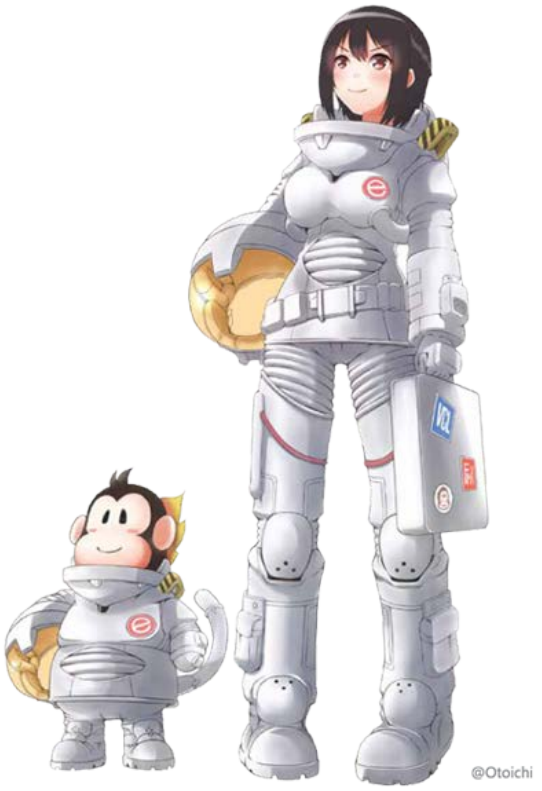
既存のVCLアプリでもモバイル活用！
簡単マルチデバイス連携術

第34回 エンバカデロ・デベロッパーキャンプ



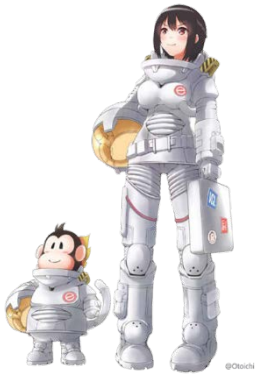
エンバカデロ・テクノロジーズ
毛利　春幸

embarcadero®
DEVELOPER CAMP

はじめに

- 既存のVCLアプリケーションからでは、大規模な改修をしないとモバイルやクラウドを活用できないと諦めていませんか？

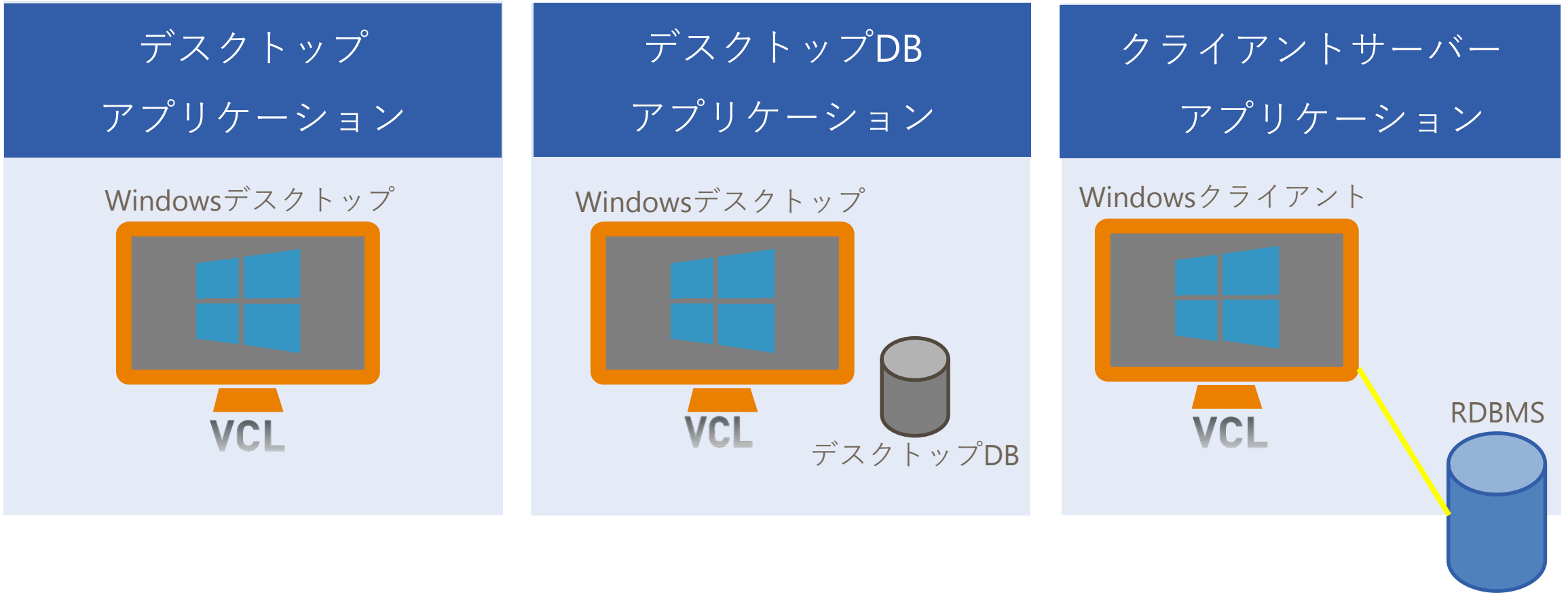
  アプリケーション間連携の手法を用いれば、コンポーネントによって簡単にモバイル、多様なデータソース、クラウドを利用できます。
  新たにモバイル向けのフルシステムを組むのではなく、今までのVCLアプリと連動させたシンプルなモバイル化の例をファーストステップとして、多様なデータベース、クラウド接続へと展開していく方法を説明します。



embarcadero DEVELOPER CAMP

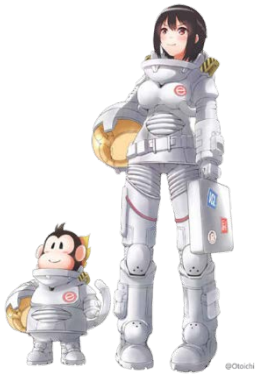
既存のVCLアプリケーション

- 大規模な改修をしないとモバイルやクラウドを活用できないと諦めていませんか？



コンポーネントによって簡単にモバイル

- 事例
  - 既存の見積管理システム(VCL)

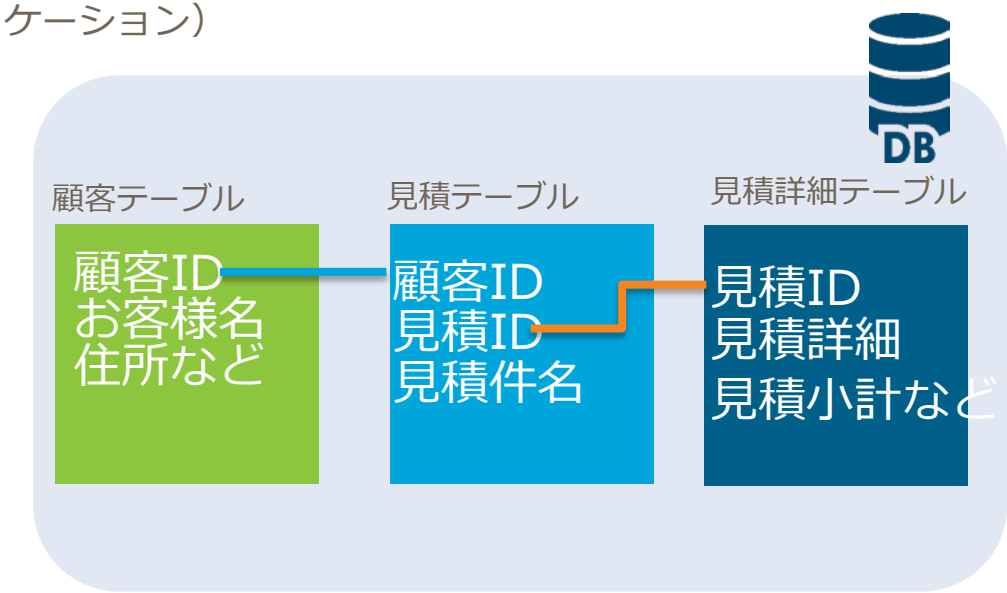


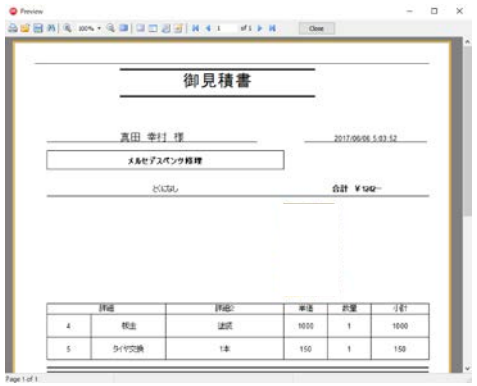
embarcadero DEVELOPER CAMP

既存の見積管理システム

- 見積管理システムの画面（Delphiで構築したWindowsアプリケーション）
  - 顧客管理
  - 見積管理
  - 見積詳細(印刷)

- 使用機能等
  - データベース: InterBase
  - 印刷・帳票： FastReport

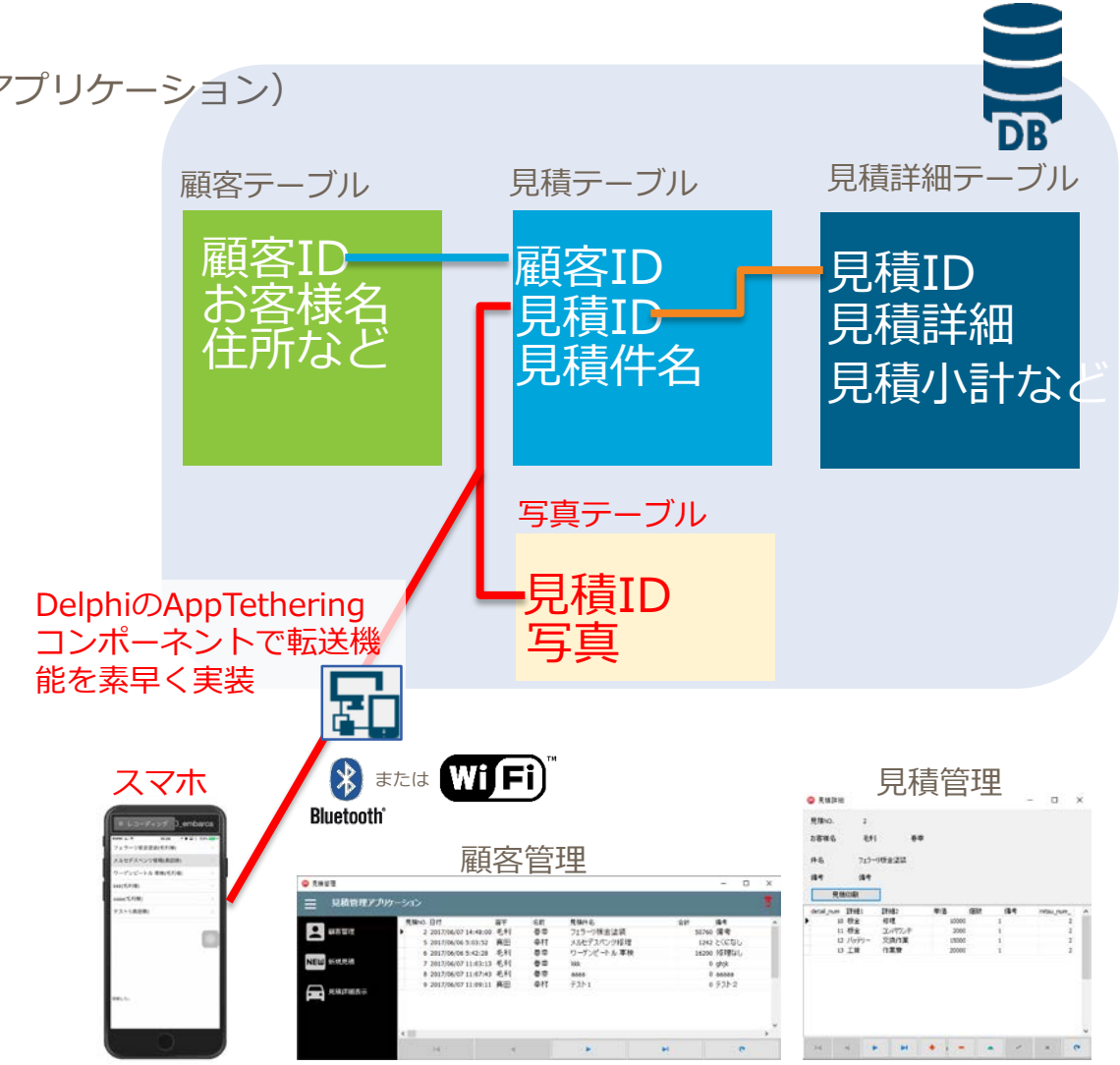




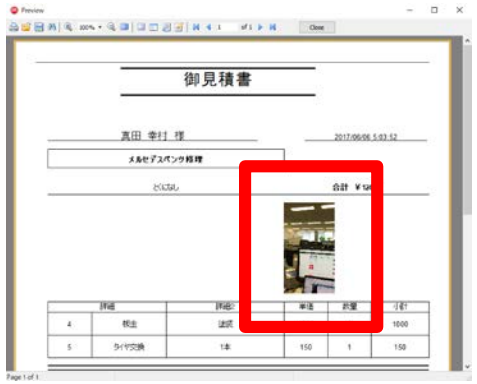
見積管理システムのモダナイゼーション

- 見積管理システムの画面（Delphiで構築したWindowsアプリケーション）
  - 顧客管理
  - 見積管理
  - 見積詳細(印刷)
  - 写真取得アプリ

- 使用機能等
  - データベース: InterBase
  - 印刷・帳票： FastReport
  - アプリ通信：AppTethering

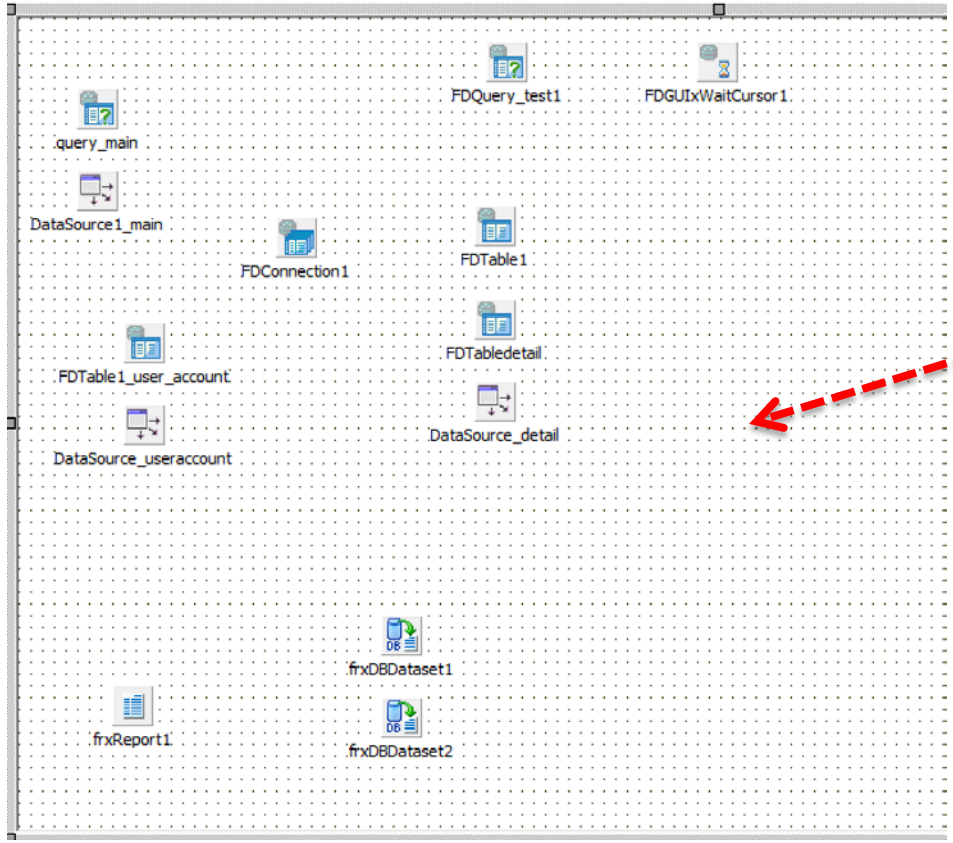


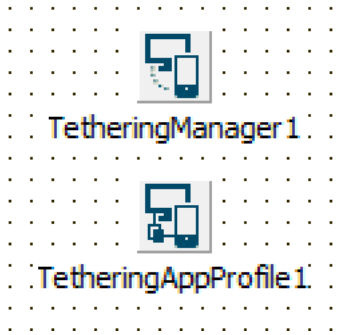
見積詳細印刷



データモジュール

- データモジュール内に TFDConnection, TFDQuery, TDataSourceなど既存で利用されたコンポーネント





既存システム側は
TTetheringManagerとTTetheringAppProfileを
追加で配置するだけです。(設定は必要です)

アプリケーション テザリング

- RTL にはアプリケーション テザリング コンポーネントが用意されているため、アプリケーションは同じマシンまたはリモート マシンで動作している他のアプリケーションとやり取りできます。
- アプリケーションでは、アプリケーション テザリングを使用して次のことをたやすく行えます。
  - アプリケーション テザリングを使用している他のアプリケーションの検出
    - （対象は、当アプリケーションと同じデバイスか接続している他のデバイスで動作しているもの）。
  - アクションのリモート実行
    - あるアプリケーションでアプリケーション テザリングを使ってアクションを公開できます。そのとき、他のアプリケーションでは、前者のアプリケーションに用意されているこれらのアクションのどれでもリモートで呼び出すことができます。
  - アプリケーション間でのデータの共有
    - アプリケーション テザリングを使用すると、標準データ型とストリームを共有できます。

アプリケーション テザリング プロトコル

- 特定のトランスポートやプロトコルに依存せず、アプリケーション テザリング API を使用して新しいトランスポートおよびプロトコルを実装することもできます。 RTL では、IP 接続および クラシック Bluetooth 接続に対するサポートが組み込まれています。IP は、同じデバイスで動作するアプリケーション間の接続もサポートします。

プラットフォーム	IP 接続	クラシック Bluetooth 接続
Windows	○	○
OS X	○	○
iOS	○	-
Android	○	○

アプリケーション テザリング | コンポーネント

- アプリケーション テザリングには次のコンポーネントが必要です。

名称	説明
TTetheringManager	アプリケーション テザリングを使用している他のアプリケーションを検出できます（対象となるのは、当該アプリケーションと同じデバイスか接続している他のデバイスで動作しているもの）
TTetheringAppProfile	当該アプリケーションが TTetheringManager を使って前もってペアにされた他のアプリケーションと共有するアクションおよびデータを定義できます

アプリケーション テザリング | リモート アプリケーション接続

- 2 つのアプリケーション間で TTetheringAppProfile コンポーネント間でアクションおよびデータを共有できるようにする

- アプリケーションどうしが自分の TTetheringManager を使って互いを検出してペアにする必要があり、その後で、それらのアプリケーションの TTetheringAppProfile がリンクされます。

アダプタおよびプロトコルの要件

- アプリケーションのペア設定用とデータの交換用にアプリケーション テザリング機能に用意されているさまざまなアダプタとプロトコル

型名	用途
TTetheringNetworkAdapterCommon	同じデバイスで動作しているアプリケーション間や、ローカル エリア ネットワーク（LAN）やインターネットなどの IP ネットワークを介して到達可能なアプリケーション間の接続用のアダプタ
TTetheringBluetoothAdapter	クラシック Bluetooth を使って到達可能かつペアになっている別個のデバイスで動作しているアプリケーション間の接続用のアダプタ
TTetheringTCPProtocol	上記のアダプタで使用される通信プロトコル

アプリケーション接続

方法	詳細
グループを使用してアプリケーションに接続	各アプリケーションの TTetheringAppProfile コンポーネントの Group プロパティに同じ文字列を入力します。 各アプリケーションの TTetheringManager コンポーネントの AutoConnect メソッドを実行時に呼び出します。 たまたま同じ Group 文字列を使用している第三者のアプリケーションに当アプリケーションが誤って接続しないように、Group 文字列には、できるだけ一意なものを選びます。たとえば、Group の文字列として UUID を使用できます。
手動でリモート アプリケーションに接続	1. DiscoverManagers メソッドを呼び出して、アプリケーション テザリングを使用している他のアプリケーションの検出 2. RemoteManagers のリストから、TTetheringManager.PairManager を呼び出して、希望するものとペアにします 3. 検出した RemoteProfiles のリストを読み取り、TTetheringAppProfile コンポーネントに対して Connect

サブネット外部のアプリケーションへの接続

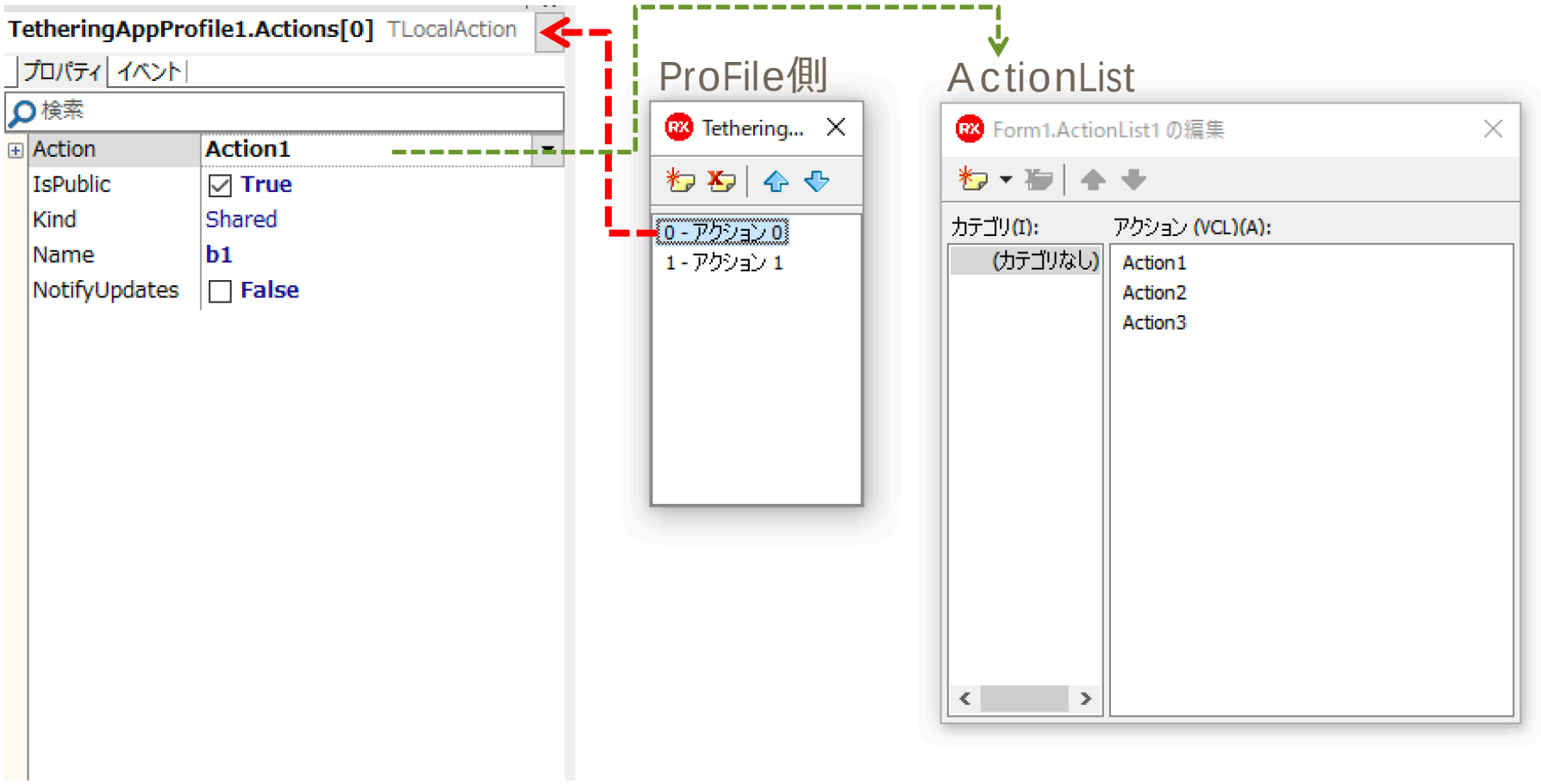
- デフォルトでは、AutoConnect と DiscoverManagers はどちらも、アプリケーションの動作デバイスが存在するローカル エリア ネットワーク（LAN）のサブネットで検出を実行します。

メソッドのオプション パラメータ Target を使用して、この動作をオーバーライドし
IP アドレスやサブネットを指定することができます。

リモート マネージャを検索する IP アドレスを指定するには、その IP アドレスを Target として指定します。
IP アドレスのサブネットを指定するには、4 番目の数を 0 に設定した IP アドレスを指定します。たとえば、Target に
"192.168.4.0" と指定した場合、マネージャは 192.168.4.x サブネットでリモート マネージャを検索します。
(これより広いサブネットを指定することはできません。たとえば、"192.168.0.0" はサポートされていません。)

アプリケーション テザリング | ActionListの共有

- ActionListとTetheringAppProfile.Actionsに入れるだけです。

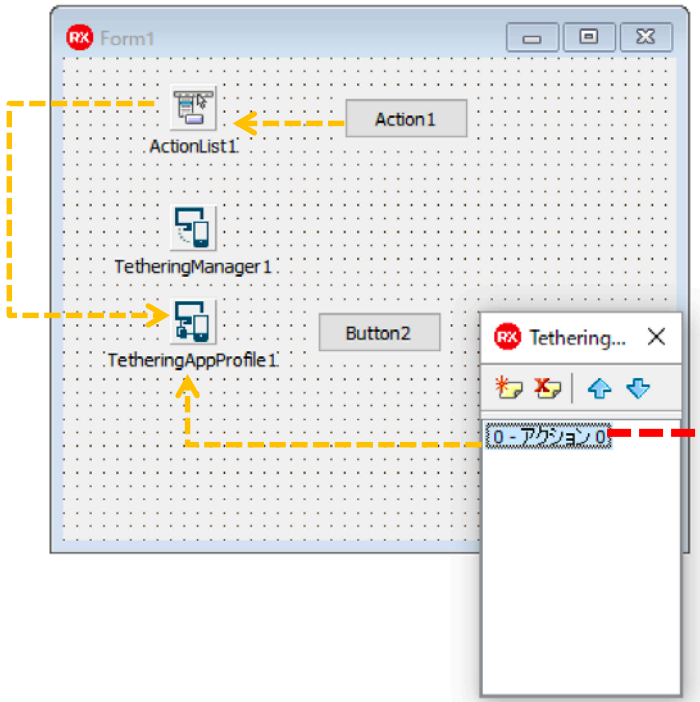


アプリケーション テザリング | TLocalAction

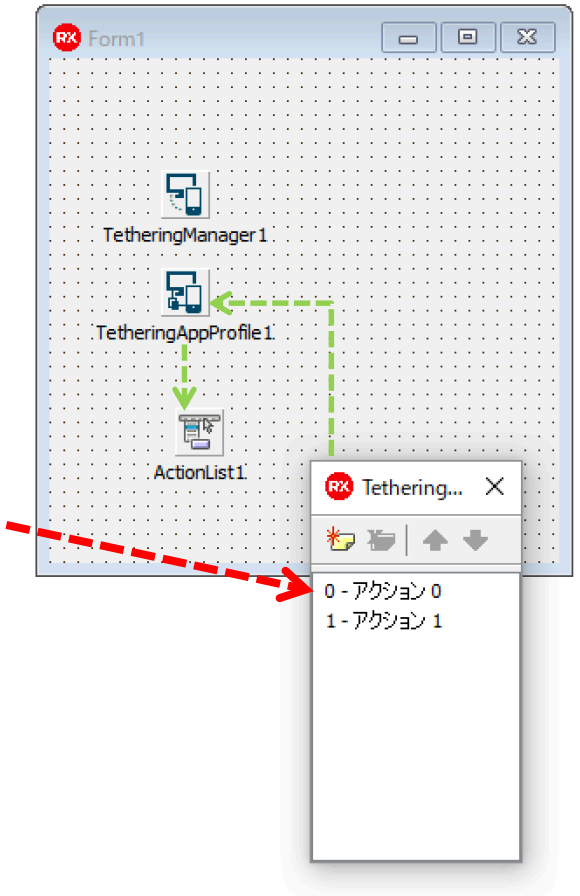
プロパティ	詳細
Action	
IsPublic	リモートプロファイルと共有する（True）否か（False）
Kind	Mirror(送信)/Shared(公開受信)
Name	一意に識別する文字列
NotifyUpdates	自動的に変更をアナウンスするか（True）、否か（False）を指定

アプリケーション テザリング | ActionList

送信側



受信側

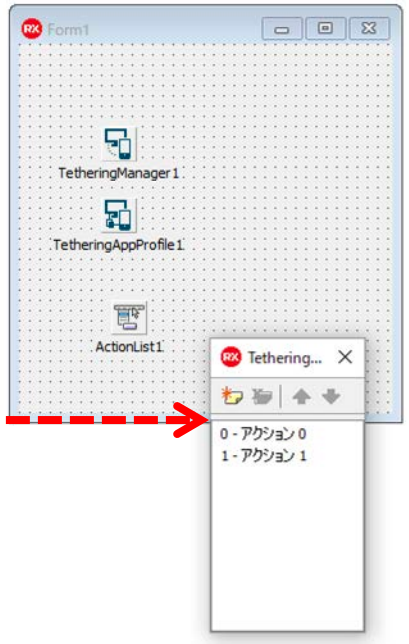


RunRemoteActionとRunRemoteActionAsync

- TTetheringAppProfile には RunRemoteAction というメソッドが用意されており、これを使用してリモート アクションを実行することができます。
  - (RunRemoteAction**Async**は非同期)

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    for (auto prof_:TetheringManager1->RemoteProfiles)
    {
        TetheringAppProfile1->RunRemoteAction(prof_, "b1");
    }
}
```

受信側



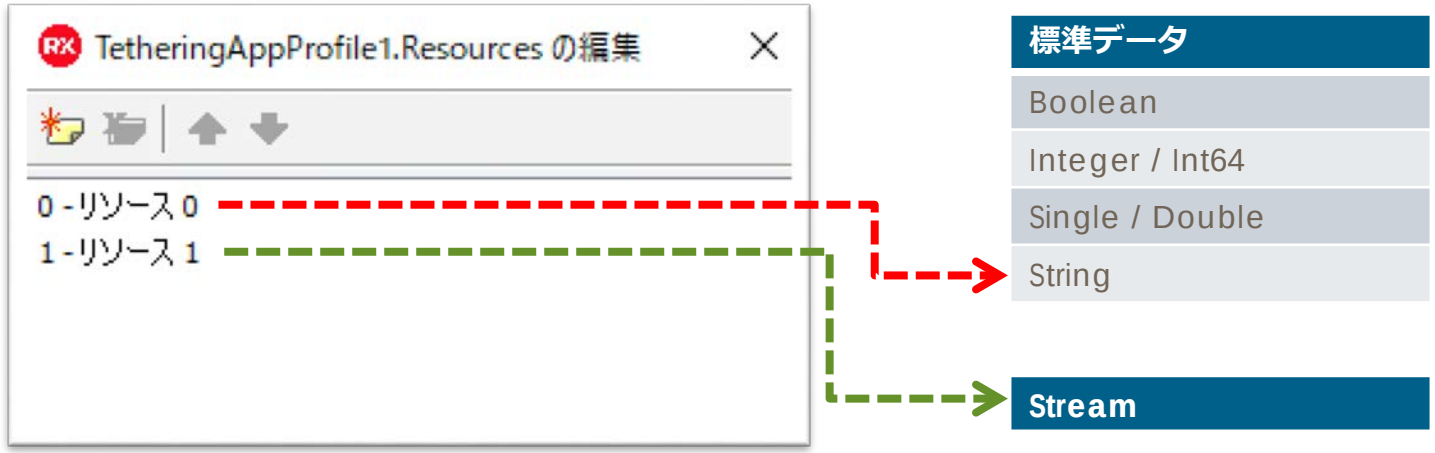
アプリケーション テザリング | データの共有と送信

- アプリケーション テザリングを使ったデータの送信は、以下のいずれかの方法で行うことができます

機能	詳細
共有リソースを定義する	リソースで値をラップします。接続されたアプリケーションは、共有リソースの値を読み取ったり、共有リソースをサブスクライブして共有リソースの値が変化したときに更新を受け取ることができます。
一時リソースとしてデータを送信する	実行時に文字列やストリームを送信し、それを、接続されたアプリケーションで一時リソースとして受け取ることができます。接続されたアプリケーションが一時リソースの更新をサブスクライブすることはできません。元のアプリケーションは一時リソースを一度しか送信しないからです。

アプリケーション テザリング | データの共有

- TLocalResource
  - 標準データ型（Boolean、Integer、Int64、Single、Double、String）と TStream をラップしています。
  - 接続されたアプリケーションとリソースを共有することができます。



アプリケーション テザリング | データの共有

- 送信側

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    TetheringManager1->AutoConnect();
    TetheringAppProfile1->Resources->FindByName("r1")->Value = "文字列";
}
```

- 受信側

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    TetheringManager1->AutoConnect();
    UnicodeString s_ = TetheringAppProfile1->Resources->FindByName("r1")->Value.AsString;
}
```

アプリケーション テザリング |リモート リソースを要求する

- TTetheringAppProfile コンポーネントには GetRemoteResourceValue というメソッドが用意されており、これを使用してリモート リソースを要求することができます。

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    TetheringManager1->AutoConnect();
    for (auto aProf_ : TetheringManager1->RemoteProfiles)
    {
        auto aResource_  = TetheringAppProfile1->GetRemoteResourceValue(aProf_, "r1");
        UnicodeString s_ = aResource_->Value.AsString;

        ShowMessage(s_);
    }
}
```

アプリケーション テザリング |リモート リソースを要求する

- GetRemoteProfileResources というメソッドが用意されています。このメソッドは、TTetheringProfileInfo のインスタンスとして指定されたリモート アプリケーション プロファイルが共有しているリモート リソースのリストを返します。

アプリケーション テザリング |データ送信

- 指定されたリモート プロファイルへ送信

メソッド名	説明
SendStream	指定されたストリームを指定されたリモート プロファイルへ送信し、成功した場合には True を、失敗した場合には False を返します。
SendString	指定された文字列を、指定されたリモート プロファイルに送信します。

アプリケーション テザリング |データ送信

- 送信側コード

```
void __fastcall TForm1::Button1Click(TObject *Sender) {
 TetheringAppProfile1->SendString(
     TetheringManager1->RemoteProfiles->Items[0], "Description", "送信データ");
}
```

- 受信側

```
void __fastcall TForm1::TetheringAppProfile1ResourceReceived(TObject * const Sender,
TRemoteResource * const AResource)
{
  ShowMessage(AResource->Value.AsString);
}
```

アプリケーション テザリング | 送信

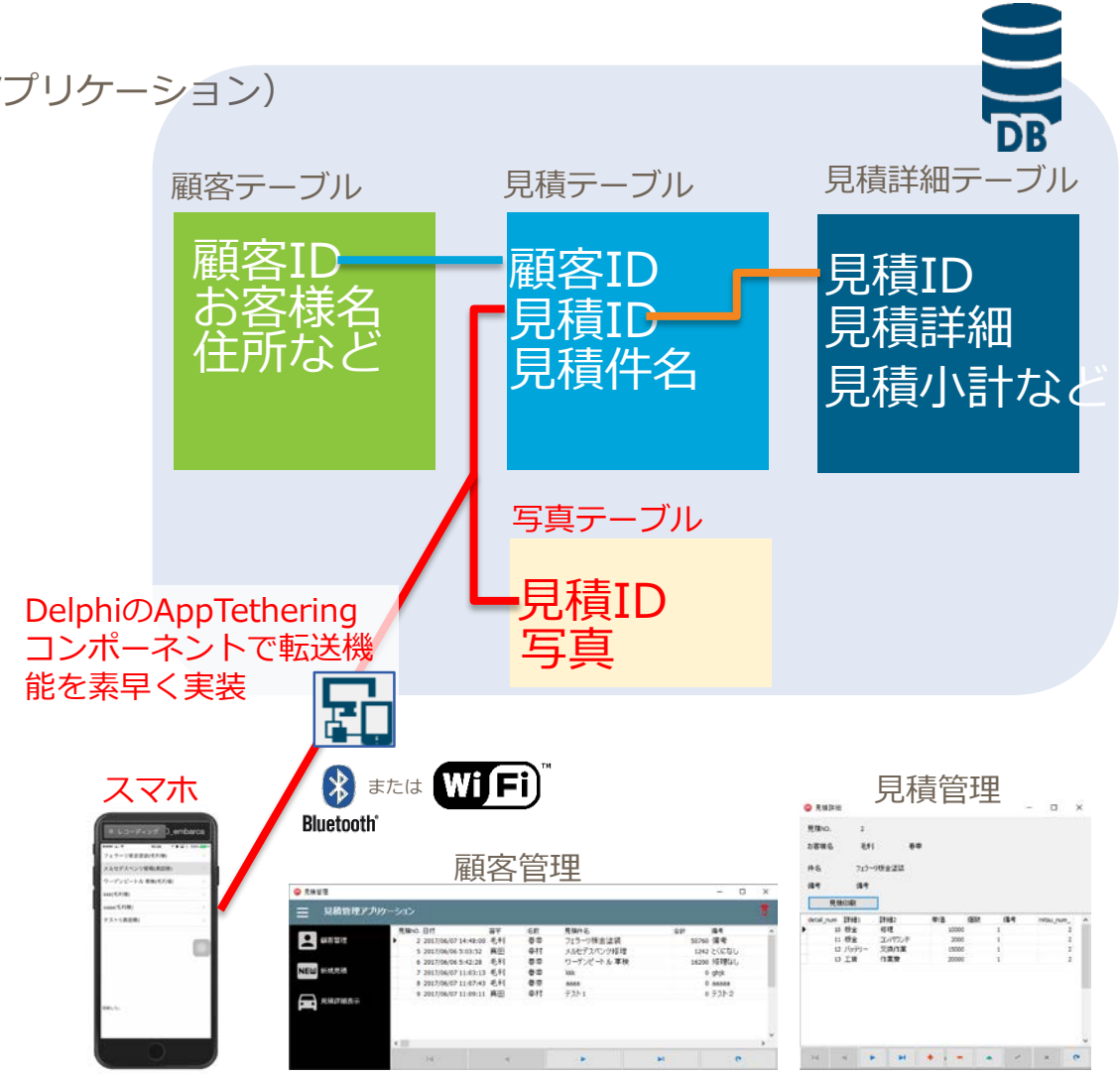
- リモート プロファイルの接続を使って、指定されたコマンドを送信

メソッド名	機能
SendCommand	指定された接続または指定されたリモート プロファイルの接続を使って、指定されたコマンドを送信します。
SendCommandWithResponse	指定されたコマンドを指定された接続または指定されたリモート プロファイルの接続を使って送信し、応答コマンドを受信するまで待機し、受信したコマンドを返します。

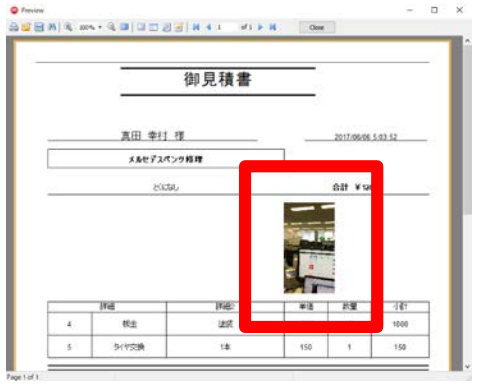
アプリケーション テザリングのモダナイゼーションまとめ

- 見積管理システムの画面（Delphiで構築したWindowsアプリケーション）
  - 顧客管理
  - 見積管理
  - 見積詳細(印刷)
  - 写真取得アプリ

- 使用機能等
  - データベース: InterBase
  - 印刷・帳票： FastReport
  - アプリ通信：AppTethering

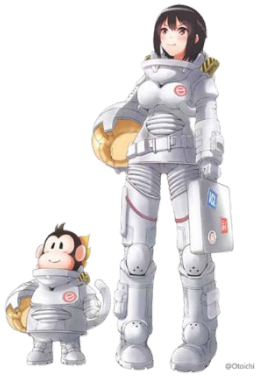


見積詳細印刷



多様なデータベースへの接続を実現

- RAD Serverを使った さまざまなDB接続
- 事例
  - 飲食店 在庫管理

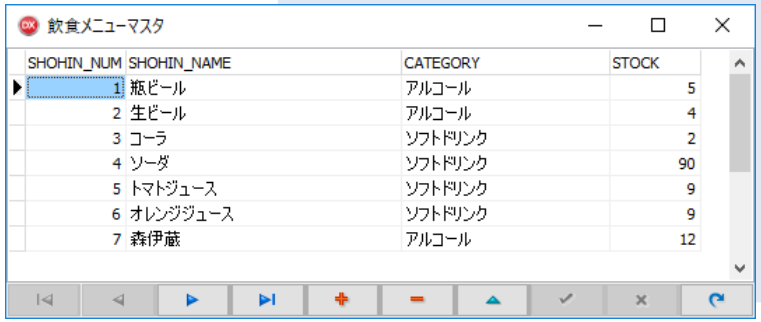


embarcadero DEVELOPER CAMP

業務拡大中 飲食店 在庫管理

- 飲食店 在庫管理アプリの画面等
  - 在庫テーブル表示・更新画面
- 使用機能等
  - データベース

在庫テーブル表示・更新画面



在庫テーブル

ID
商品名
カテゴリ
数量

飲食店 在庫管理アプリのモダナイゼーション

- 飲食店 在庫管理アプリの画面等
  - 在庫テーブル表示・更新画面
  - 在庫情報の取得、更新
  - バックヤード：Androidアプリ
  - 厨房：iPhoneアプリ
- 使用機能等
  - データベース：MySQL
  - Android / iOSアプリ：Delphi
  - 中間サーバー： RAD Server

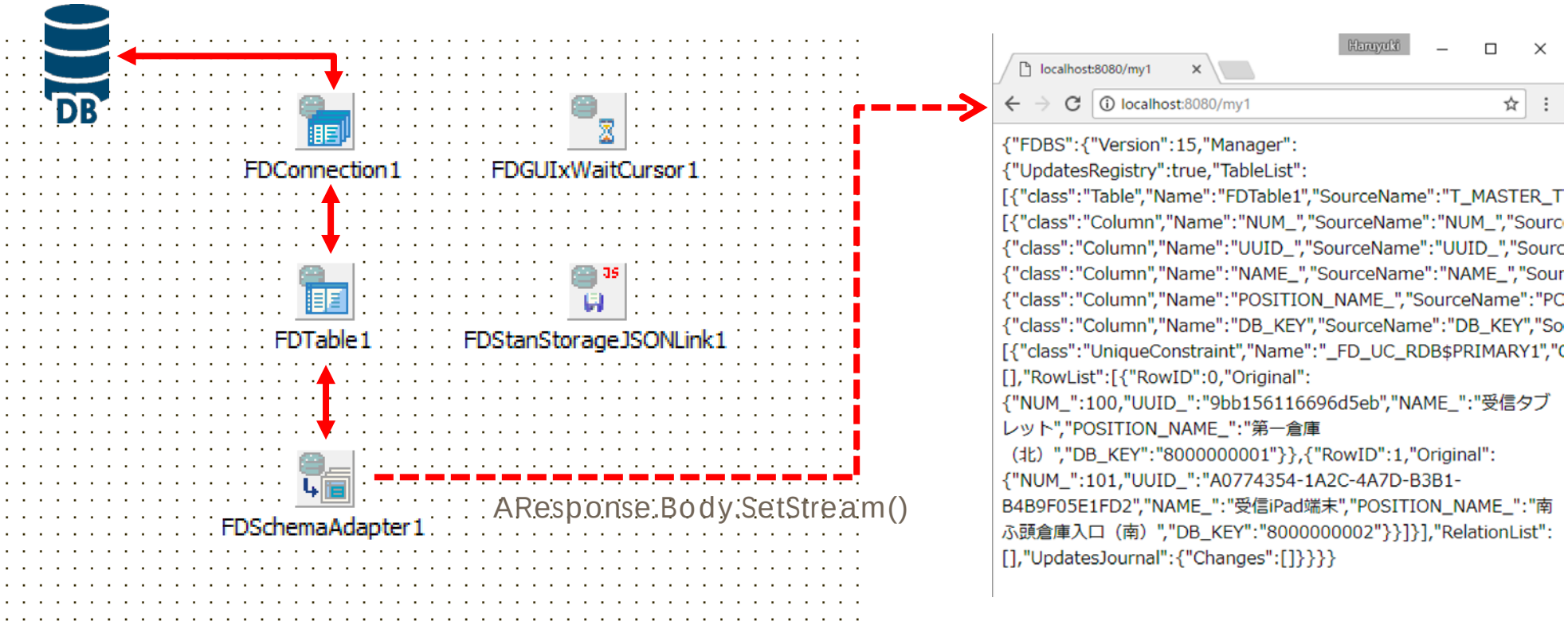
RAD Server 特徴

- ユーザーがクラウドや社内にホストできるモバイル エンタープライズ アプリケーション プラットフォーム（MEAP）
  - カスタム REST API やエンタープライズ データベース データを公開します。エンタープライズ データには、FireDAC データ アクセス ライブラリを使ってアクセスできます。

  - REST API
  - リモート データベース アクセス
  - ユーザーの追跡
  - アナリティクス

RAD Server側 プログラム

- TFDSchemaAdapter を使用すると、集中管理キャッシュ更新を管理することができます。



RAD Server側 プログラム

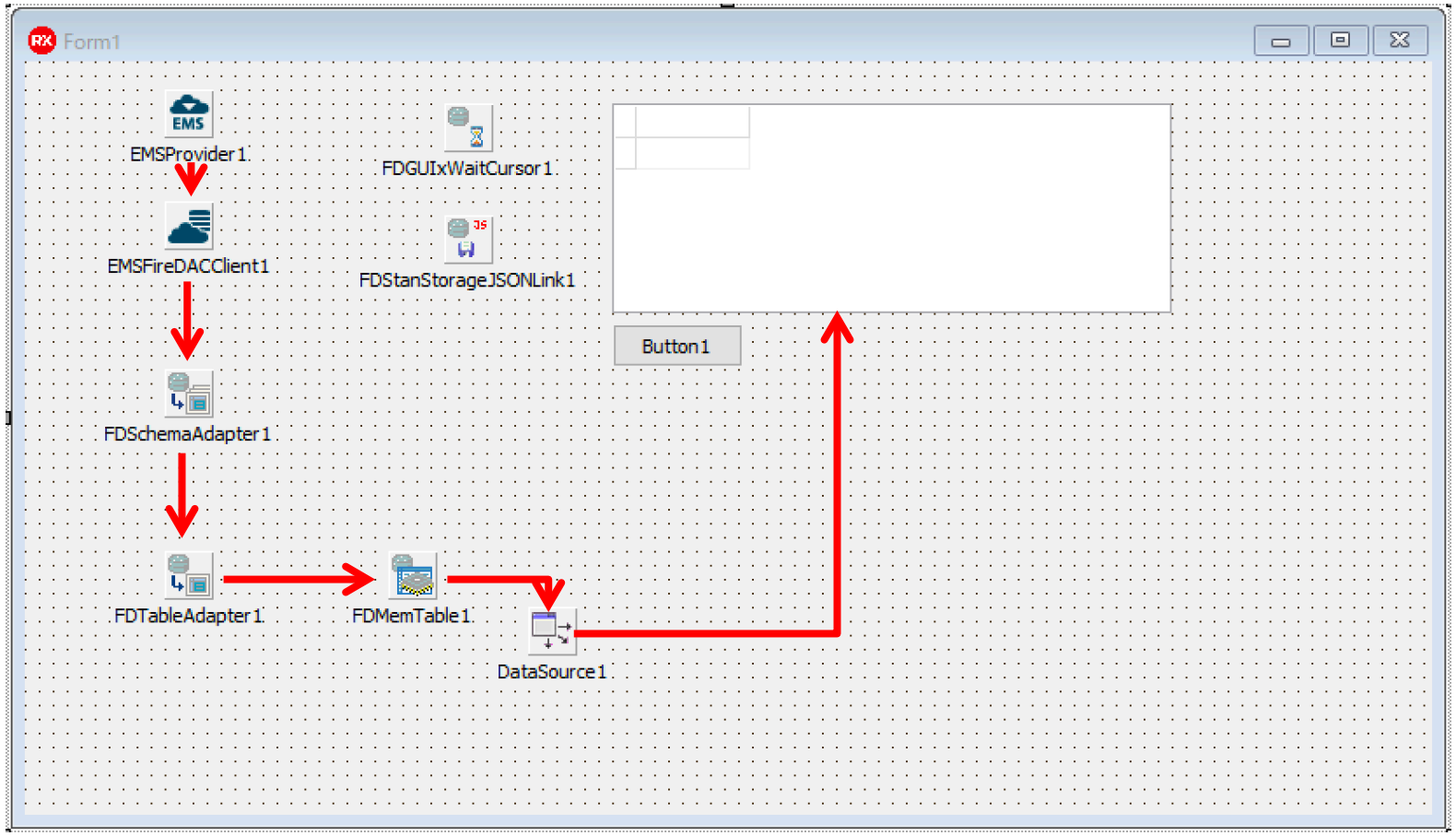
- データセットを共有する為のコード

```
procedure TMyResource1.Get(const AContext: TEndpointContext; const ARequest: TEndpointRequest;
const AResponse: TEndpointResponse);
var
  ms: TMemoryStream;
begin
  ms  := TMemoryStream.Create;
  try
    FDTable1.Active := True;
    FDSchemaAdapter1.SaveToStream(ms, TFDStorageFormat.sfJSON);
    AResponse.Body.SetStream(ms, 'application/json', True);
  finally

  end;
end;
```

RAD Serverからのクライアント側

- クライアント側はコンポーネントの配置のみです



RAD Serverからのクライアント側コード

- RAD ServerからTFDSchemaAdapterのデータ取得の際に1行コードが必要です。

```
procedure TForm1.Button1Click(Sender: TObject);
Begin
  EMSFireDACClient1.GetData;
end;
```

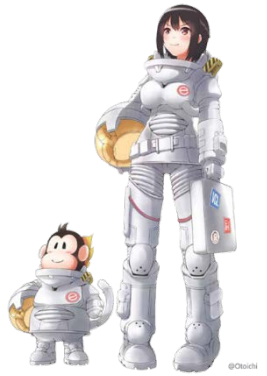
※ FDTableAdapter1.DatSTableNameは TFDTable::Nameです。

飲食店 在庫管理アプリのモダナイゼーション

- 飲食店 在庫管理アプリの画面等
  - 在庫テーブル表示・更新画面
  - 在庫情報の取得、更新
  - バックヤード：Androidアプリ
  - 厨房：iPhoneアプリ
- 使用機能等
  - データベース：MySQL
  - Android / iOSアプリ：Delphi
  - 中間サーバー： RAD Server

クラウド接続

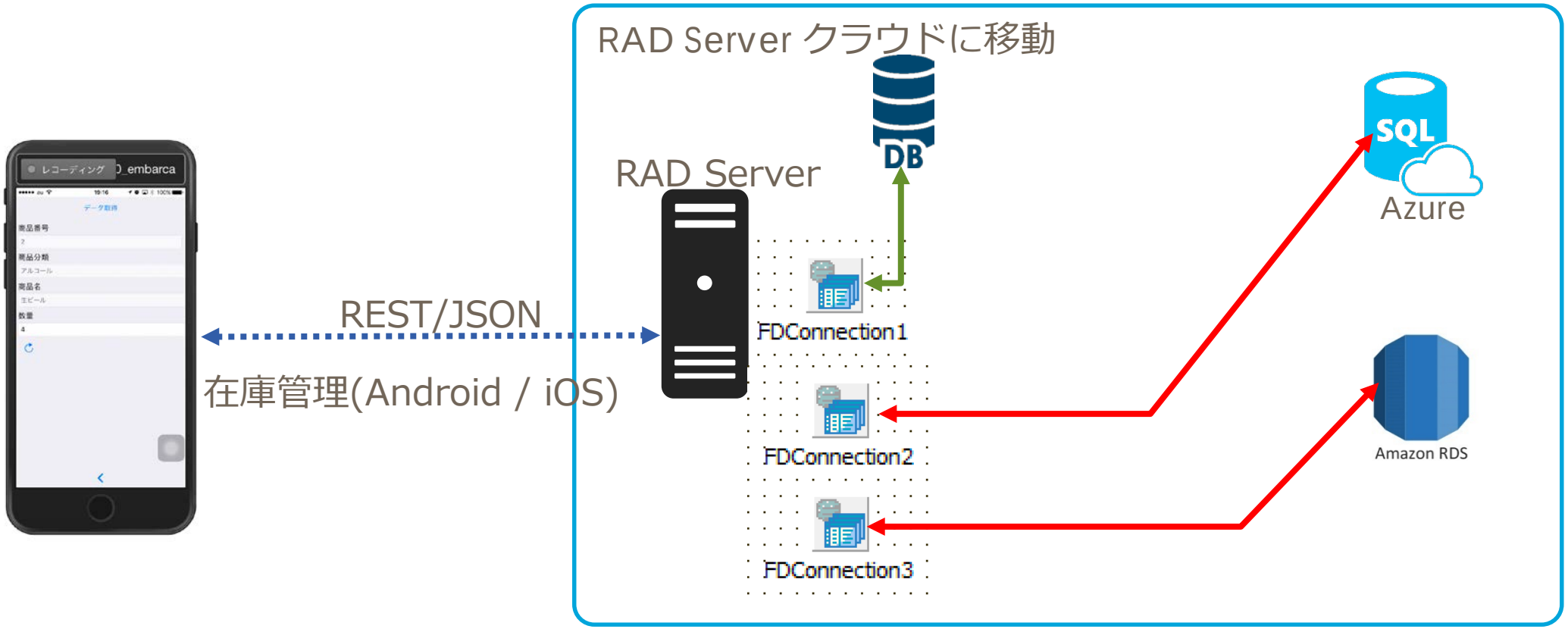
- RAD Server + FireDACを利用した クラウド接続多様性
- Amazon Simple Storage Service はインターネット用のストレージサービスです。



embarcadero DEVELOPER CAMP

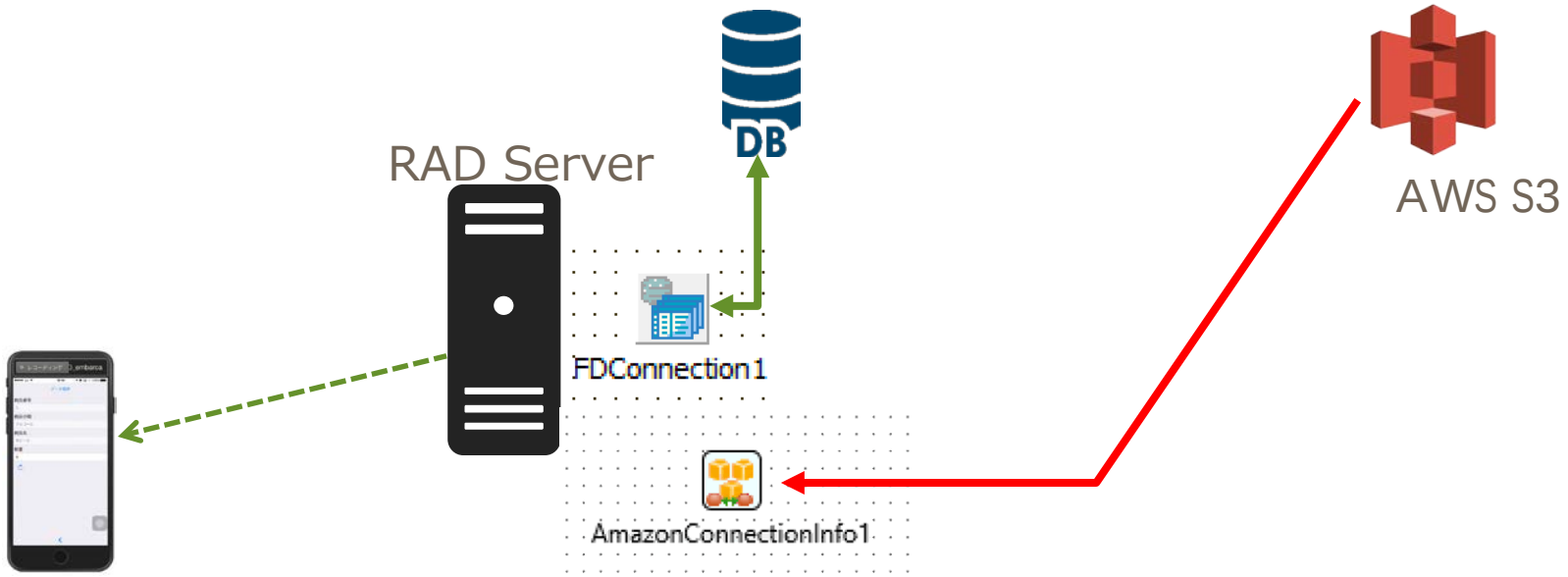
RAD Server FireDACを利用した クラウドDB接続

- RAD Server TFDConnection接続先をクラウドへ



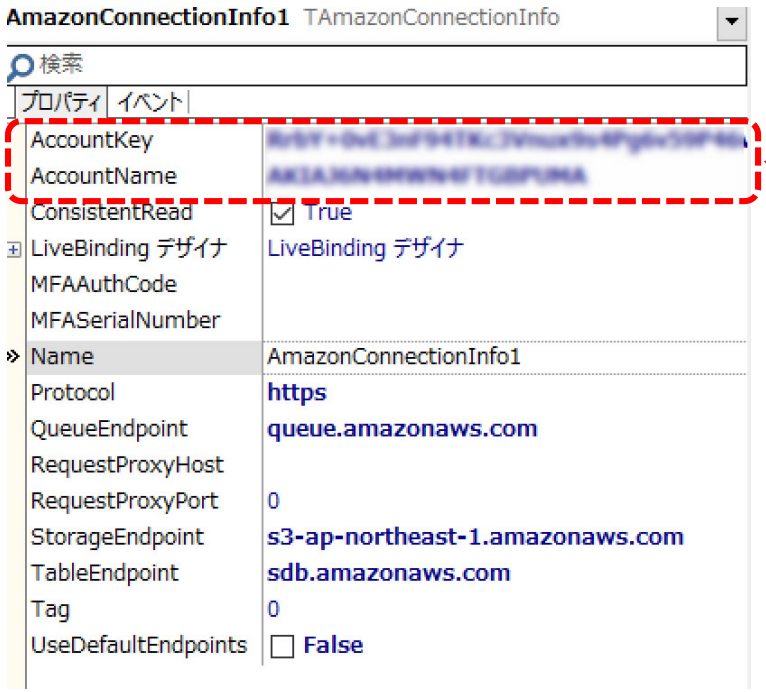
RAD Server を利用した クラウド接続多様性

- RAD Server TAmazonConnectionInfoでクラウドストレージへ



クラウド AWS S3接続

- Delphi/C++ Builderは AWS接続のための TAmazonConnectionInfoコンポーネントが用意されています



IAM(Identity and Access Management)

クラウド AWS S3接続

- S3接続側コード

```
function TResourceResource1.aws_s3(fname_: String): TStream;
var
  s3_:    TAmazonStorageService; res_:   TCloudResponseInfo; mm_:    TMemoryStream;
begin
  Result  := nil;
  AmazonConnectionInfo1.StorageEndpoint := 's3-ap-northeast-1.amazonaws.com';
  s3_   := TAmazonStorageService.Create(AmazonConnectionInfo1);
  res_  := TCloudResponseInfo.Create;
  try
    try
      mm_   := TMemoryStream.Create;
      s3_.GetObject('embarcader',fname_, mm_, res_);
      Result  := mm_;
    except
    end;
  finally
    res_.DisposeOf;
    s3_.DisposeOf;
  end;
end;
```

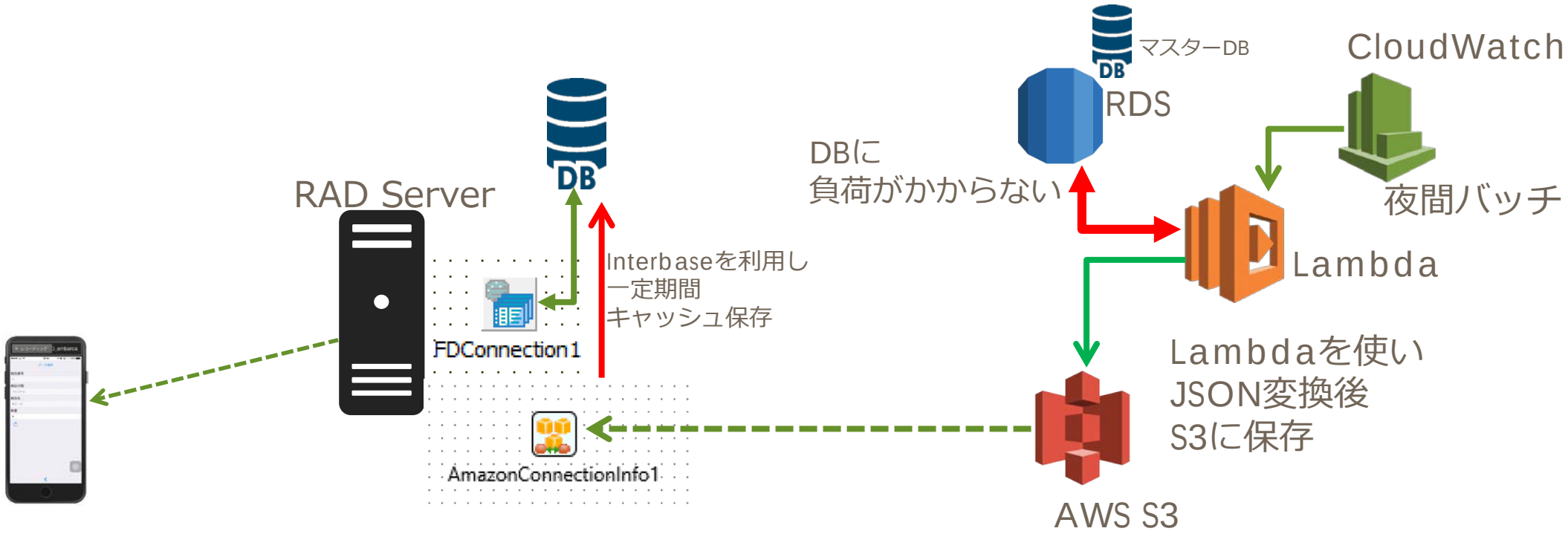
クラウド AWS S3接続

- S3からデータ取得後 SetStream()

```
procedure TResourceResource1.Get(const AContext: TEndpointContext; const ARequest:
TEndpointRequest; const AResponse: TEndpointResponse);
var
  mm_:  TMemoryStream;
  fs_:  TFileStream;
begin
  // Sample code
  mm_ := aws_s3('amane-phone.jpg') as TMemoryStream;
  if Assigned( mm_ )  then
  begin
    mm_.Position  := 0;
    AResponse.Body.SetStream(mm_,'image/jpeg', True);
  end;
end;
```

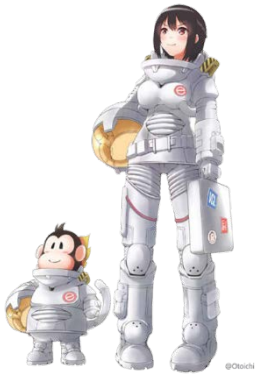
RAD Server を利用した クラウド接続多様性

- RAD Server TAmazonConnectionInfoでクラウドストレージへ



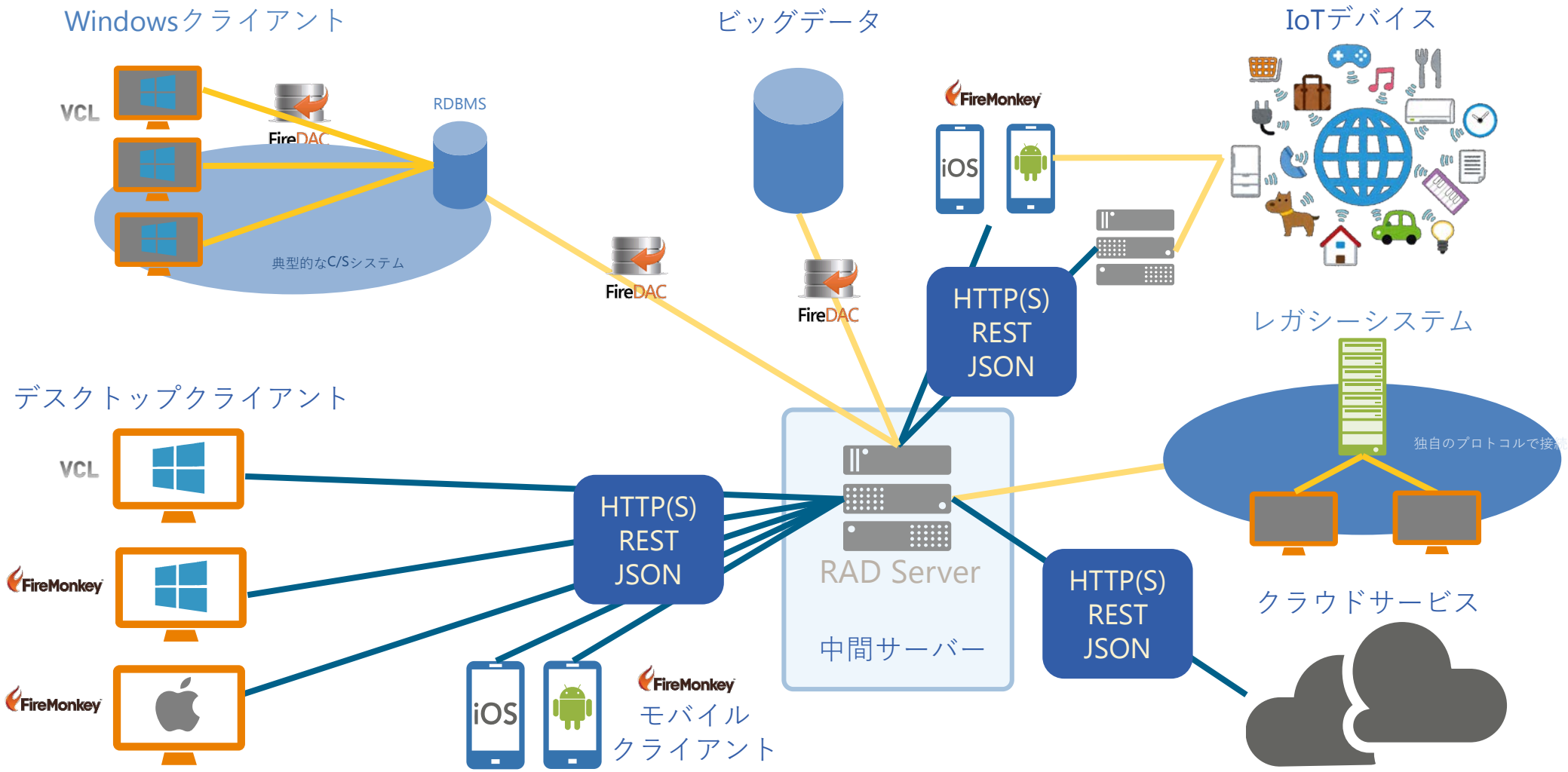
最終段階

- 新たにモバイル向けのフルシステムを組むのではなく、今までのVCLアプリと連動させたシンプルなモバイル化の例をファーストステップとして、多様なデータベース、クラウド接続へと展開



embarcadero DEVELOPER CAMP

最終段階



まとめ

- コンポーネントによって簡単にモバイル
  - Application Tetheringコンポーネント
- 多様なデータベースへの接続を実現
  - 既存のシステム + RAD Serverを用いる事でのモバイルDB接続
- クラウド接続
  - FireDACを利用した クラウドDB接続

THANKS!

www.embarcadero.com/jp

第34回 エンバカデロ・デベロッパーキャンプ