【セッションNo. 3】

知って得する！現役ヘルプデスクが答えるDelphiテクニカルエッセンス5.0

株式会社ミガロ.
RAD事業部 技術支援課
吉原 泰介

【アジェンダ】

Q1.サイレントインストールを行うには？

Q2.Flash動画を扱うには？

Q3.AnsiStringとUnicodeStringの違いは？

Q4.複数アプリ間でパラメータを受け渡すには？

Q1.サイレントインストールを行うには？

【質問】

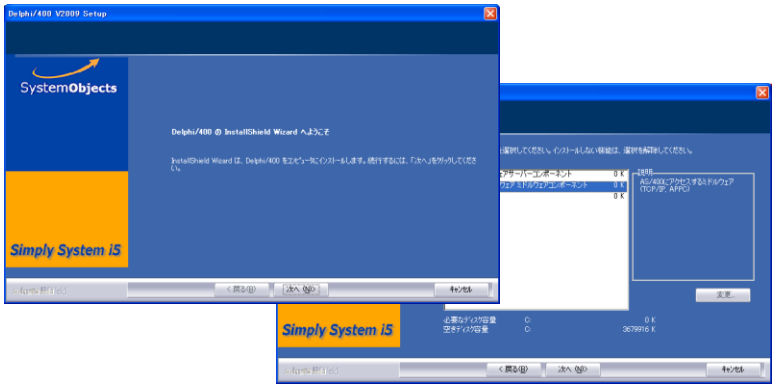
サイレントインストールとはどうやって行うのでしょうか？

【回答】

サイレントインストールとは、インストールのダイアログを使用せずにコマンド実行からバッチ的にインストールを行う方法です。
Delphi/400ではメンテナンスサービスにてサイレントインストール用のファイルや手順をご提供しております。

■サイレントインストールとは？

- サイレントインストールとは、インストールダイアログが表示されずユーザーからの入力を必要としないインストール方法です。

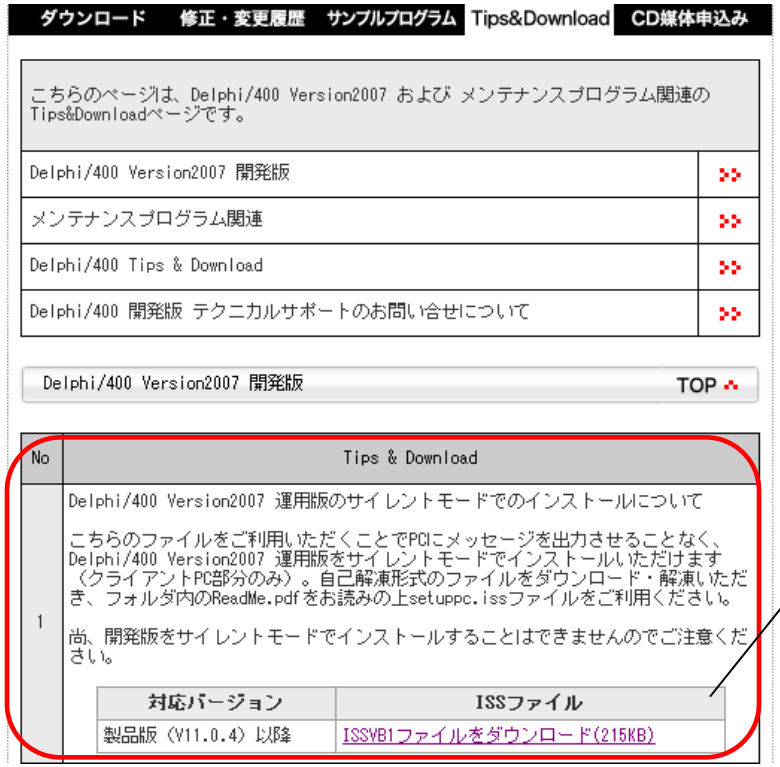


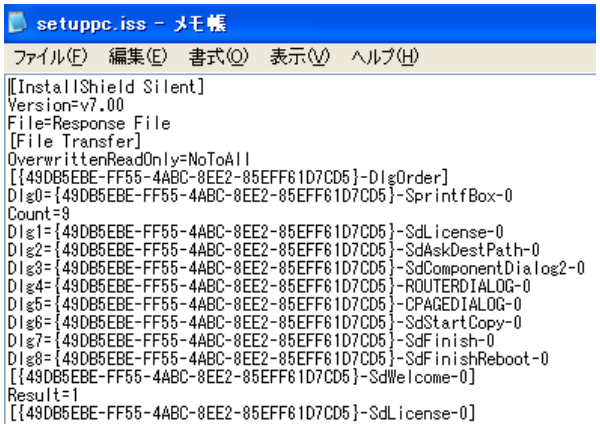
■ダイアログを省略する利点
・ユーザーにインストール操作をさせない。
・クライアント台数の多い場合インストール作業を省略する。

- Delphi/400 サイレントインストール前提
  ・インストール対象はDelphi/400運用版
  ・BDEなどが必要な環境がインストールされている

■応答ファイル(ISSファイル)

- ダイアログ応答をしない代わりに応答ファイル(ISSファイル)を使用してインストールが行われます。



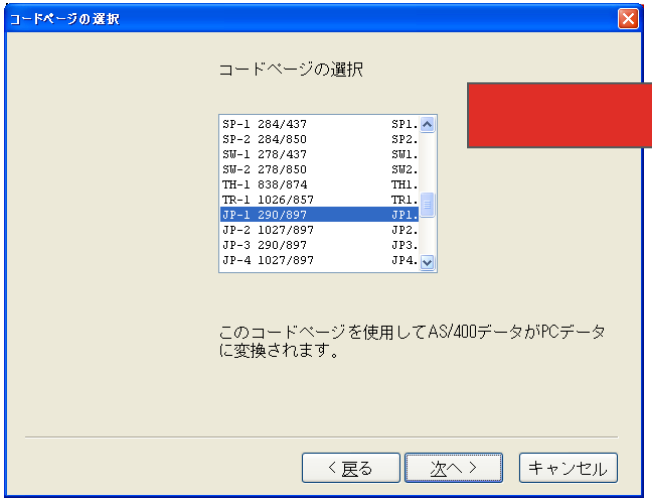


※ISSファイルについては
Delphi/400メンテナンスサービスで
バージョンごとに専用ファイルを
ご提供をさせて頂いております。

■応答ファイル(ISSファイル)の編集

- ISSファイルのセクションごとの設定を編集することで応答内容を調整することができます。

コードページの応答



[CPAGEDIALOG-0]セクション

```
[CPAGEDIALOG-0]
EBCDICFILE=JP1.EBC
ASCIIFILE=JP1.ASC
Result=1
```

ディフォルトはJP-1となっていますが、
JP-2 をご使用の場合は
EBCDICFILE=JP2.EBC およびASCIIFILE=JP2.ASC に
といった形で指定できます。

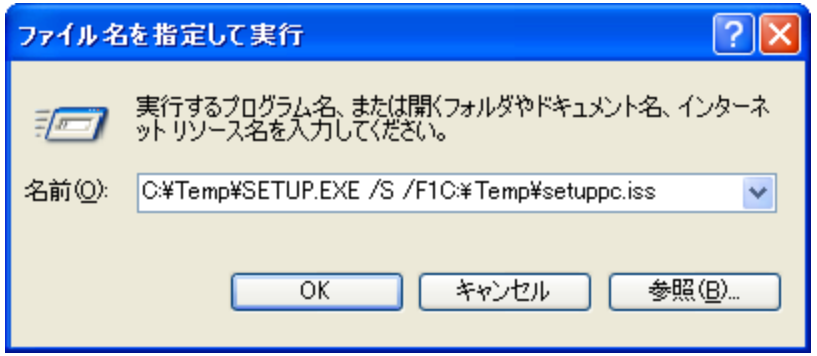
■実行方法

- サイレントインストールは「コマンドプロンプト」や「ファイル名を指定して実行」から実行できます。

例）C:ドライブのTempフォルダにsetuppc.iss ファイルとDelphi/400 運用版のSETUP.EXEを置いて実行する場合
C:¥Temp¥SETUP.EXE /S /F1C:¥Temp¥setuppc.iss

【実行パラメータ】

/S	サイレントモード
/F1	実行に使うファイル指定



■実行結果ファイル(setup.log)

- サイレントインストールを行うとISSファイルと同じフォルダに実行結果(setup.log)ファイルが作成されます。

実行結果(setup.log)

```
[InstallShield Silent]
Version=v7.00
File=Log File
[ResponseResult]
ResultCode=0
[Application]
Name=Delphi/400
Version=12.0
Company=TCIS
Lang=0011
```

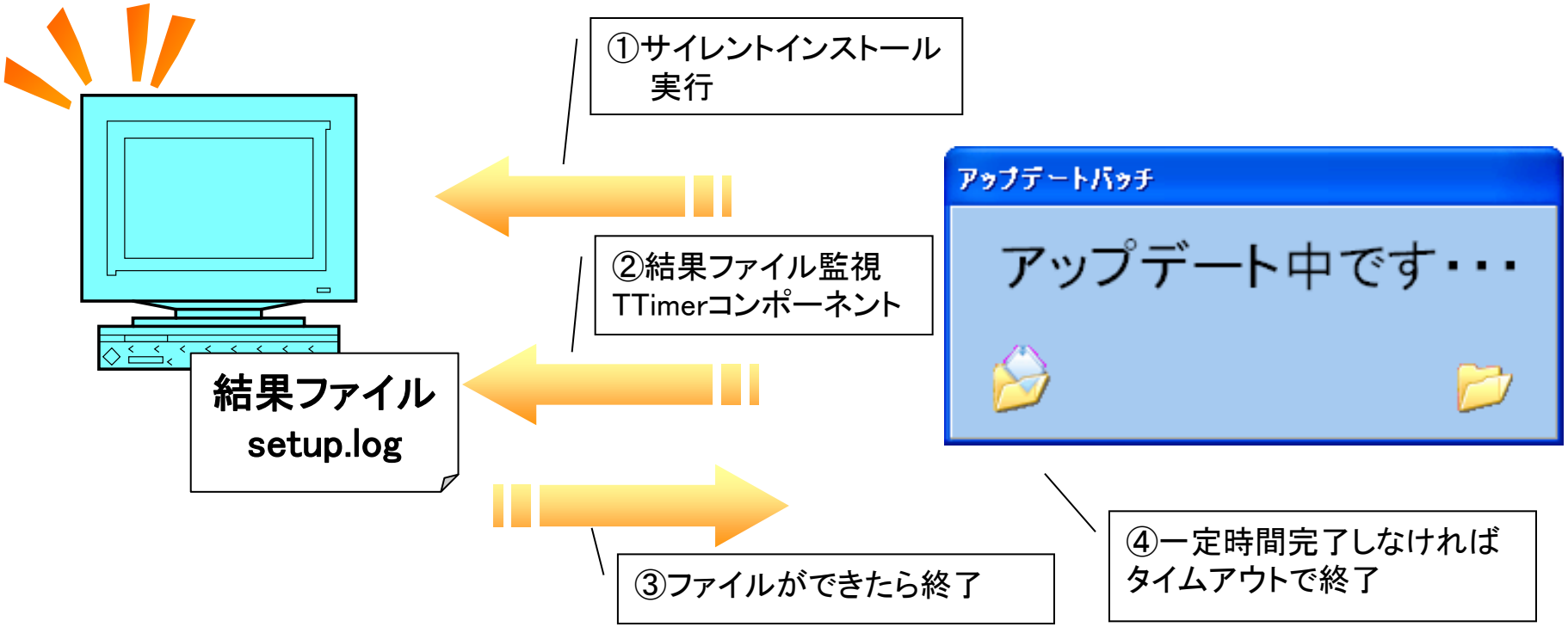
【*ResultCode*】
ResultCodeが0であればインストールが
正常終了

ResultCodeが0以外であればインストールが
異常終了
→ ISSファイルの格納場所等に間違いがないかどうか再度ご確認ください

インストールされたDelphi/400の情報

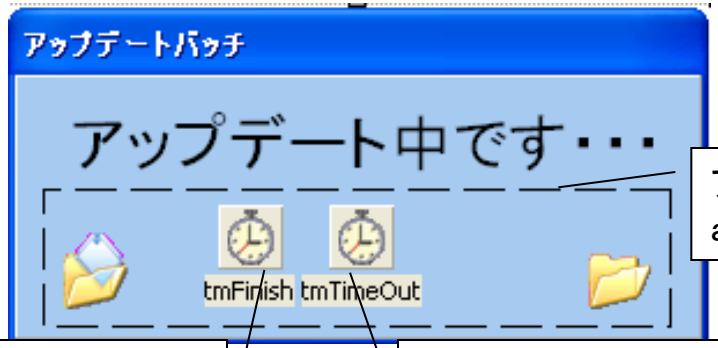
■Delphi/400アップデートアプリを作ってみよう

- サイレントインストールを行うバッチアプリを作成して Delphi/400を簡単にアップデートをできるようにする。



■Delphi/400アップデートアプリを作ってみよう

● 画面使用コンポーネント



アップデートアニメ用
anUpdate (TAnimate)

結果ファイル監視用
tmFinish(TTimer)

タイムアウト用
tmTimeOut(TTimer)

【TTimerコンポーネント】
Intervalプロパティに時間設定して
定期的にイベントを実行することが
できます。

● 変数宣言

```
private
    { Private 宣言 }
    //アプリケーション実行パス、ISSファイルパス、SetUp.exeパス、setup.logパス
    sCPath,sISSPath,logPath,sD400Path : String;
```

■Delphi/400アップデートアプリを作ってみよう

● ①サイレントインストール実行

```
procedure TForm1.FormShow(Sender: TObject);
var
  SI: TStartupInfo;
  PI: TProcessInformation;
begin
  sCPath      := ExtractFilePath(Application.ExeName);      //アプリケーション実行パスの取得
  sISSPath    := sCPath + 'setuppc.iss';                    //ISSファイルパス決定
  logPath     := sCPath + 'setup.log';                      //setip.logのファイルパス決定
  //Setup.exeのファイルパス決定
  sD400Path   := sCPath + 'SETUP.EXE';

  //Delphi/400 Setup.exeの存在確認
  if not(FileExists(sD400Path)) then
  begin
    //エラーメッセージ
    showmessage('フォルダ内にDelphi/400のSETUP.EXEがありません。');
    //終了
    Close;
  end;
```

SETUP.EXEが別の場所にある場合は
別のパスを設定する

■Delphi/400アップデートアプリを作ってみよう

● ①サイレントインストール実行

```
//ISSファイルの存在確認
if not(FileExists(sISSPath)) then
begin
  //エラーメッセージ
  showmessage('フォルダ内にDelphi/400のsetuppc.issがありません。');
  //終了
  Close;
end;
//結果監視タイマー開始
tmFinish.Enabled  := True;
//タイムアウトタイマー開始
tmTimeOut.Enabled  := True;
//setip.logファイルを削除
DeleteFile(logPath);
//描画調整
Application.ProcessMessages;
//サイレントインストールの命令コマンドを実行
GetStartupInfo(SI);
CreateProcess(nil, PChar('"' + sD400Path + '" /S /F1"' + sISSPath + '"'),
              nil, nil, False, CREATE_DEFAULT_ERROR_MODE, nil, nil, SI, PI);
end;
```

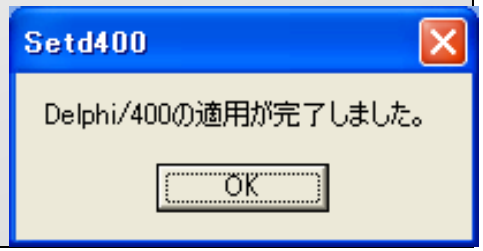
前回setup.logが残っている場合
監視ができないので事前に削除

<SETUPEXEパス> /S/F1<ISSファイルパス>
注意:パスをダブルクオーテーション""で括らないと
パスの内容によっては正しくパスを扱えません

■Delphi/400アップデートアプリを作ってみよう

● ②結果監視～③ファイルができたら終了

```
procedure TForm1.tmFinishTimer(Sender: TObject);
begin
 //setip.logファイルが存在するか監視
  if (FileExists(logPath)) then
  begin
    //setup.logファイルが生成されたら
    //タイムアウトタイマー終了
    tmTimeOut.Enabled  := False;
    //完了メッセージ
    Showmessage('Delphi/400の適用が完了しました。');
    //終了
    Close;
  end;
end;
```

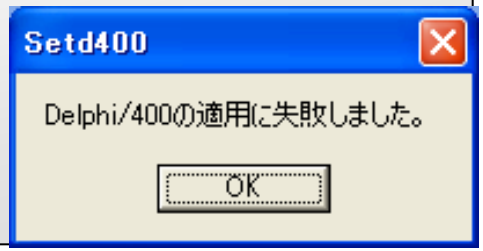


結果を詳しく判定するのであれば
Setup.logの Resultcodeを読み込んで成功/失敗を判定する

■Delphi/400アップデートアプリを作ってみよう

- ④タイムアウト制御
  インストール処理が何らかの理由で完了しない場合
  一定時間たつとアプリケーションを終了させる

```
procedure TForm1.tmTimeOutTimer(Sender: TObject);
begin
  //タイムアウト時間が過ぎたらエラーメッセージ
  Showmessage('Delphi/400の適用に失敗しました。');
  //終了
  Close;
end;
```



PC初回インストール用のバッチアプリを作成する場合は
BDE&アプリのインストールやAliases.cfg上書きなどの
拡張が必要です。

Q2.Flash動画を扱うには？

【質問】

Delphi/400のアプリケーション上でFlash動画を表示したいのですが可能ですか？

【回答】

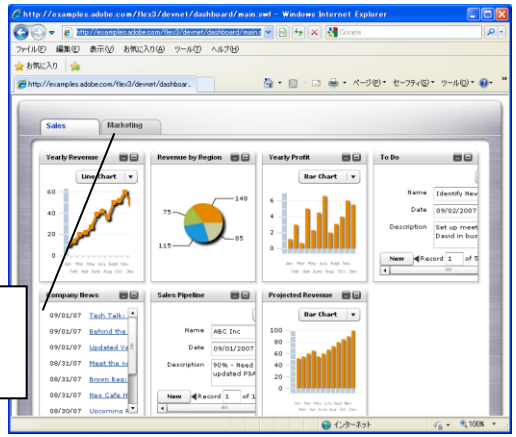
FlashはshockwaveFlashというActiveXとして機能が提供されています。
Delphi/400ではActiveXコントロールの取り込みができるので、それによる実装で実現することが可能です。

■Flashとは

- Web上で画像や動画、音楽がコンテンツとして組み込まれることが多くなりましたが、こういったWeb技術として広く使われるようになったのが「Flash」です。

HPの動画コンテンツなどは
ほとんどFlashを使用しています

動画公開で有名な「You Tube」
Flashで動画を使用しています

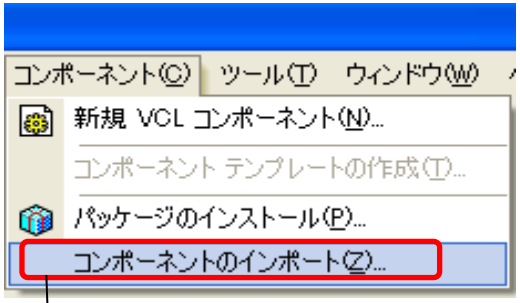


ActionScriptなどを使う「Flex」
もFlashを使用しています

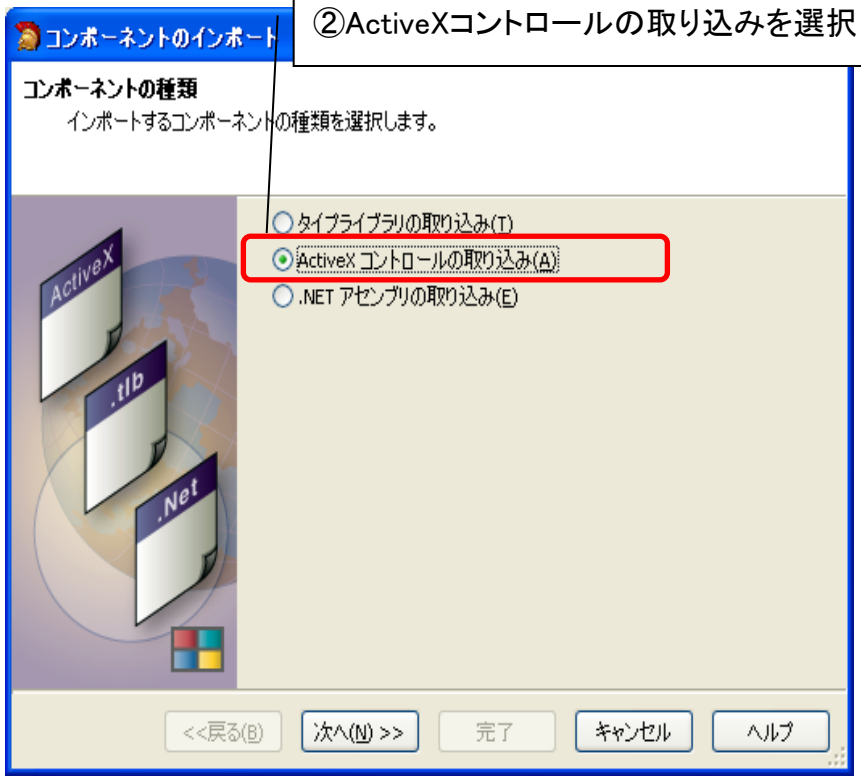
■ActiveXコントロール　Shockwave Flashを取り込む

- FlashをDelphi/400上で利用するためには
  ActiveXコントロールを利用します。

**ActiveXコントロール
取り込み手順**

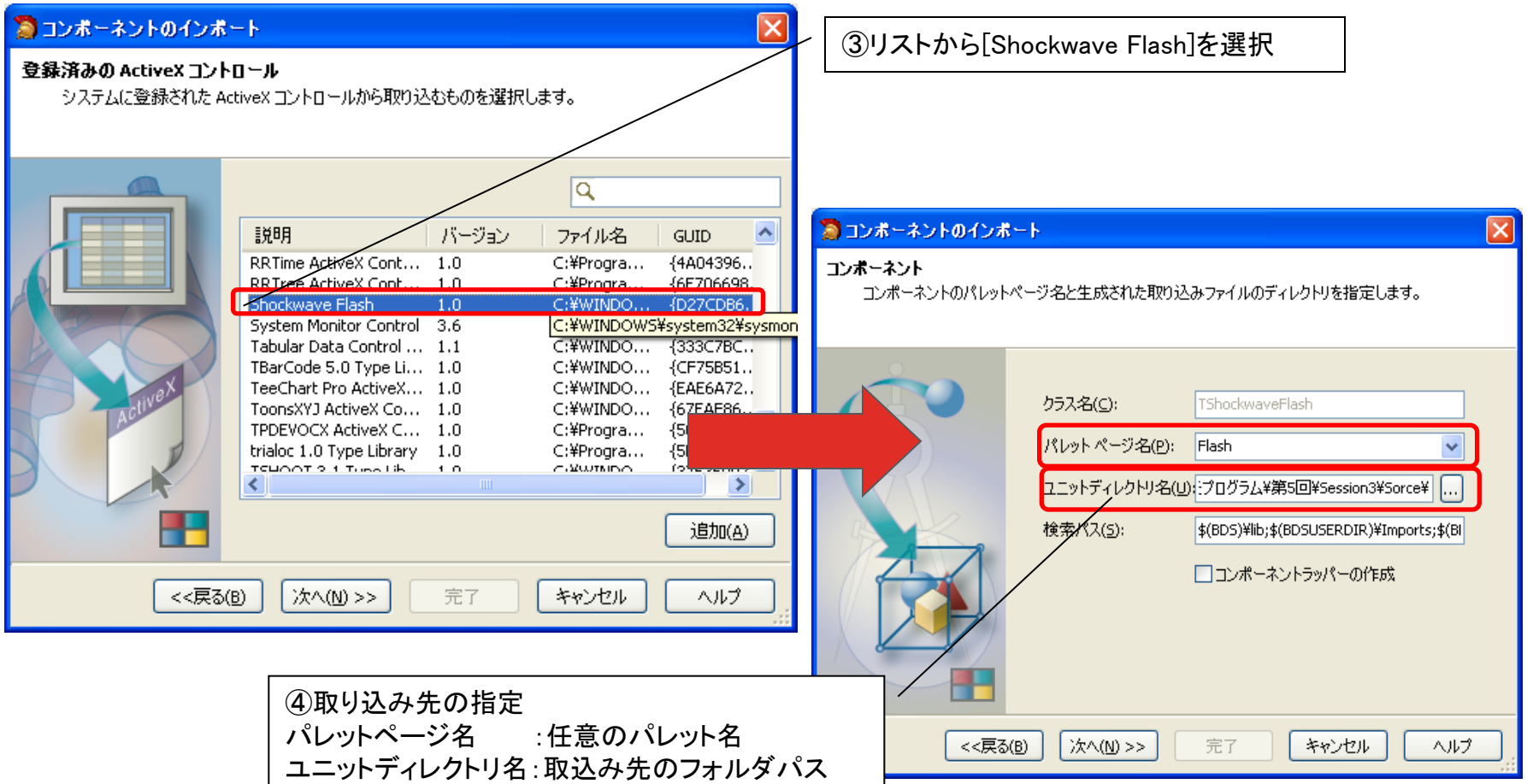


①[コンポーネント]
→ [コンポーネントのインポート]
からウィザードを起動

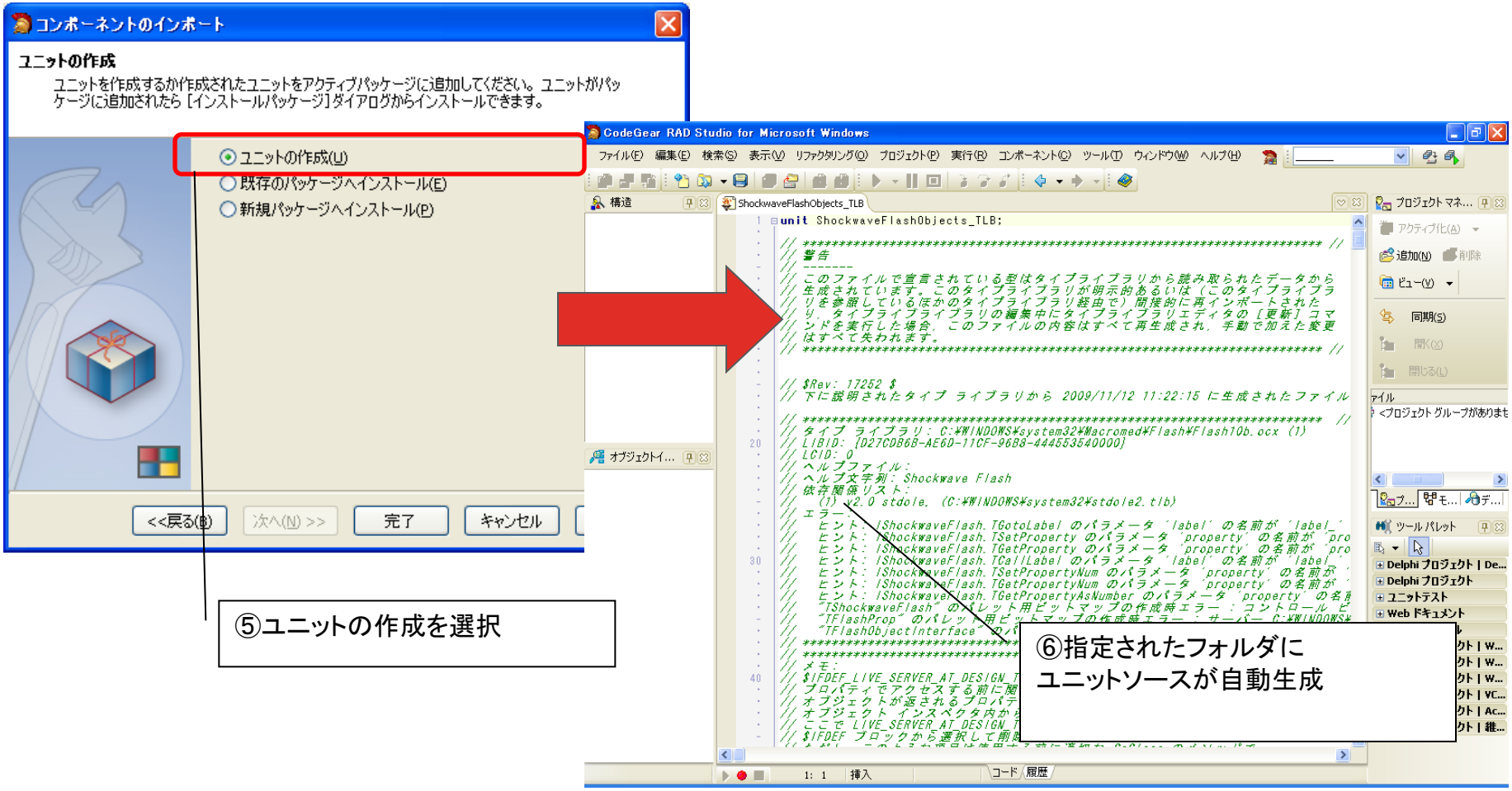


②ActiveXコントロールの取り込みを選択

■ActiveXコントロール　Shockwave Flashを取り込む

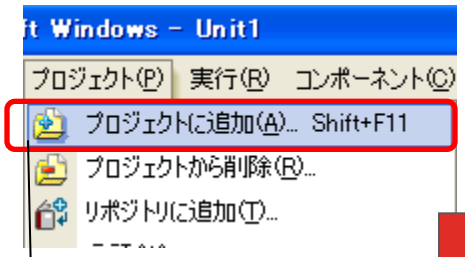


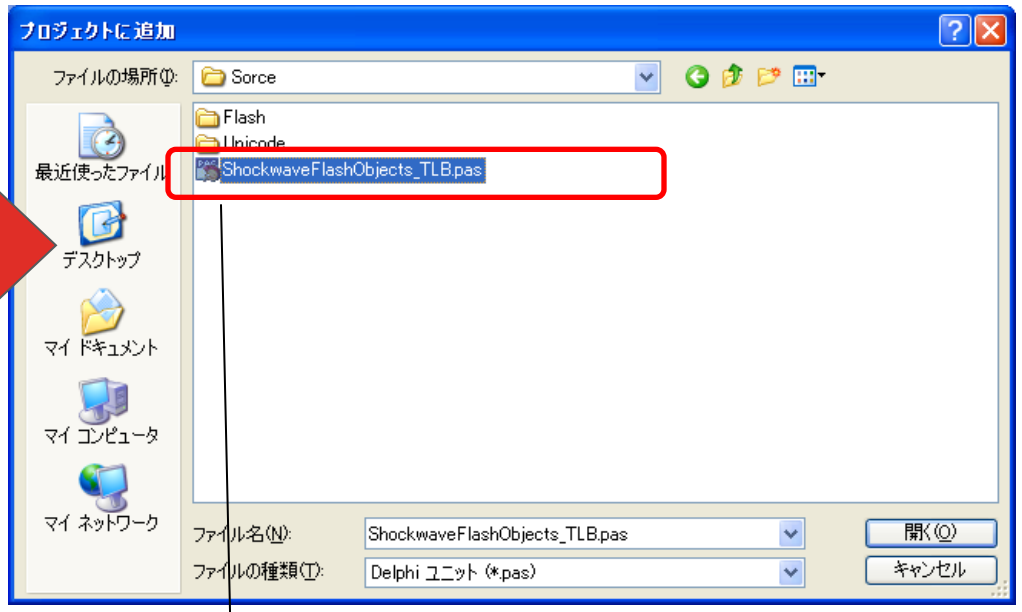
■ActiveXコントロール　Shockwave Flashを取り込む



■Shockwave Flashをプロジェクトに組み込む

自動生成されたユニットを組み込む手順

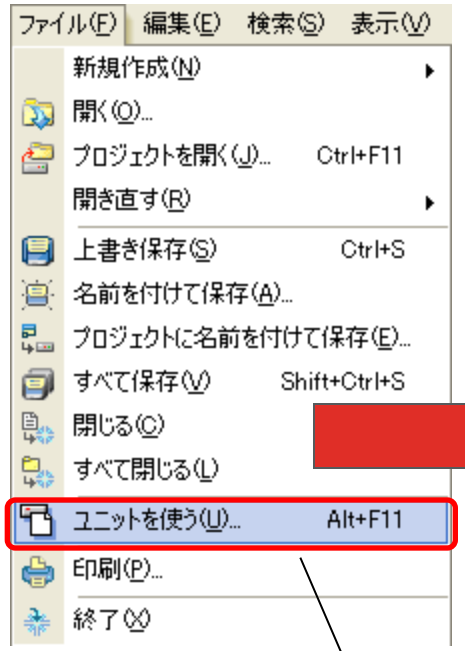


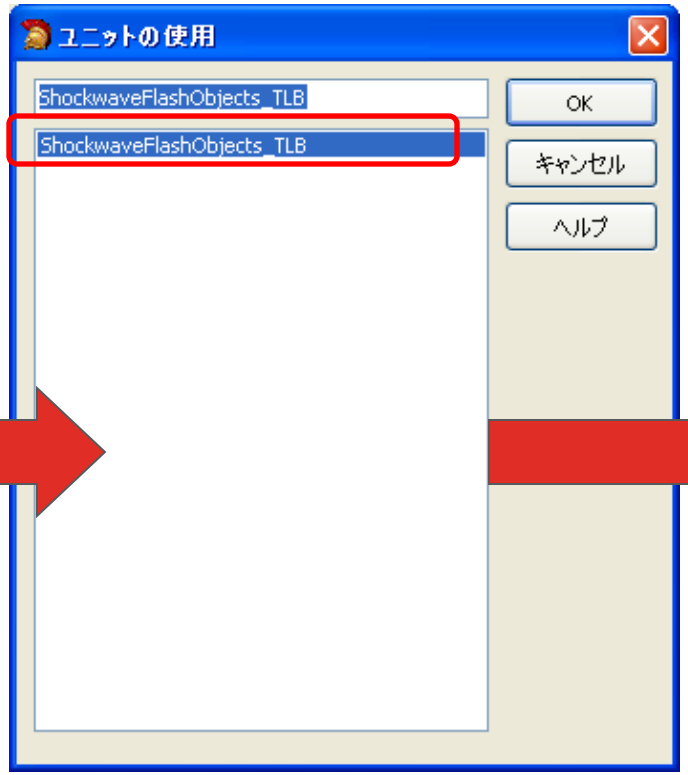


⑦使用したいプロジェクトで
生成ファイルを追加

⑧自動生成されたユニットソースを指定
ShockwaveFlashObjects_TLB

■Shockwave Flashをプロジェクトに組み込む






```
var
  Form1: TForm1;

implementation

uses ShockwaveFlashObjects_TLB;

{$R *.dfm}

end.
```

⑨ユニットを使うで
ShockwaveFlashObjects_TBL
を選択

⑩プログラムのuses節に
ShockwaveFlashObjects_TLB
が追加される

ShockwaveFlashObjects_TLBが提供する機能

TShockwaveFlashクラス　・・・自動生成ユニットが提供

Movieプロパティ

再生するFlash Movieを指定するプロパティ

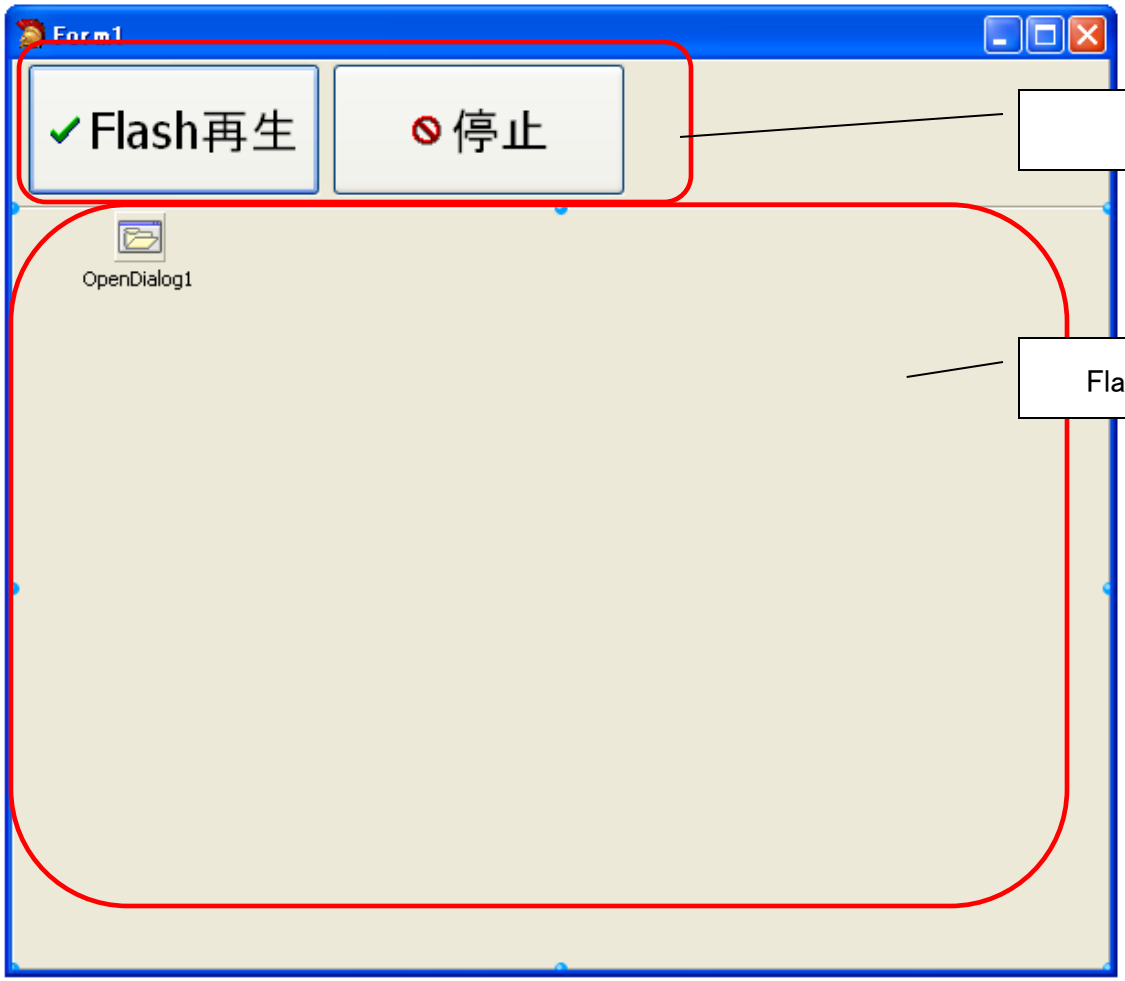
Playメソッド

Flash Movieを再生するメソッド

Stopメソッド

Flash Movieを停止するメソッド

■Flashプレイヤーの実装



再生・停止ボタン

Flash表示スペース（TPanel）

■Flashプレイヤーの実装

宣言部

```
private
    { Private 宣言 }
    //表示用Flash
    fFlashplayer : TshockwaveFlash;
```

終了処理(表示Flash破棄)

```
procedure TForm1.FormDestroy(Sender: TObject);
begin
  //表示用Flashplayerを破棄
  fFlashplayer.Free;
end;
```

初期処理(表示Flash生成)

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  //Flashをクラスから生成
  fFlashplayer := TFlash.Create(Self);
  //最大化
  fFlashplayer.Align := alClient;
  //表示用FlashplayerのPanelの上に設定（親子関係）
  fFlashplayer.Parent := pnlFlash;
end;
```

コンポーネントではないので、
生成してPanel上で表示する

■Flashプレイヤーの実装

再生ボタンクリック

```
procedure TForm1.btnPlayClick(Sender: TObject);
begin
  //ダイアログを開く
  if OpenDialog1.Execute then
  begin
    //選択したファイルをMovieとして設定
    fflashplayer.Movie := OpenDialog1.FileName;
    //Flashを実行
    fflashplayer.Play;
  end;
end;
```

ダイアログで
Flashを選択

Playメソッド実行

停止ボタンクリック

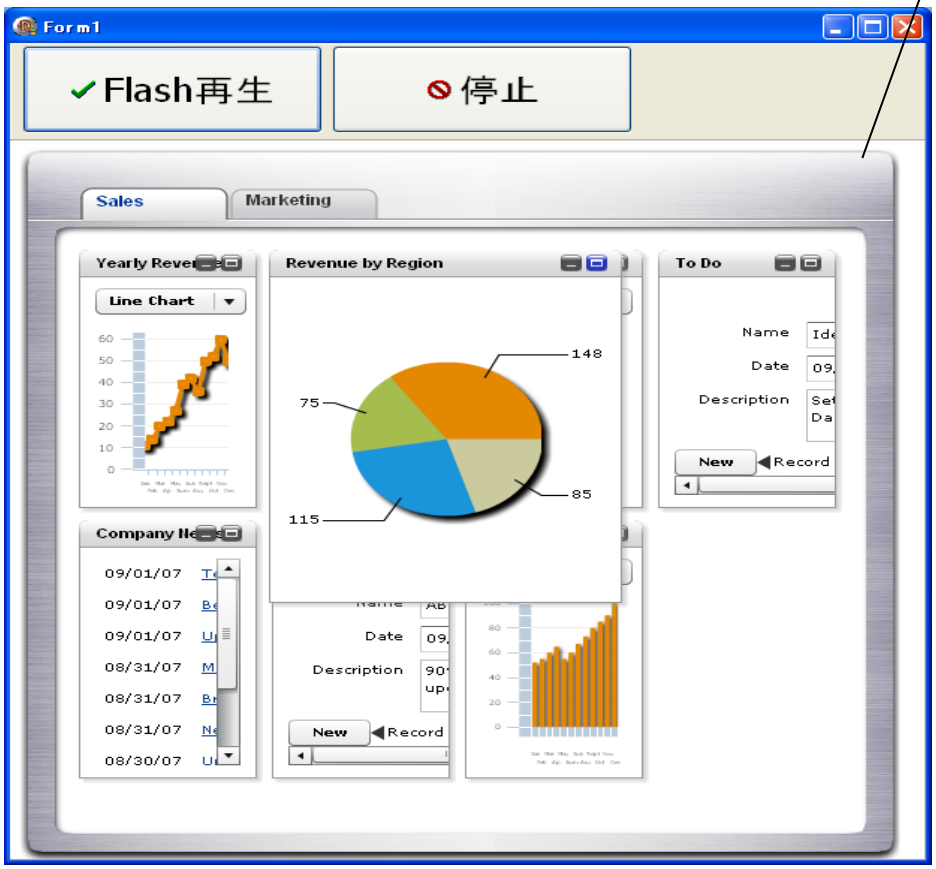
```
procedure TForm1.btnStopClick(Sender: TObject);
begin
  //Flashを停止
  fFlashplayer.Stop;
end;
```

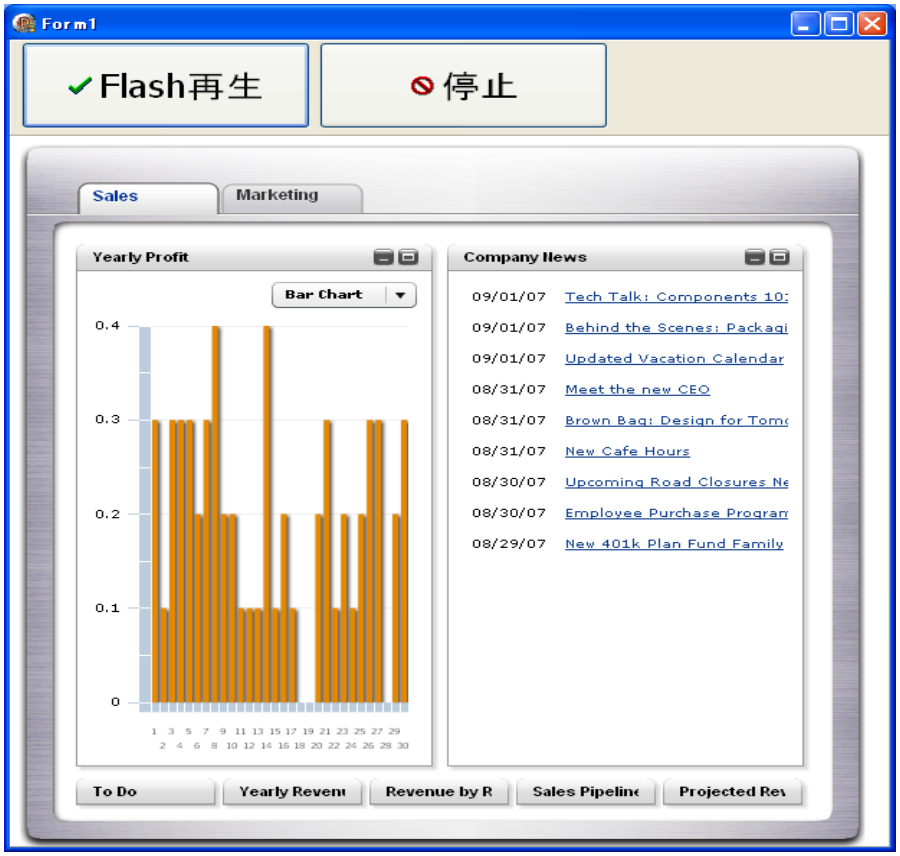
Stopメソッド実行

再生ボタンを押すと
ダイアログからFlashファイルを
選択して再生

■Flexアプリを実行しよう

Flexで作成されたFlashアプリケーションも
もちろん実行可能





■YouTubeを再生しよう

YouTube再生ボタンクリック

```
procedure TForm1.btnTubeClick(Sender: TObject);
begin
  //入力のVIDを指定してMovieに設定
  fFlashplayer.Movie := 'http://www.youtube.com/v/'
                        + edtVID.text;
  //Flashを実行
  fFlashplayer.Play;
end;
```

URLと入力した動画IDを設定

YouTubeのURLとVIDを指定すればWeb経由で動画を再生することも可能

補足資料

■TShockwaveFlashクラスの機能修正

- Flashの構造上の問題で
  「Stopメソッドを呼んでも必ずしも停止しない現象」
  があります。これに修正するためにTShockwaveFlash
  クラスからTFlashというクラスを用意します。

```
//宣言部
type
  //Stop、Playメソッドの動作を補正するためのクラス
  TFlash = class(TshockwaveFlash)
  private
    FPlaying : Boolean;
  protected
    procedure WndProc(var vMsg: TMessage); override;
  public
    procedure Play; reintroduce;
    procedure Stop; reintroduce;
    property  Playing: Boolean read FPlaying;
  end;
```

補足資料

■TShockwaveFlashクラスの機能修正

- 実装部ではWM_TIMERというWindowsメッセージを処理することでフレームの制御を行います。

```
//実装部
{ TFlashPlayer }
//Playメソッド
procedure TFlash.Play;
begin
  FPlaying := True;

  inherited play;
end;

//Stopメソッド
procedure TFlash.Stop;
begin
  FPlaying := False;

  inherited stop;
end;
```

Playメソッドで
FPlayingフラグをTrue

Stopメソッドで
FPlayingフラグをFlase

```
//メッセージ
procedure TFlash.WndProc(var vMsg: TMessage);
begin
  case vMsg.Msg of
    WM_TIMER:
      if (FPlaying) then
      begin
        inherited;
      end
      else
      begin
        vMsg.Result := 0;
      end;
    else
      inherited;
  end;
```

Windowsメッセージ処理時に
FPlayingフラグがTrueで
あれば処理する

Windowsメッセージ処理時に
FPlayingフラグがFlaseで
あれば処理をしない(Result 0)

Q3.AnsiStringとUnicodeStringの違いとは？

【質問】

Delphi/400 V2009からUnicode標準となっていますが
Unicodeになるとどのような違いがあるのでしょうか？

【回答】

V2009以降はUnicodeStringを標準で扱うようになり、
使用できる文字の範囲も格段に広がりました。
しかしV2007まで使用してきたAnsiStringとは文字種類
だけでなく、バイトサイズなどいくつかの点が異なります。

■UnicodeStringのポイント

- ①AnsiStringとは　UnicodeStringとは
- ② AnsiStringとUnicodeStringの互換性
- ③全角・半角、バイトサイズの概念

■①AnsiStringとは　UnicodeStringとは

- AnsiStringとは
  Delphi/400 V2007まで標準で使用されていた
  文字列型がAnsiStringになります。
  例えばEditで入力する文字列やString型はディフォルトでAnsiStringとして動作しています。

～Delphi/400 V2007

```
var
  teststr     : String;  //AnsiString
```

日本語OS環境ではいわゆるShift-JISで扱っている
文字コード範囲を使用できます。

■①AnsiStringとは　UnicodeStringとは

- UnicodeStringとは
  Unicode(ユニコード)は国際化文字セットの名称で、世界中のほぼ全ての文字を含むテキストを管理し表現することが可能です。
  Delphi/400 V2009ではこのUnicodeを標準として採用しており、例えばEditで入力する文字列やString型はディフォルトでUnicodeString(UTF-16LE)として動作しています。

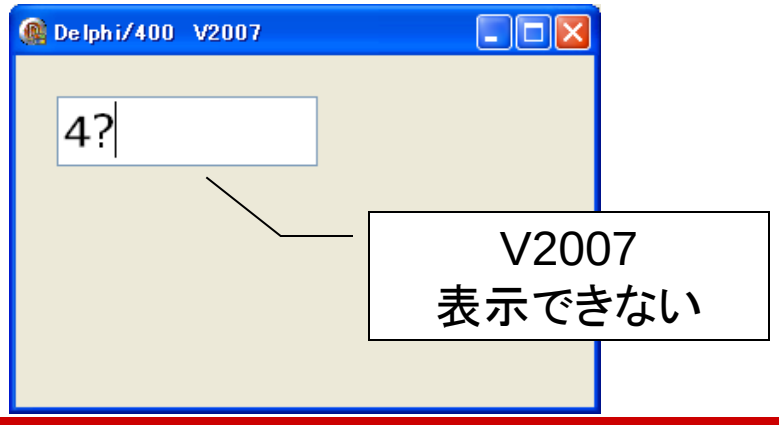
Delphi/400 V2009～

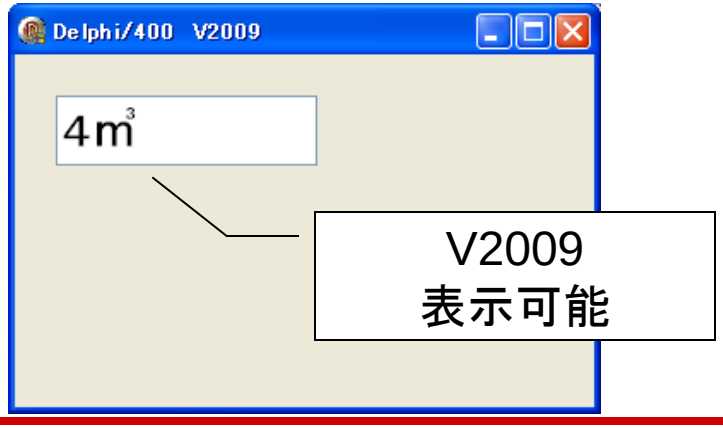
```
var
  teststr     : String;  //UnicodeString
```

■①AnsiStringとは　UnicodeStringとは

- AnsiStringとUnicodeStringの違い
  AnsiStringとUnicodeStringでは、扱える文字の種類が違います。例えばDelphi2007までのアプリケーションで入力すると“?”になって扱えない文字があります。

例）“4㎥”（4ﾘｯﾎﾟｳﾒｰﾄﾙ）と入力してみると・・・

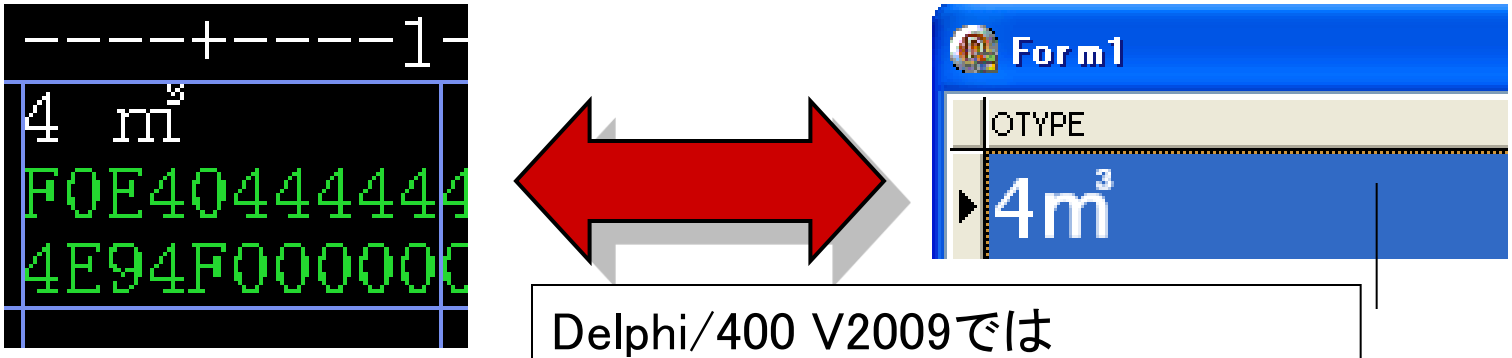




■①AnsiStringとは　UnicodeStringとは

- IBM i (AS/400)上でのUnicode文字

IBM i (AS/400)上でもCCSID1399でUnicode相当の文字を扱うことができます。

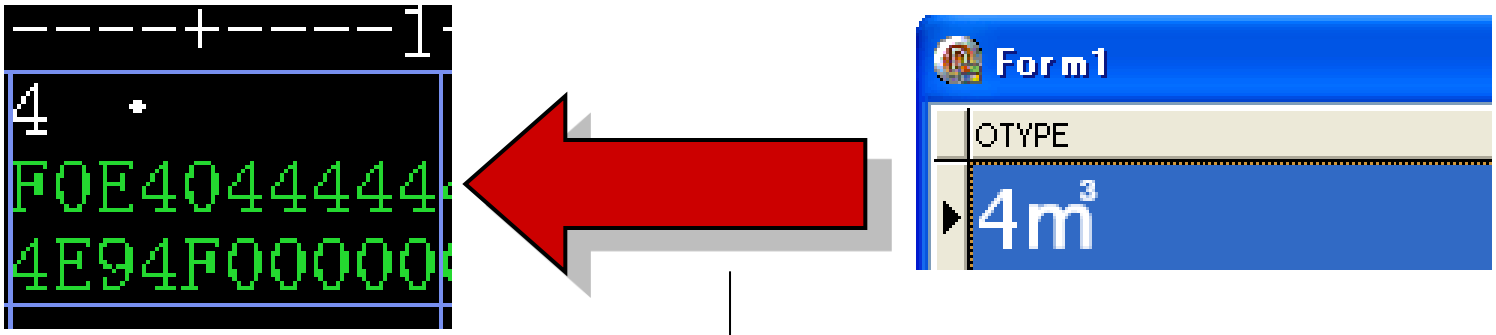


Delphi/400 V2009では
DBExpressでCCSID1399の文字列を扱えます

■①AnsiStringとは　UnicodeStringとは

- IBM i (AS/400)上でのUnicode文字

しかし従来のCCSID5035/5026では
これらの文字を扱うことができません。



V2009で入力ができても IBM i で扱えないCCSIDの場合、
逆に入力規制が面倒になるので、旧バージョンとV2009を
IBM i の環境によって選択することをお勧めします

■②AnsiStringとUnicodeStringの互換性

- AnsiString ⇔ UnicodeStringの代入
  AnsiStringとUnicodeStringは互いに代入することが可能です。

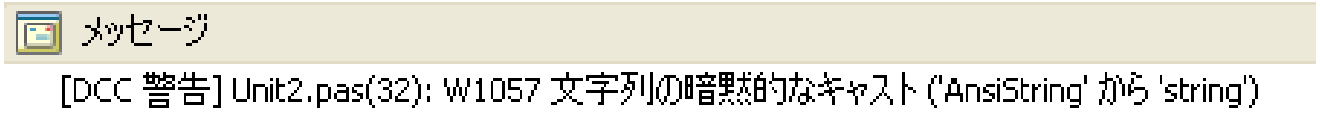
```
procedure TForm1.Button1Click(Sender: TObject);
var
  Ansistr    : AnsiString;    //AnsiString
  Unicodestr : UnicodeString; //UnicodeString
begin
  Ansistr    := Unicodestr; //UnicodeStringをAnsiStringに代入
  Unicodestr  := Ansistr;    //AnsiStringをUnicodeStringに代入
end;
```

どちらから代入しても
コンパイル上は問題なし

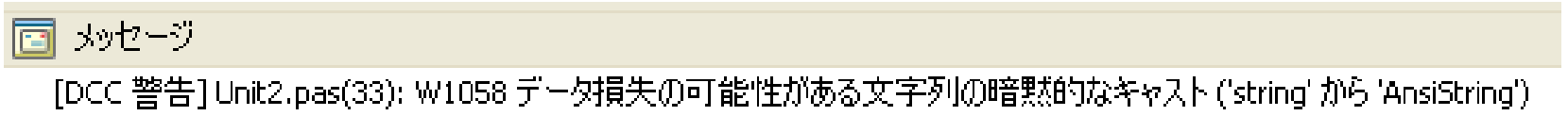
■②AnsiStringとUnicodeStringの互換性

- コンパイル時の警告
  これらは代入時に暗黙の文字列変換が行われていますが、コンパイル時に警告が表示されます。

AnsiStringをUnicodeStringに代入時



UnicodeStringをAnsiStringに代入時

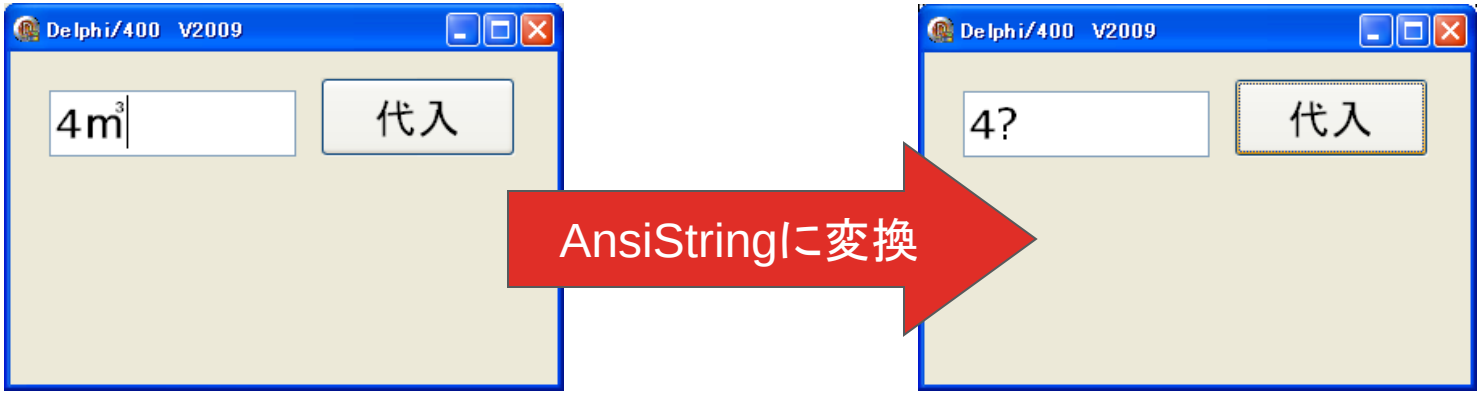


■②AnsiStringとUnicodeStringの互換性

- 変換時の文字列欠損

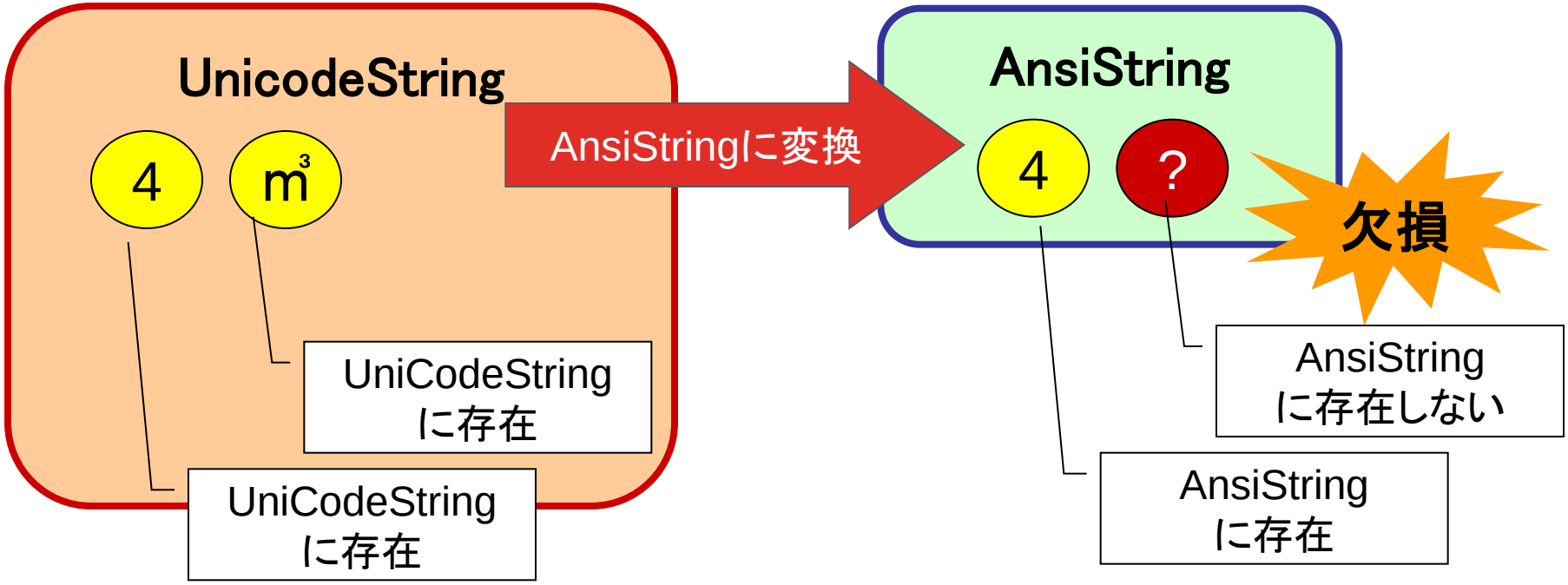
UnicodeStringをAnsiStringに代入時の注意点

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Ansistr    : AnsiString;  //AnsiString
begin
  Ansistr    := Edit1.Text; //UnicodeStringをAnsiStringに代入
  Edit1.Text := Ansistr;    //AnsiStringをUnicodeStringに代入
end;
```



■②AnsiStringとUnicodeStringの互換性

- 変換時の文字列欠損
  UnicodeStringをAnsiStringに代入時の注意点
  AnsiStringに存在しないUnicodeStringの文字列を代入すると
  文字列が欠損してしまう。



■③全角・半角、バイトサイズの概念

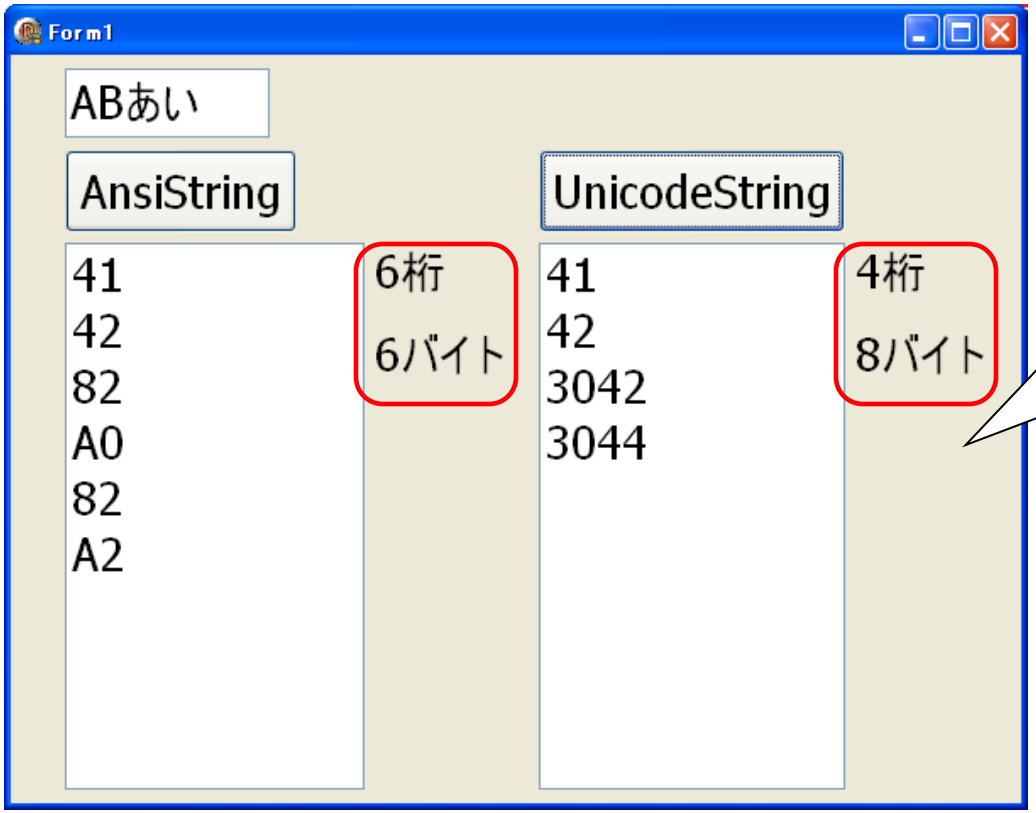
- Shift−JISベースのAnsiのプログラムでは
「全角2バイト文字、半角1バイト文字」という
固定観念があります。しかしUnicodeのプログラムでは、
マルチバイト文字として扱われますので
全角、半角といった概念がありません。

「全角2バイト文字、半角1バイト文字」ルール前提で
作成されたプログラムロジックはバイト計算の違い
などによってこれまでと異なる動作になってしまうので注意！

例）シフト文字考慮のために全角・半角をバイト関係の関数で
判断する場合など

■③全角・半角、バイトサイズの概念

- 実際にプログラム内の桁数やバイト数を見てみる。



同じ文字列を解析しても
桁数やバイト、コードの
持ち方が全く異なる

■③全角・半角、バイトサイズの概念

- 実際にプログラム内の桁数やバイト数を見てみる。

AnsiString

文字	A	B	あ		い	
バイト	1	2	3	4	5	6
バイナリ	41	42	82	A0	82	A2

UnicodeString

文字	A		B		あ		い	
バイト	1	2	3	4	5	6	7	8
バイナリ	00	41	00	42	82	A0	82	A2

これまでのAnsiプログラムのように全角や半角をバイト数などで判断することはできません。

■③全角・半角、バイトサイズの概念

- では全角・半角をどのように判断すれば良いのか？

  Unicodeコンソーシアムで策定されている
  EastAsianWidth属性を利用することができます。

EastAsianWidth
Unicodeにおける東アジア圏における各文字の全角/半角の定義に関しては、Unicodeコンソーシアムの『EastAsianWidth.txt』という変換表に定義されています。
http://unicode.org/Public/UNIDATA/EastAsianWidth.txt

■③全角・半角、バイトサイズの概念

EastAsianWidthを利用するために便利な情報

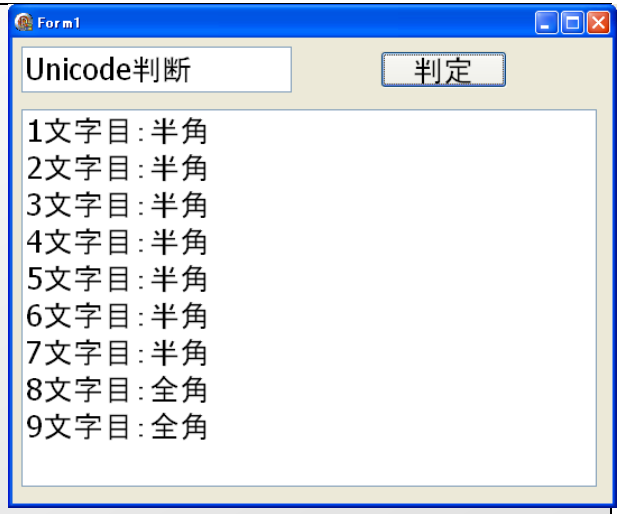
- 公開されている提供ソース
（エンバカデロ・テクノロジーズ提供ではありません）

【MECSUtilsユニット】
http://cc.embarcadero.com/item/26061
【リファレンス】
http://homepage1.nifty.com/ht_deko/tech021.html

■③全角・半角、バイトサイズの概念

- MECSUtilsユニットを使ってみる（MecsIsFullWidth関数）

```
procedure TForm1.btnExecClick(Sender: TObject);
var
  UnicodeStr: UnicodeString; //UnicodeString
  fnt: TFontCharset;         //使用フォント
  i: Integer;                //U処理用インデックス
begin
  UnicodeStr := Edit1.Text;           //Edit1のテキスト
  fnt        := Edit1.Font.CharSet;   //Edit1のフォント
  //ログのクリア
  ListBox1.Items.Clear;
  //Edit1のテキストを1ワードずつ判断
  for i:=1 to MecsLength(UnicodeStr) do
    //文字列をインデックス、使用フォントでMecsIsFullWidthを判断
    if MecsIsFullWidth(UnicodeStr, i, IsFarEastCharSet(fnt)) then
      //全角判断ログ
      ListBox1.Items.Add(IntToStr(i) + '文字目：全角')
    else
      //半角判断ログ
      ListBox1.Items.Add(IntToStr(i) + '文字目：半角');
end;
```



MecsIsFullWidthという提供関数で判断可能
※MECSUtilsはミガロ.開発ソースではありませんのでDLしてください。

■Delphi2009～のUnicodeについて参考情報（開発元）

- Delphi Unicodeワールド パートI: Unicodeとは？ なぜ必要なのか？ そして、Delphiでどのような作業を行うのか？
  http://dn.codegear.com/jp/article/38695

- Delphi Unicodeワールド パートII: RTLの新機能と、Unicodeをサポートするクラス
  http://dn.codegear.com/jp/article/38698

- Delphi Unicodeワールド パートIII: コードをUnicode対応にする
  http://dn.codegear.com/jp/article/38699

Q4.複数アプリ間でパラメータを受け渡すには？

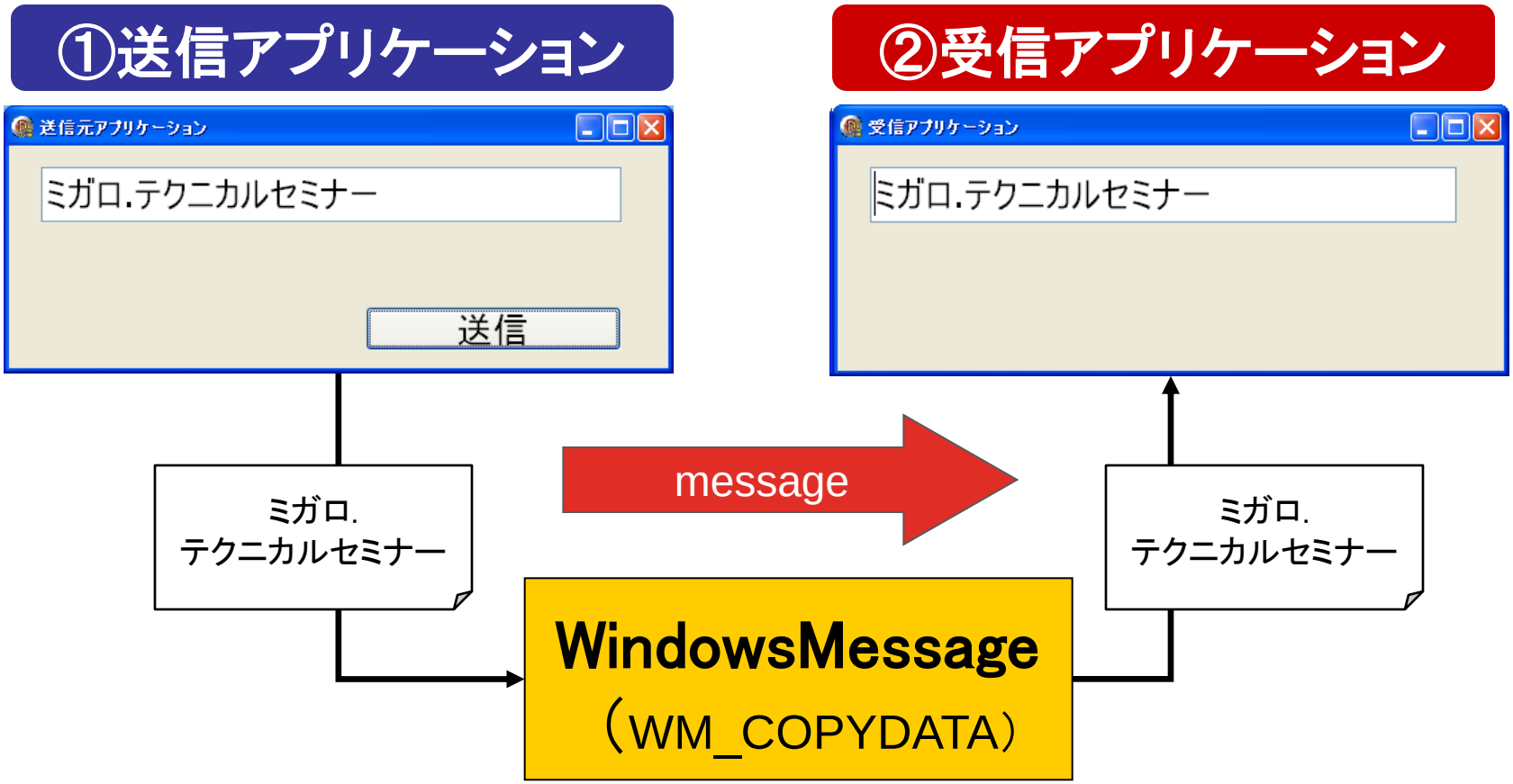
【質問】

起動中のアプリケーション間でパラメータを受け渡して
処理を行うことはできますか？

【回答】

アプリケーション間で通信を行う手法は色々ありますが、
WM_COPYDATAというWindowsMessageを利用
すると簡単に値の受渡しが行えます。

■WindowsMessageを利用したアプリケーション間の通信



■パラメータ(メッセージ)送信プログラムの作成

●【①送信アプリケーション】

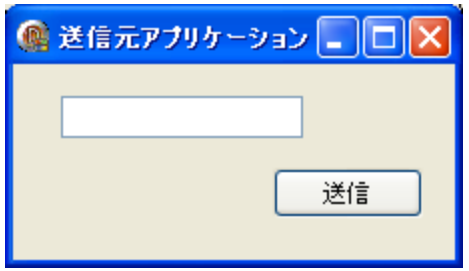
①-1. WM_COPYDATA を利用してメッセージを送信する場合、
送信先のWindowHandleを取得する

①-2. 次に送信する情報をTCopyDataStruct 構造体に設定する

TCopyDataStruct

dwData 任意の32ビット値
cdData 送受信するデータのサイズ(バイト数)
lpData 送受信するデータのポインタ

①-3. WM_COPYDATAをメッセージ送信する



■パラメータ(メッセージ)送信プログラムの実装

```
procedure TForm1.Button1Click(Sender: TObject);
var
  hWindow : HWND;
  CDS     : TCopyDataStruct;
  SendStr : String;
begin
  //送信先ウィンドウハンドルの取得
  hWindow := FindWindow('TForm2',nil);
  if (hWindow < 1) then
  begin
    exit;
  end;
  //画面入力値(ここではEdit1の文字列)
  SendStr := Edit1.Text + #0;
  //任意の値
  CDS.dwData:=999;
  //バイトサイズ
  CDS.cbData:=Length(SendStr) * sizeof(Char);
  //送信文字列
  CDS.lpData:=PChar(SendStr);
  //メッセージ送信
  SendMessage(hWindow,WM_COPYDATA,Self.Handle,LPARAM(Addr(CDS)));
end;
```

①-1.送信先のWindowHandleを取得する
(ここではTForm2)

①-2.送信する情報をCopyDataStruct 構造体に設定
バイトサイズはWideCharの場合Lengthのみで
算出しないように注意
もしくはByteLengthで取得する(Delphi2009)

①-3. WM_COPYDATAをメッセージ送信する

Delphi2007以前と
2009以降で扱いが
変わるもの

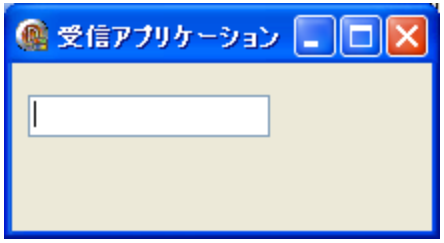
[V2009～]
String:UnicodeString
Char:WideChar
Pchar:PWideChar
SendMessage:
SendMessageW

[～V2007]
String:AnsiString
Char:AnsiChar
Pchar:PAnsiChar
SendMessage:
SendMessageA

■パラメータ(メッセージ)の受信プログラムの実装

●【②受信アプリケーション】

②-1.WM_COPYDATAをメッセージからメッセージを取得する




```
private
   { Private 宣言 }
   procedure WMCopyMessage(var WMCopyData: TWMCopyData);
     message WM_COPYDATA;

procedure TForm2.WMCopyMessage(var WMCopyData: TWMCopyData);
var
  ReceiveStr: String;
begin
  //受け取ったパラメータ（メッセージ）を処理
  ReceiveStr:=PChar(WMCopyData.CopyDataStruct.lpData);
  Edit1.Text:=ReceiveStr;
end
```

②-1.WM_COPYDATA を受け取るためのメッセージハンドラを作成

②-1.パラメータ(メッセージ)を受け取って処理

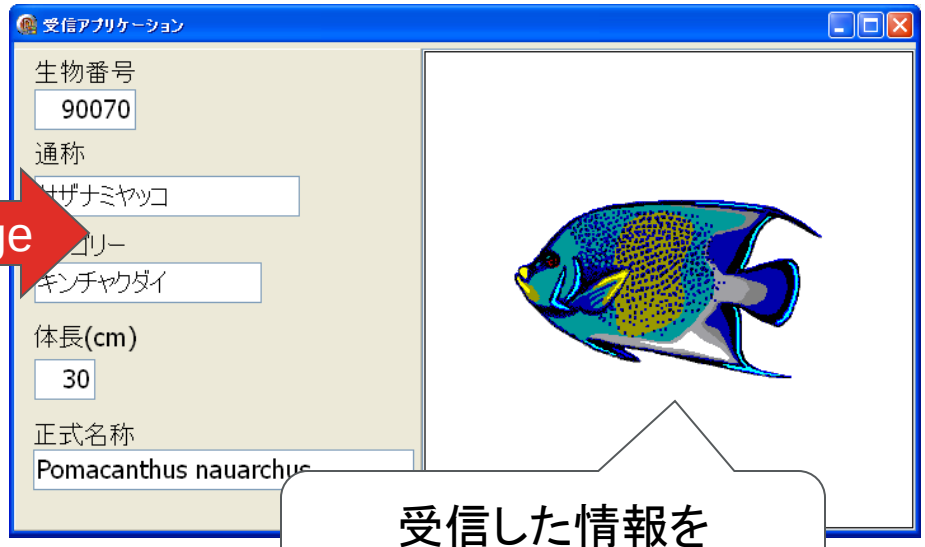
■アプリケーション間の送受信を応用する

- 同様にアプリケーション間でメッセージによって同期をとった連携動作も可能

送信アプリケーション

message

受信アプリケーション



レコード移動ごとに現在の情報を送信

受信した情報を元にデータを読込み

ご静聴ありがとうございました