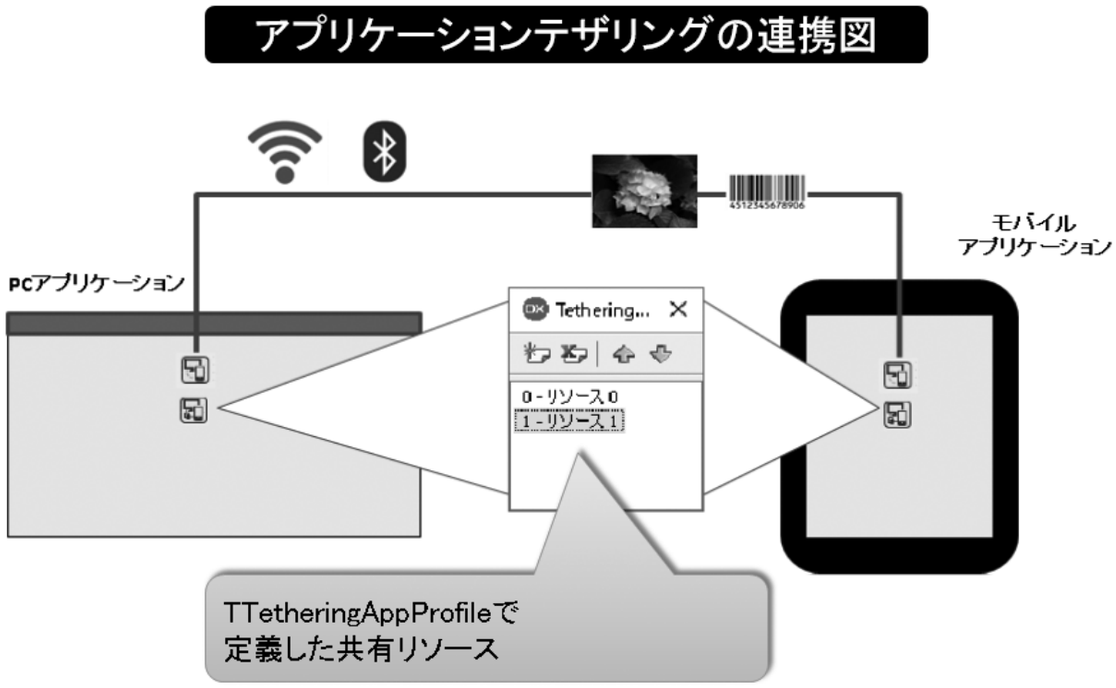
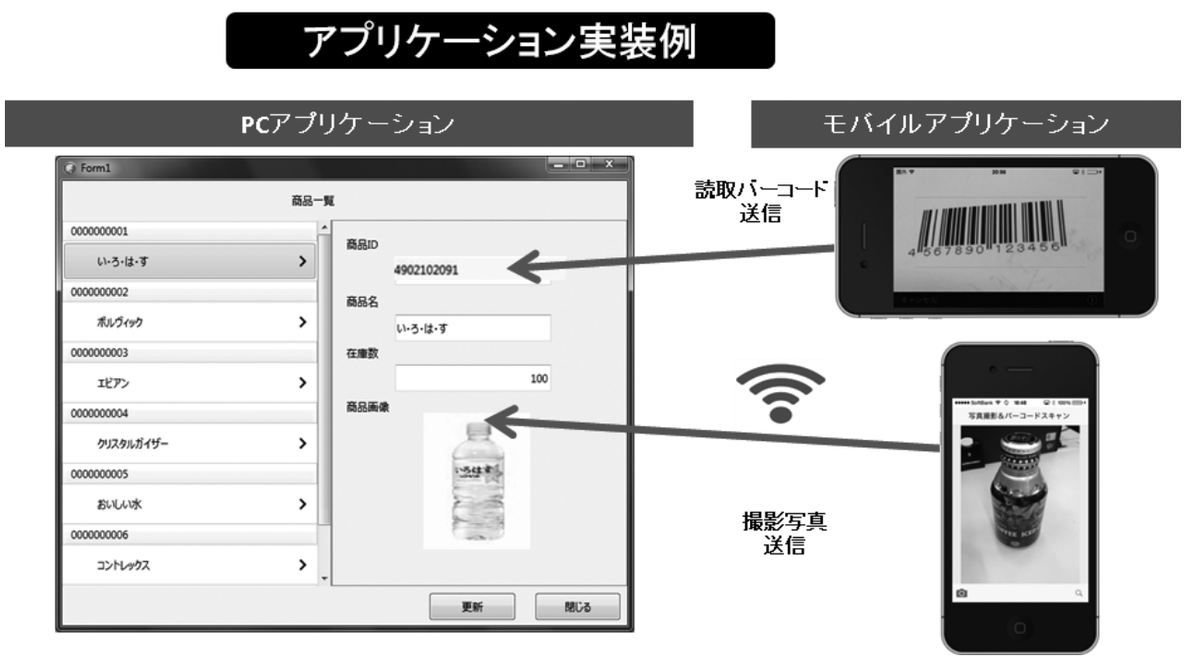


図4



バーコードの受信については、【ソース 4】のように、画面項目（ここではTEdit）に送られてきたコードを設定するだけである。これで、PC アプリケーション側のテザリング実装が完成した。

最後にコンパイルした PC アプリケーションとモバイルアプリケーションを同じネットワーク上、あるいは Bluetooth で通信が可能な環境で起動して動作確認する。

モバイルアプリケーションで写真を撮れば、PC 側に写真画像が表示され、バーコードをスキャンすれば、PC 側に読み取ったコードが自動入力される（【図 4】の実装例が完成する）。

これにより、デジタルカメラから画像を手動でアップしたり、専用端末でスキャンしていたバーコードなどの外部デバイス作業をモバイル端末と簡易なアプリだけで代用できる。

3-4. アプリケーションテザリングの応用活用

補足として、いくつかのアプリケーションテザリングの応用テクニックについても検証した。

アプリケーションテザリングでは、TTetheringAppProfile の Resources プロパティのアイテムを共有するだけでなく、アクション（処理）も共有できる。これは Resources プロパティと同様に、TTetheringAppProfile の Actions プロパティのアイテムとして設定する。【図 9】

Actions にアイテムを追加すると、アイテムの Action プロパティのリストに割り当てられる。この Action を共有しておけば、PC 側で設定している Action をモバイルアプリケーション側から実行可能になる。

たとえば、PC 上で PowerPoint を OLE で操作する Action 処理を作成しておき、モバイル端末でプレゼンテーション資料をリモコン操作するアプリケーションも簡単に実現できる。実際にそうしたリモコンとして使うモバイルアプリケーションは、市販製品として販売されている。

もちろん、このアプリケーションテザリング技術は PC アプリケーションとモバイルアプリケーションの連携だけではなく、PC アプリケーション同士、あるいはモバイルアプリケーション同士でも活用できる。同一システムでログインしている端末やデバイスのステータス情報を共有したり、アプリケーション間でのチャットなどのグループメッセージ機能としても幅広く応用が考えられる。

ただしこの技術は、冒頭で説明したとおり、同じネットワークあるいは Bluetooth などでの通信が前提である。たとえば、ネットワークに接続していない拠点や事業所間などでは活用できない。そのため、実際にアプリケーション同士で通信していなければ、データやアクションも共有できない。

本稿のプログラムでは、起動時にテザリング接続されることを前提としているが、接続されているかを可視化できるよう工夫しておいたほうが、使い勝手がよい。

たとえば、”接続中”と表示する TLabel を配置しておき、Visible プロパティを False にしておく。そして TTimer コンポーネントを配置し、OnTimer イベントを使って【ソース 5】のようにアプリケーションテザリングの接続カウントを監視して、Visible プロパティを True に設定する仕組みで実装できる。

接続のカウントは、TTethering Manager の RemoteProfiles.Count プロパティでリアルタイムな接続数を取得できる。もちろん TTimer のような常時監視型ではなく、ボタンなどで明示的に実行してチェックする仕組みでもよい。こうした実装を加えると、実行時に PC アプリケーションにモバイルアプリケーションが接続されると、”接続中”の画面表示になるので、操作時にわかりやすい。

4.おわりに

本稿ではアプリケーションテザリングという技術を検証し、既存の外部デバイスを使った機能をモバイルアプリケーションで簡単に代用できることを紹介した。単純に代用が可能になっただけでなく、どちらも Delphi/400 のアプリケーションで作成できる点が大きなメリットである。これにより、自社業務に合わせてアプリケーション側を柔軟に対応していける。

またアプリケーションテザリングを使ったモバイルアプリケーションであれば、DataSnap サーバーのような中間サーバーを構築しなくても、【図 10】のように PC アプリケーションがサーバーの代用となるので、簡単に実装・運用できる。DataSnap のような高度なサーバー機能はないが、モバイルアプリケーション開発の入門としても手軽に試せて便利である。

モバイル端末の企業導入は非常に増えてきたが、アプリケーションまで自社開発している企業はまだまだ少ない。

しかし本稿のプログラム例を一読いただければわかるが、これまでの PC アプリケーション開発と同じように、モバイルアプリケーションも Delphi 言語で開発できる。

日本では業務アプリケーションを自社開発する企業が多く、今後は PC アプリケーション同様にモバイルアプリケーションも自社開発の需要が増えてくると予想される。

そうしたシステム開発の中で、この Delphi/400 のアプリケーションテザリングやモバイルアプリケーション開発技術が役立つように、今後も技術検証や情報発信に努めていきたい。

M

図5

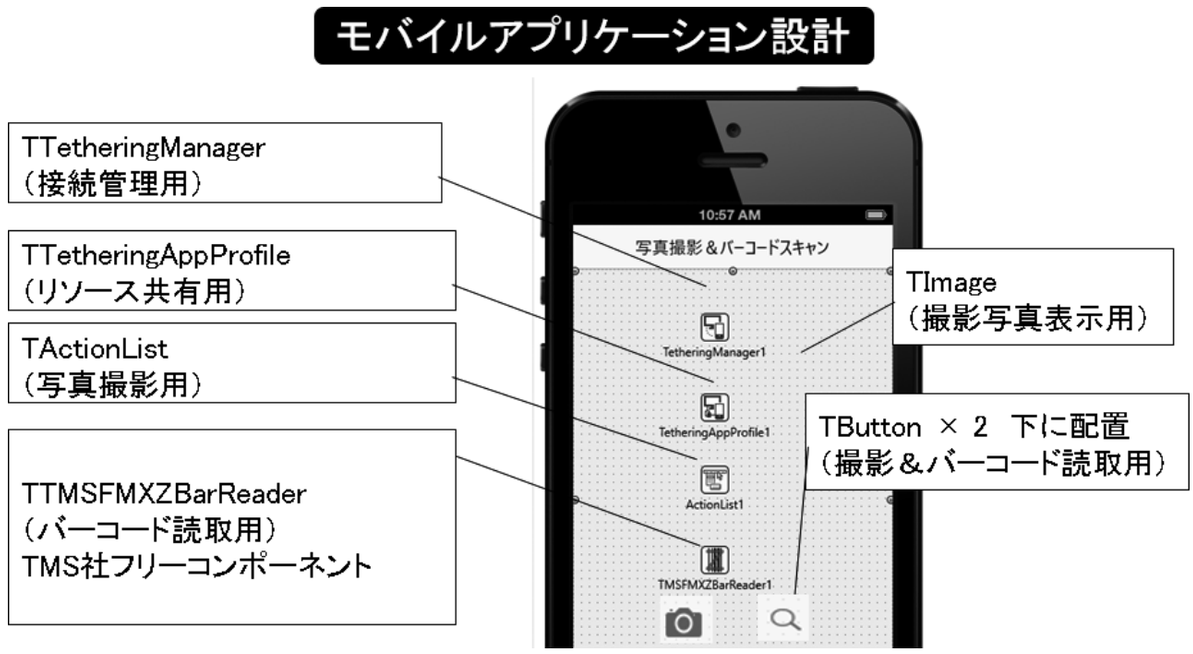


図6

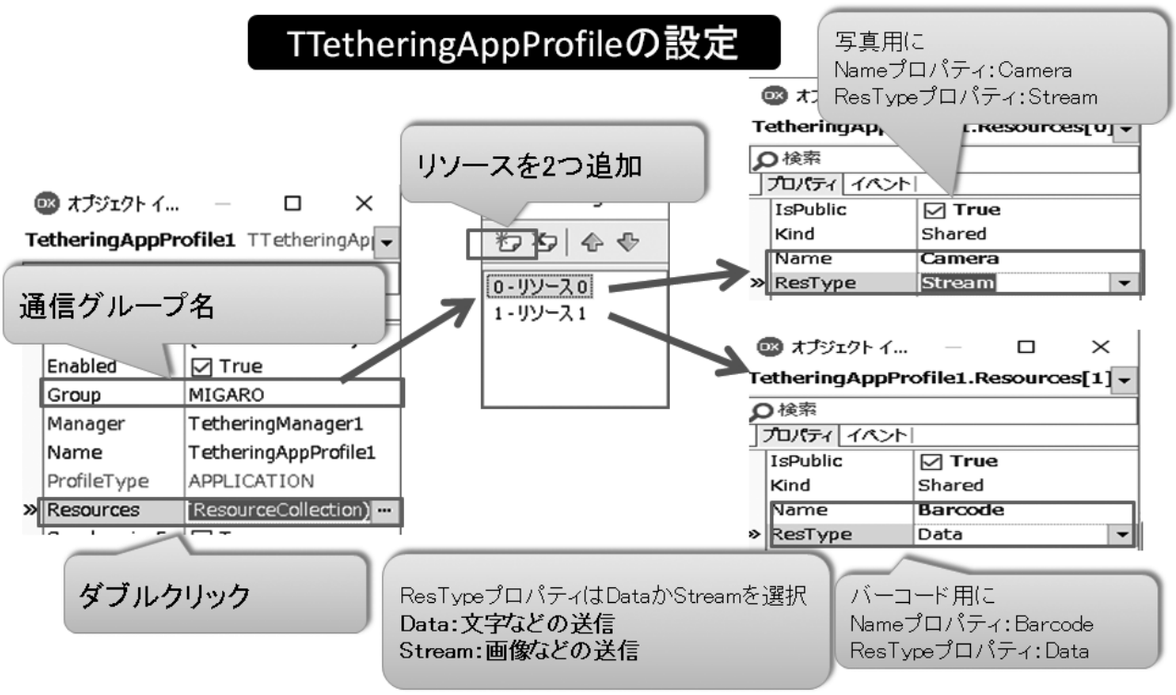
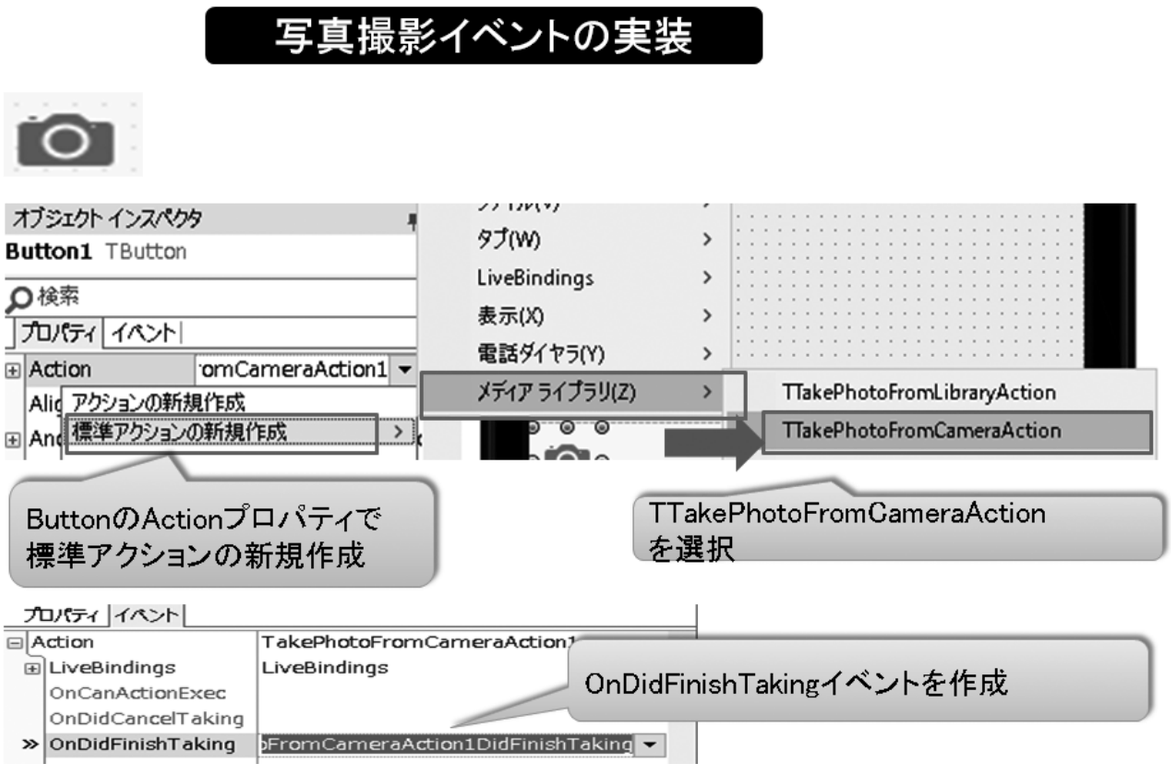


図7



ソース1

OnDidFinishTakingイベント(撮影写真を送信)

```
procedure TForm1.TakePhotoFromCameraAction1DidFinishTaking(Image: TBitmap);
var
  FStream: TMemoryStream;
begin
  FStream := TMemoryStream.Create;  //写真用のStreamを作成
  image.SaveToStream(FStream);      //撮影写真をStreamに格納
  TetheringAppProfile1.Resources.Items[0].Value := FStream; //共有リソースに送信
  Image1.Bitmap.Assign(Image);      //画面に写真を表示
end;
```

ソース2

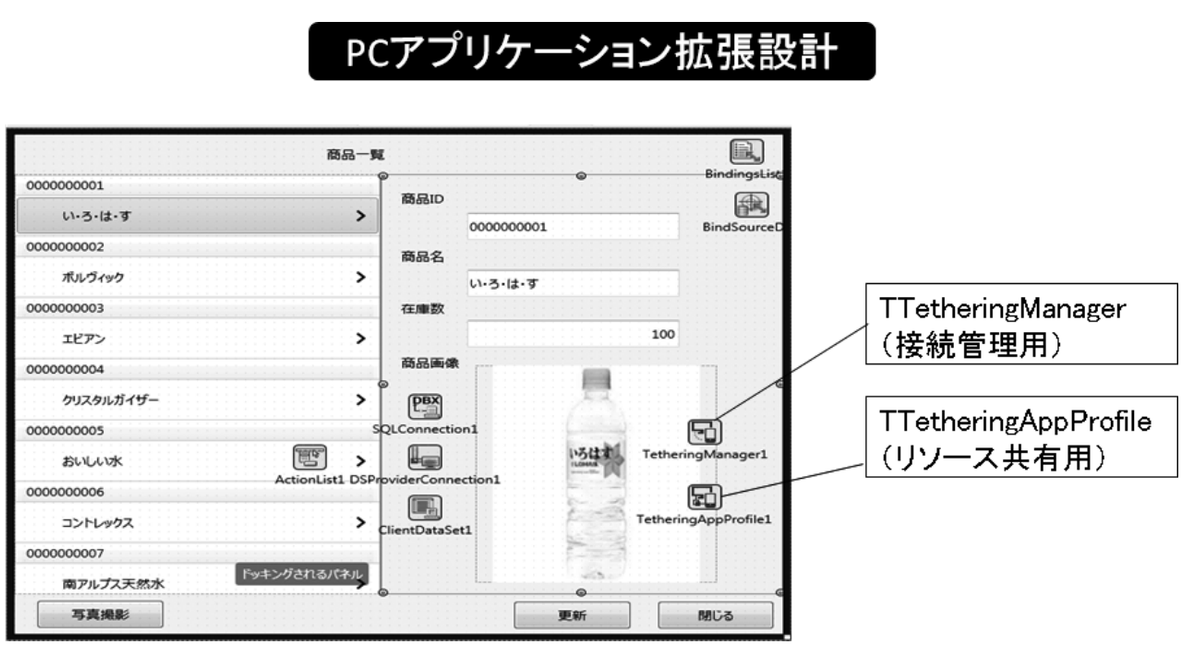
OnClickイベント（バーコード撮影起動）

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  TMSFMXZBarReader1.Show; //バーコード撮影を起動
end;
```

OnGetResultイベント（取得バーコード送信）

```
procedure TForm1.TMSFMXZBarReader1GetResult(Sender: TObject; AResult: string);
Begin
 //読み取ったバーコード値を共有リソースに送信
  TetheringAppProfile1.Resources.Items[1].Value := AResult;
end;
```

図8



ソース3

OnResourceReceivedイベント(撮影写真を受信)

```
//ソース1で送信された画像を受信する
procedure TForm1.TetheringAppProfile1ResourcesOResourceReceived
  (const Sender: TObject; const AResource: TRemoteResource);
begin
  AResource.Value.AsStream.Position := 0;                    //Streamのポジション
  Image1.Bitmap.LoadFromStream(AResource.Value.AsStream);    //画面に受信画像を設定
  Image1.Repaint;                                            //再描画
end;
```

ソース4

OnResourceReceivedイベント(取得バーコードを受信)

```
//ソース2で送信されたバーコード値を受信する
procedure TForm1.TetheringAppProfile1Resources1ResourceReceived
  (const Sender: TObject; const AResource: TRemoteResource);
begin
  Edit1.Text := AResource.Value.AsString;  //画面に受信値を設定
end;
```

図9

ソース5

OnTimerイベント(接続表示)

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  Label1.Visible := (TetheringManager1.RemoteProfiles.Count > 0);
end;
```

接続カウントがあれば
Labelを表示

図10