WHITE PAPER

ADOCEとEmbedded Visual Basicを使用したAdaptive Server Anywhere for Windows CEのデータアクセス

はじめに

ActiveX Data Objects for CE (ADOCE)は、Windows CEデータアクセス開発者間で評判の高いツールです。ADOCEは、ActiveX Data Objects (ADO)のWindows CEプラットフォーム対応版です。ADOCEはEmbedded Visual Basic (EVB)と統合し、OLE DBプロバイダを使用してデータベースのデータにアクセスします。

このホワイトペーパでは、Adaptive Server Anywhere for Windows CE (ASA for CE)およびiAnywhere OLE DBプロバイダと共にADOCEの使用に関して説明していきますが、ADOCEやEmbedded Visual Basicのプログラミングについては、特に説明していません。また、ADOCEクライアント・アプリケーションのサンプルや、EBF (emergency bug fix)、未解決の問題、iAnywhere OLE DBプロバイダであるASAProvのサポート対象に関する情報も提供していきます。あわせて次のリソースを使用することをお奨めします。

- MSDNライブラリ

  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ado270/htm/mdmscadoprogrammersguide.asp
- Adaptive Server Anywhereのマニュアル（[スタート] - [プログラム] - [Sybase SQL Anywhere 7] - [SQL Anywhere Documentation]の順に選択）
- ホワイトペーパー『ADOとVisual Basicを使用したAdaptive Server Anywhereのデータへのアクセス』

このホワイトペーパーは、Adaptive Server Anywhere 7.0.3 ビルド 2067 に基づいていますが、バージョン 7.0.4 や 8.x を含む新しいバージョンの Adaptive Server Anywhere でも使用できます。

Adaptive Server Anywhere バージョン 8 を使用している場合は、ODBC データ・ソースを作成する時や Adaptive Server Anywhere のバージョンを確認するときに、開始行、ディレクトリ、ファイル名の 7 を 8 に置き換えてください。

このホワイトペーパーの執筆中に、ADOCE の使用方法を改良するためのいくつかの修正が Adaptive Server Anywhere に対して行われました。したがって、問題を回避するために、サポートにご契約いただき、最新の保守リリースを使用することをお奨めします。

AddBookmark プロジェクトでは、ロックを最適にしたい場合は、キーセット駆動型カーソルを使用する必要があります。詳細については、ホワイトペーパー『ADO と Visual Basic を使用した Adaptive Server Anywhere のデータへのアクセス』の「はじめに」を参照してください。

参考のため、このホワイトペーパーにはコード・サンプルを掲載しています。プロジェクトが機能することを確認するためには、コード・サンプルをコピー・アンド・ペーストするのではなく、実際のプロジェクトを使用することをお勧めします。

米国Sybase Inc. の Web サイト http://www.sybase.com/content/1019003/adoceprojects.zip から、入手できます。

目次

ADO と ADOCE

ADOCE はデスクトップ ADO のサブセットで、特に Windows CE プラットフォームに対応するように設計されています。ADO と同様に、iAnywhere Solutions の ASAProv や Microsoft の OLE DB プロバイダである MSDASQL のような OLE DB プロバイダを使用するデータ・ソースに対応しています。ADOCE コントロールを使用するときは、デスクトップ版の ADO と同じ構文を使用できまる上、ADO から ADOCE への移植は非常に簡単です。また、ADO でも ADOCE でも同じエラー値や文字列が報告されるため、エラー処理が簡単です。ただし、Microsoft の基準に基づいて、ADO でサポートされているオブジェクトやコレクションがすべて ADOCE でサポートされるとは限りません。

ADOCE でサポートされるオブジェクトやコレクションの詳細については、11 ページの「ADOCE オブジェクト」を参照してください。

ADO の詳細については、ホワイトペーパー『ADO と Visual Basic を使用した Adaptive Server Anywhere のデータへのアクセス』の「ActiveX Data Objects」を参照してください。

ADOCE 3.0 と ADOCE 3.1

このホワイトペーパーは、Microsoft ADOCE 3.0 Control for Pocket PC に基づいています。ただし、ここに掲載されている例の中には、最新バージョンであるバージョン 3.1 を使用しているものもあります。これは、Blob を使用するときに 3.1 Control の一部の機能が必要なためです。

Microsoft ADOCE 3.1 Control の新機能の詳細については、MSDN ライブラリの [Embedded Developer Documentation] - [Windows CE Application Frameworks] - [Microsoft ADOCE version 3.1 Start Page] - [What's New in ADOCE Version 3.1] を参照してください。

入門

この項では、ODBC データ・ソースの設定方法、ADOCE コントロール・ライブラリの参照方法、Adaptive Server Anywhere 7 for Windows CE の設定方法を説明します。

Adaptive Server Anywhere 7 for Windows CE の設定

Adaptive Server Anywhere 7.0.3 for Windows CE は、Adaptive Server Anywhere 7.0.3 のインストール時に自動的にインストールされるものではありません。Adaptive Server Anywhere for Windows CE をインストールするためのチェックボックスを選択してください。Adaptive Server Anywhere 7.0.3 を以前にインストールしている場合は、最新のビルドまたは保守リリースを使用していることを確認してください。

使用しているバージョンを確認するには、次の手順に従います。

1 win32 ディレクトリに移動します。デフォルトでは、このディレクトリは次の場所にあります。

```
c:\ Program Files\Sybase\SQL Anywhere 7\CE
```

2 ADOCE アプリケーションに使用しているものと一致するデバイス・フォルダを開きます。

3 dboledb7.dll を右クリックしてポップアップ・メニューから [プロパティ] を選択します。

4 図 1 に示すように、[バージョン情報] タブで、[ファイル バージョン] が 7.0.3 2082 であることを確認します。Adaptive Server Anywhere 7.0.4 以上のバージョンを使用している場合は、バージョン番号が異なります。

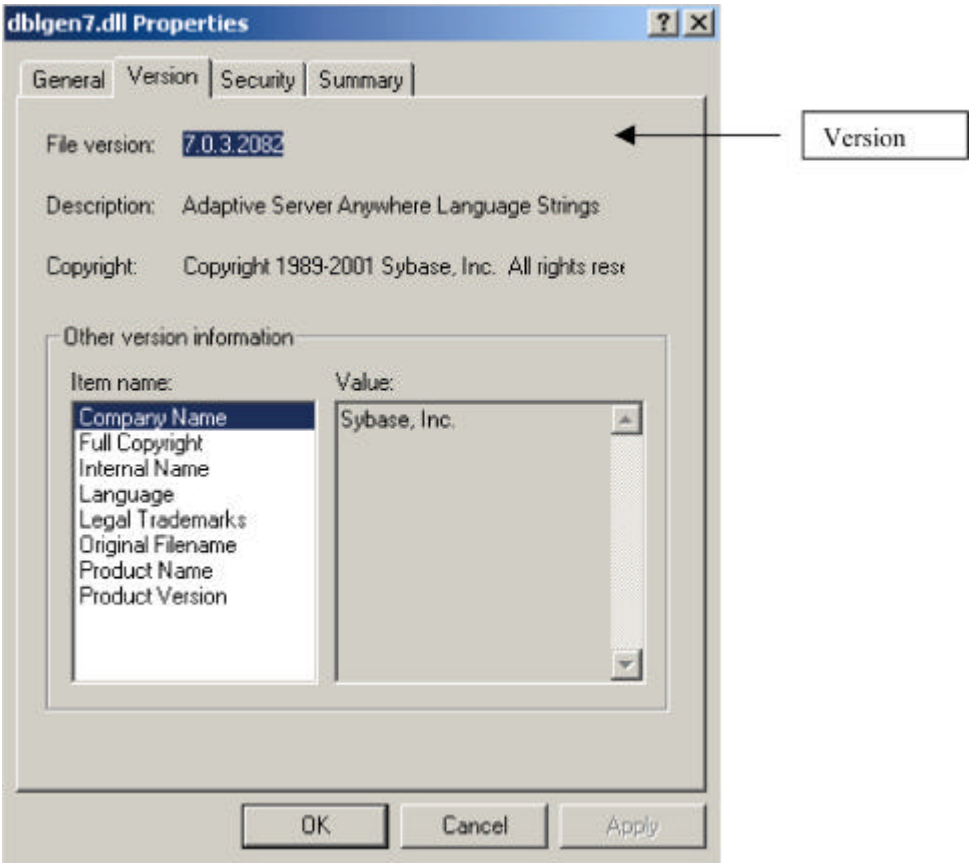


図 1：Adaptive Server Anywhere のビルド・バージョンの確認

このホワイトペーパーは、Adaptive Server Anywhere 7.0.3 ビルド 2067 に基づいています。このホワイトペーパーの執筆中に、ADOCE の使用方法を改良するためのいくつかの修正が Adaptive Server Anywhere に対して行われました。したがって、問題を回避するために、最新の保守リリースを使用してください。

最新の保守リリースをダウンロードするには、次の手順に従います。

1 Web ブラウザで次の URL を開きます。
www.sybase.com

2 左のウィンドウ枠で、[Downloads] を選択します。

3 左のウィンドウ枠で、[EBFs/Update] を選択します。Sybase Web アカウントをすでにお持ちの場合は、ユーザ名とパスワードを入力します。お持ちでない場合は、[Sign Me Up] をクリックします。

4 [Product Families] のリストで [SQL Anywhere Studio] をクリックします。

5 最新の保守リリースを探してダウンロードします。

特定のバグ・フィックスまたは保守リリースに関する詳細については、該当する EBF または更新の隣にある [Info] ボタンをクリックします。[Info] リンクは新聞のような形のアイコンで表示されます。[Info] ページ上で OLE DB を検索します。

ODBC データ・ソースの作成

ODBC データ・ソースを作成するには、次の手順に従います。

1 [スタート] - [プログラム] - [Sybase SQL Anywhere 7] - [Adaptive Server Anywhere 7] - [ODBC Administrator] の順に選択します。

2 [File DSN] タブで [Add] をクリックします。

3 リストから Adaptive Server Anywhere 7.0 を選択して [Next] をクリックします。

4 ファイル名を入力してデータ・ソースをデスクトップ・マシンに保存するか、または [Browse] をクリックして必要なファイルのロケーションを探します。DSN は、Windows CE デバイスまたはエミュレータに移動します。

5 [Finish] をクリックします。[ODBC Configuration for Adaptive Server Anywhere] ダイアログ・ボックスが開きます。

6 図 2 に示すように、[ODBC] タブで、[Data Source Name] に ASA 7.0 Sample と入力します。

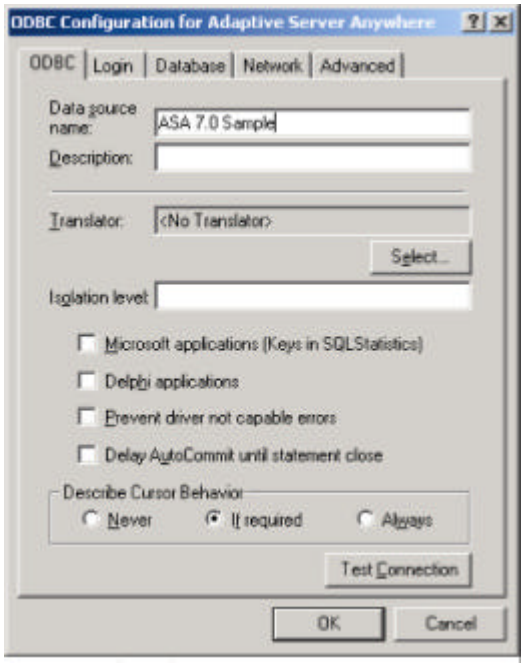


図 2：[ODBC] タブ

7 図 3 に示すように、[Login] タブで、[User ID] に dba、[Password] に sql と入力します。

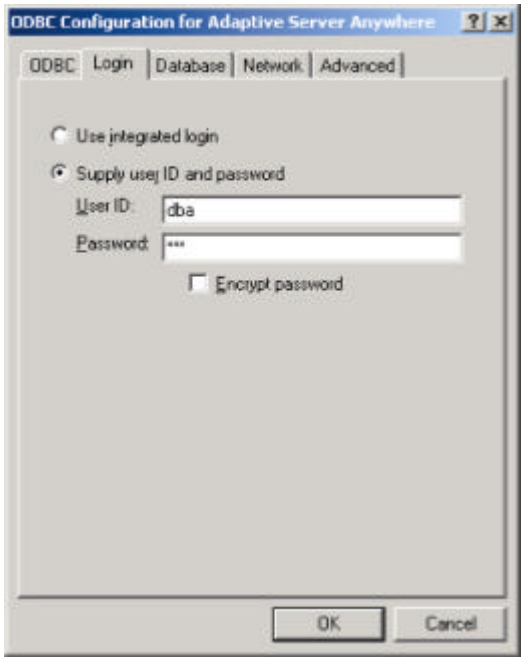


図 3：[Login] タブ

8 [Database] タブで、[Browse] ボタンを使用してデータベース・ファイルを指定します。図 3 に示すように、[Start Line] に dbeng7.exe と入力します。-c 12M はオプションです。

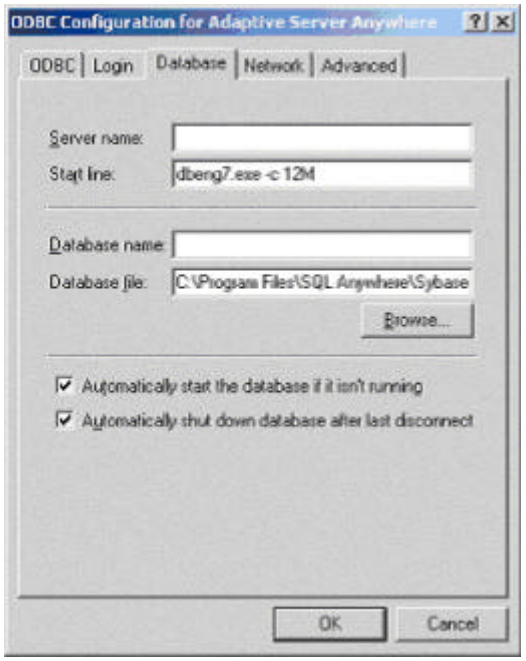


図 4：[Database] タブ

9 [ODBC] タブで [Test Connection] をクリックして接続完了のメッセージを待ちます。

10 [OK] をクリックして [ODBC Configuration] ダイアログ・ボックスを閉じます。

独自のユニークなファイル DSN を作成するときは、データ・ソース作成時にマシン上にあるデータベースを参照する必要があります。これにより、ファイル DSN をデバイスまたはエミュレータのデスクトップにコピーできます。エミュレータのデスクトップは次のディレクトリにあります。

```
<Windows CE の Tools ディレクトリ>\ wce300\ MS Pocket PC\ emulation\
palm300
```

このファイルのコピーを作成したら、ワードパッドやメモ帳などのテキスト・エディタを使用してファイルを開きます。ファイルに保存されている Databasefile 行と Start 行の両方を変更する必要があります。この 2 行を、デバイスまたはエミュレータ上の asademo.db および dbsrv7.exe のロケーションに変更します (2 つは同じロケーションにあります)。Databasefile 行と Start 行は、次のようになります。

```
databasefile=\Program Files\ASA\asademo.db
start=\Program Files\ASA\dbsrv7.exe
```

これを、元の名前と同じ .DSN ファイル名を使用して保存し、テキスト・エディタを閉じます。

このホワイトペーパーに掲載されている例では、使いやすいように、ODBC データ・ソースを使用してデータベースに接続します。ODBC データ・ソースを使用せずに接続を行う場合は、データベースのファイル名と場所を含めてください。このホワイトペーパーでは、すべてのサンプルで、Adaptive Server Anywhere サンプル・データベースである asademo.db を使用します。ASA 7.0 Sample という ODBC データ・ソースは、asademo.db データベース用に作成されているため、サンプル・コードを使用するときに再作成する必要はありませんが、この DSN ファイルがデバイスまたはエミュレータのデスクトップ上にあることを確認する必要があります。

ファイル DSN は、作成する必要はありません。すべての仕様は、DSN へのリファレンスにではなくコード内に含めることができます。

データベース接続の詳細については、Adaptive Server Anywhere のマニュアルの次の項を参照してください。

- Adaptive Server Anywhere バージョン 7 を使用している場合は、『ASA User's Guide』の第 2 章「Connecting to a Database」。
- Adaptive Server Anywhere バージョン 8 を使用している場合は、『ASA Database Administration Guide』の第 2 章「Connecting to a Database」。

Embedded Visual Basic: ADOCE コントロールの参照

Embedded Visual Basic プロジェクトを開始する前に、統合開発環境 (IDE) で適切な ADOCE コントロールを参照する必要があります。アプリケーションを実行すると、正しいコンポーネントがデバイスにインストールされます。

このホワイトペーパーでは ADOCE 3.0 Control を使用します。このコントロールはデフォルトであるため、プロジェクトを作成するときに、ライブラリを参照する必要はありません。ただし、このホワイトペーパーの一部の例では ADOCE 3.1 Control を使用しています。その場合は、次の手順を使用します。

ADOCE 3.1 コントロールを参照するには、次の手順に従います。

1 [Projects] - [References] を選択します。

2 図 5 に示すように、[Microsoft CE ADO Control 3.1] を選択して [OK] をクリックします。

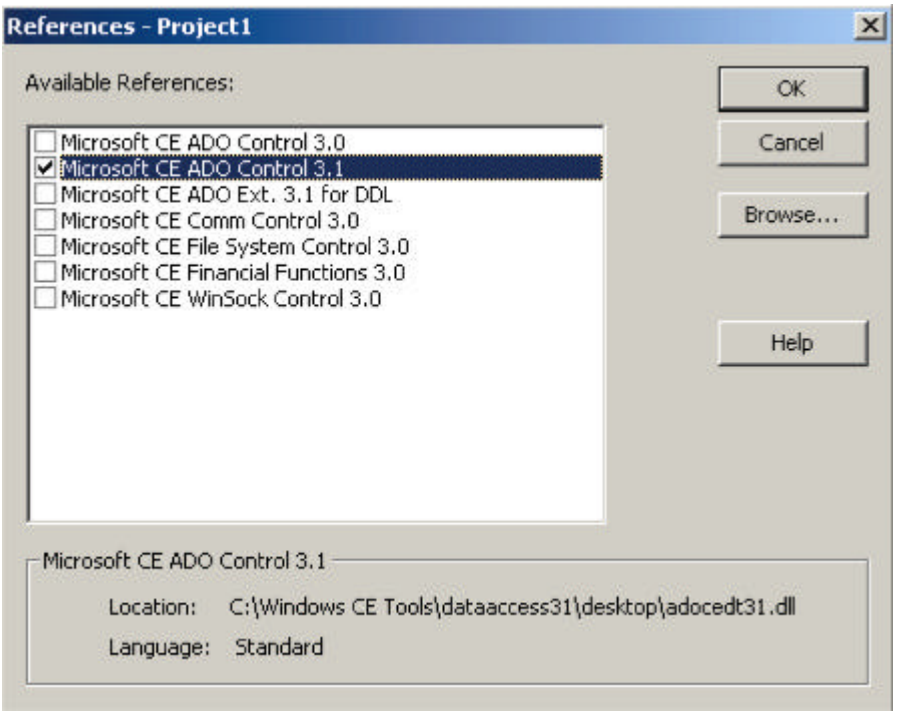


図 5：ADO 3.1 ライブラリの参照

プロバイダ

プロバイダの詳細については、ホワイトペーパー『ADO と Visual Basic を使用した Adaptive Server Anywhere のデータへのアクセス』の「プロバイダ」を参照してください。

ADOCE オブジェクト

ADOCE Control には、4 つのオブジェクトと 1 つのコレクションがあります。

- Connection オブジェクト
- Recordset オブジェクト
- Field オブジェクト
- Error オブジェクト
- Fields コレクション

これらのオブジェクトの詳細については、MSDN ライブラリの [Embedded Developer Documentation] - [Windows CE Application Frameworks] - [Microsoft ADOCE Version 3.1 Start Page] - [ADOCE Reference] - [Objects] を参照してください。

Connection オブジェクト

Connection オブジェクトは、データ・ソースに接続するときに必要です。1 つの Connection オブジェクトが、データ・ソースへの個々の接続をそれぞれ表します。

Connection オブジェクトの宣言

Connection オブジェクトを作成するには、次の構文を使用します。

```
Dim Connect As ADOCE.Connection
  Set Connect = CreateObject ("ADOCE.Connection.3.0")
```

CreateObject 関数を使用して ADOCE 3.0 コントロールへの参照を作成するときは、バージョン番号を含めてください。バージョン番号を含めなければ、デフォルトで旧バージョンのコントロールが使用されます。デバイス上に旧バージョンがない場合は、エラーが返されます。

使用しているサブルーチンの外で接続を開いたままにしておきたい場合は、接続をグローバルに宣言してください。Connection オブジェクトをサブルーチン内だけで宣言すると、サブルーチンを終了するとすぐに、メモリを開放するために接続が閉じられます。

Connection Open メソッドの使用

このメソッドには 2 つの使用方法があります。次の例では、Adaptive Server Anywhere 7.0 の Sample データベースを使用します。

```
Private Sub Command1_Click()
Dim ConnString As String

Dim Connect As ADOCE.Connection

ConnString = "Provider=ASAProv;Data Source=ASA 7.0 Sample;" & _
     "User Id=dba;Password=sql;"

Set Connect = CreateObject("ADOCE.Connection.3.0")
Connect.Open ConnString

End Sub

Private Sub Command2_Click()
    Dim Connect As ADOCE.Connection
```

```
    Set Connect = CreateObject("ADOCE.Connection.3.0")

    Connect.Provider = "ASAProv"
    Connect.ConnectionString = "Data Source = ASA 7.0 Sample"
    Connect.Open
End Sub
```

接続の切断

```
Connect.close
Set Connect = nothing
```

Connect.close で ADOCE Connection オブジェクトが閉じられます。接続を nothing に設定すると、データベースが切断され、エンジンが停止します。

Recordset オブジェクト

レコードセットは仮想データベース・テーブルで、レコードセットのフィールドとローは、Windows CE ベースのエミュレータまたはデバイス上の、実際のデータベース・テーブルにあるフィールドとローに対応します。レコードセット内のデータを変更すると、変更内容はメモリに格納されるため、基本となるデータベースが更新される前であれば、変更をキャンセルできます。ADOCE データベースはシングルユーザ・アクセスを目的としているため、ADOCE はバッチ更新をサポートしません。

Recordset オブジェクトの詳細については、MSDN ライブラリを参照してください。

Recordset Open メソッドの使用

レコードセットを開くための構文は次のとおりです。

```
RS.Open Source, ActiveConnection, cursortype, locktype, options
```

Recordset オブジェクトを開く前に、Connection オブジェクトを開いておく必要があります。ADOCE は、暗黙的な接続をサポートしていません。Recordset オブジェクトを使用してデータベースに暗黙的に接続することはできません。次の RSConnect という例で、これを説明します。

```
Option Explicit

Dim Connect As ADOCE.Connection
Dim RS As ADOCE.Recordset
Dim ConnString As String
```

```
Dim SQLstring As String

Private Sub Command1_Click()
Set Connect = CreateObject("ADOCE.Connection.3.0")
  Set RS = CreateObject("ADOCE.RecordSet.3.0")

  ConnString = "Provider=ASAProv.70;" & _
       "Data Source=ASA 7.0 Sample;" & _
       "User Id=dba; Password=sql;"

  Connect.Open ConnString

  SQLstring = "Select * FROM Product "
  Set RS.ActiveConnection = Connect
  RS.Open SQLstring, Connect, adOpenKeyset, & _
       adLockOptimistic, adCmdText

End Sub

Private Sub Command2_Click()
  RS.Close
  Set RS = Nothing
  Connect.Close
  Set Connect = Nothing

End Sub
```

Source パラメータ

ADOCE で、Source パラメータは、Recordset Open メソッドに必ず含めなければならない唯一のパラメータで、SQL 文、テーブル名、ストアド・プロシージャのいずれかを変形したものです。上記の例では、SQL 文を使用しています。Adaptive Server Anywhere は、Transact-SQL (T-SQL) と Watcom-SQL の大容量のサブセットを使用してデータベースをクエリします。

Transact-SQL と Adaptive Server Anywhere の互換性については、Adaptive Server Anywhere のマニュアルの次の項を参照してください。

- Adaptive Server Anywhere バージョン 7 を使用している場合は、『ASA User's Guide』の第 6 部「The Adaptive Server Family」の第 32 章「Transact-SQL compatibility」。
- Adaptive Server Anywhere バージョン 8 を使用している場合は、『ASA SQL User's Guide』の第 12 章「Transact-SQL Compatibility」。

Active Connection

Active Connection パラメータは、Open Recordset 文に含めるか、またはその直前に呼び出すことができます。Active Connection パラメータを Open Recordset 文の前に呼び出す場合は、次のコマンドを使用します。

```
RS.ActiveConnection = Connect
```

Cursor Types

Cursor Types は、データベース内を移動する際の移動方法を指定する、オプションのパラメータです。基本となるデータベースに対する更新が Recordset オブジェクトにどのように反映されるかにも影響します。次の表に、ASAProv が ADOCE でサポートする 4 つのカーソル・タイプを示します。

定数	値	説明
AdOpenForwardOnly	0	前方専用カーソル。静的カーソルと同じですが、レコードを前方にしかスクロールできません。ただし、デスクトップ・コンピュータ用の ADO との互換性を保つために、デフォルトになっています。
AdOpenKeyset	1	キーセット駆動型カーソル。他のユーザが行った追加や削除は参照できません。ただし、他のユーザが行ったデータ変更は参照できます。レコードセットを使用するすべてのタイプの移動が可能です。
AdOpenDynamic	2	動的カーソル。他のユーザが行った追加、変更、削除を参照でき、レコードセットを使用するすべてのタイプの移動が可能です。
AdOpenStatic	3	静的カーソル。データの検索やレポートの作成に使用できるレコードのセットの静的コピー。他のユーザが行った追加、変更、削除は参照できません。

他のカーソル・タイプを指定しない場合のデフォルトのカーソル・タイプは adOpenForwardOnly です。ADOCE の SQLOLE DB プロバイダでは、動的タイプ、静的タイプのカーソルは使用できません。またADOCE の ASAProv プロバイダは、4 つのカーソル・タイプすべてをサポートします。

ASAProv と SQLOLEDB の全般的なサポート内容の比較については、ホワイトペーパー『ADO と Visual Basic を使用した Adaptive Server Anywhere のデータへのアクセス』の「カーソル・タイプ」を参照してください。

Lock Type

Lock Type もオプションのパラメータで、プロバイダが Recordset オブジェクトを開くときに、どのタイプのロックまたは同時実行性を使用するかを指定します。ADOCE はバッチ更新をサポートしないため、データベースに対する変更は、Update メソッドを使用した場合にのみ可能になります。

次の表は、ADOCE で ASAProv がサポートしているロック・タイプを示しています。

定数	値	説明
adLockReadOnly	1	読み込み専用。レコードの追加、削除、変更はできません。
adLockPessimistic	2	レコードごとのペシミスティック・ロック。プロバイダは、通常は編集直後にデータ・ソースのレコードをロックさせることによって、レコードの編集を成功させるために必要なことを行います。レコードは追加、削除、変更できます。
adLockOptimistic	3	レコードごとのオプティミスティック・ロック。Update メソッドが呼び出された場合にのみ、プロバイダはレコードをロックします。レコードは追加、削除、変更できます。

デフォルトのロック・タイプは adLockReadOnly です。ADO の ASAProv でサポートされている adLockBatchOptimistic タイプのようなバッチ・ロックは、どのプロバイダでもADOCE ではサポートされていません。adOpenKeyset カーソル・タイプを選択している場合、ASAProv は別のアプリケーションによってローが更新済みであるという警告を出す唯一のカーソル・タイプである adLockOptimistic をサポートしています。その他のカーソル・タイプはすべて、オプティミスティック・ロック・タイプの機能を維持できません。

Options パラメータ

Options パラメータは、プロバイダによる Source パラメータの解釈方法を指定するオプションのパラメータです。

次の表では、このパラメータに関する理解を深めるために、各オプションについて詳細に説明します。

定数	値	説明
adCmdText	1	ource を SQL 文として評価
adCmdTable	2	ource をテーブル名として評価
adCmdStoredProc	4	ource をストアド・プロシージャとして評価
adCmdUnknown	8	フォルト。Source パラメータ内のコマンドのタイプは不明

adCmdTableDirect	512	CommandText を、カラムがすべて返されるテーブル名として評価

ASAProv は、これらすべてのオプションを ADOCE でサポートします。他のオプションを指定しない場合のデフォルトは adCmdUnknown です。

Update メソッドを使用したレコードセットのデータ更新

開発者は、レコードセットを使用してデータを更新し、新しいデータをデータベースに挿入する作業を頻繁に行うものですが、レコードセットの変更は、AddNew メソッドまたは Delete メソッドの形式で、または単純にフィールド・データを変更することによって実行します。そしてレコードセットに対するこれらの変更は、Update メソッドを使用してデータベースに送られます。

ASAProv プロバイダを使用してレコードセットを更新するときは、適切なロック・タイプを使用してください。また、adLockReadOnly は ADOCE レコードセットのデフォルトで、レコードの更新に使用するロック・タイプではないことに注意してください。ロック・タイプを指定しなければ、レコードセットを更新することはできません。

次の UpdateRS というコードは、データを更新し、新しいレコードをレコードセットに挿入する方法を示しています。図 6 は、新しいレコードが正常に挿入されたときに表示されるメッセージです。

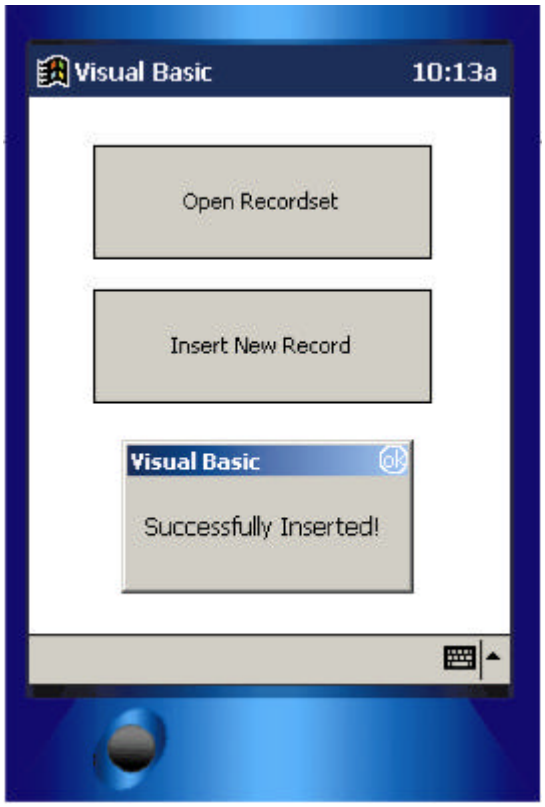


図 6：レコードセットの更新

```
Option Explicit
Dim Connect As ADOCE.Connection
Dim RS As ADOCE.Recordset
Dim SQLstring As String

Private Sub cmdOpenRS_Click()

  Set Connect = CreateObject("ADOCE.Connection.3.0")
  Connect.Provider = "ASAProv"
  Connect.ConnectionString = "Data Source=ASA 7.0 Sample"
  Connect.Open

'Order by statements or queries with criteria can have attributes
'which make them 'non-updatable' based on the ANSI standard.
'The following option can be set to circumvent/override the non-
'updatability.
'For further information on the option below, see SET OPTION
'reference in the help file.
Connect.Execute & _
     ("set temporary option ANSI_UPDATE_CONSTRAINTS='OFF'")

  SQLstring = "Select * from Product order by id"

  Set RS = CreateObject("ADOCE.Recordset.3.0")

  RS.Open SQLstring, Connect, adOpenKeyset, _

                adLockOptimistic, adCmdText
  RS.Fields("name") = "Free Stuff"
  RS.Update
  MsgBox RS.Fields("name")

End Sub

Private Sub cmdInsert_Click()

  Set RS = CreateObject("ADOCE.Recordset.3.0")

  RS.Open "select * from department", Connect, & _
      adOpenKeyset, adLockOptimistic, adCmdText
  RS.AddNew
  RS.Fields("dept_id") = "606"
  RS.Fields("dept_name") = "Accounting"
  RS.Fields("dept_head_id") = "1090"
```

```
  RS.Update
  MsgBox "Successfully Inserted!"

End Sub
```

トランザクションの使用

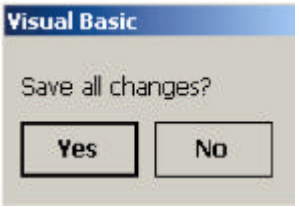
上記の例は、オートコミットを使用してレコードセットを更新する方法を示しています。オートコミットはデフォルトの設定で、COMMIT コマンドを入力しなくてもすべての変更が自動的にコミットされるます。ユーザに変更をロールバックする機会を与えるのが望ましい状況では、ADOCE はトランザクションを使用して手動によるコミットが必要なため、手動によるコミット機能も提供してます。これには BeginTrans メソッド、CommitTrans メソッド、RollbackTrans メソッドを使用します。

- ginTrans　アプリケーションのオートコミット・モードを解除します。次のメソッド、CommitTrans を呼び出すまでは、変更はデータベースにコミットされません。
- CommitTrans　このメソッドを呼び出すと、変更がデータベースにコミットされます。
- RollbackTrans　変更をデータベースにコミットしません。BeginTrans メソッドを呼び出した後の変更を行う前の状態にデータベースがロールバックされます。

BeginTrans を呼び出す前に CommitTrans や RollbackTrans を呼び出すと、エラーになります。まず BeginTrans を呼び出さなければ、他の 2 つのメソッドは使用できません。

次のコード・サンプルは、これら 3 つのメソッドでの呼び出しの使用方法を示しています。オートコミット・モードを解除する目的で、このホワイトペーパーに掲載されているすべての例でこのサンプルを使用できます。

```
Dim Connect As ADOCE.Connection
Dim RS As ADOCE.Recordset
'Insert code here to open a connection to a data source
'and open the table in the data source, ASA 7.0 Sample
Connect.BeginTrans
'Insert data collection code here.
If MsgBox("Save all changes?", vbYesNo) = vbYes Then
  Connect.CommitTrans
Else
  Connect.RollbackTrans
End If
```



さまざまなコントロールを使用したデータ表示

Embedded Visual Basic には、データベースからデータを表示するときに使用できる一連のデータ・アクセス・コントロール・コンポーネントが含まれています。

Embedded Visual Basic では、次のデータ・アクセス・コントロールを使用してデータを表示できます。

- Grid コントロール
- ListBox コントロールと ComboBox コントロール

コントロールをプロジェクトに追加する手順については、ホワイトペーパー『ADO と Visual Basic を使用した Adaptive Server Anywhere のデータへのアクセス』の「さまざまなコントロールを使用したデータ表示」を参照してください。

Grid コントロール

グリッドは、グリッドまたはテーブルのフォーマットでデータを表示します。ADOCE 用の Grid コントロールは、データ・タイプの長いものはサポートしていません。ただし、数値データ・タイプはサポートしているため、長いデータ・タイプは数値として表示します。

グリッドを構成するには、次の手順に従います。

1 コントロールをツールボックスに追加します。

2 グリッドをフォームにドラッグ・アンド・ドロップします。

ADO とは異なり、ADOCE にはデータをグリッドにバインドする手段はなく、表示するだけです。グリッドが表示された後にユーザがグリッドを更新する手段もありません。ADOCE では DataSource プロパティがサポートされていないため、レコードセット全体をループしてデータベースをグリッドに表示しなければなりません。次の DisplayData というタイトルのプロジェクトは、このデータ表示の方法を示しています。

```
Option Explicit
Dim Connect As ADOCE.Connection
Dim RS As ADOCE.Recordset
Dim SQLstring As String
Dim i As Integer

Private Sub cmdshowdata_Click()
   'Make the connection
   Set Connect = CreateObject("ADOCE.Connection.3.0")
   Connect.Provider = "ASAProv"
   Connect.ConnectionString = "Data Source=ASA 7.0 Sample"
   Connect.Open

   SQLstring = "Select * from employee"

   'Connect the recordset to the database
   Set RS = CreateObject("ADOCE.Recordset.3.0")
   RS.Open SQLstring, Connect, adOpenKeySet, _
                   adLockOptimistic, adCmdText

  GridCtrl1.Row = 1
  GridCtrl1.Rows = 1
  GridCtrl1.Cols = 14

    Do While Not RS.EOF
        GridCtrl1.Col = 0
        'Display the fields
        For i = 0 To 12
            GridCtrl1.Text = RS.Fields(i).Value
            GridCtrl1.Col = i + 1
        Next
        'Display the last field
        GridCtrl1.Text = RS.Fields(13).Value
        'Move to the next row
        GridCtrl1.Rows = GridCtrl1.Rows + 1
        GridCtrl1.Row = GridCtrl1.Row + 1
        RS.MoveNext
```

```
    Loop
End Sub

Private Sub cmdDisconnect_Click()
    RS.Close
    Set RS = Nothing
    Connect.Close
    Set Connect = Nothing
End Sub

Private Sub cmdDisconnect_Click()
    Connect.Close
    Set Connect = Nothing
End Sub
```

データをグリッドに表示のは、小容量のクエリのみに適しています。これは、大容量の結果セットを返すクエリは表示にかなりの時間がかかるためです。下記の例では、次のクエリを使用しています。

```
SELECT * from employee
```

このクエリは、employee テーブルが比較的小さい場合にのみ適しています。データを表示するとき、どのクエリが妥当かを判断するのは、ユーザ次第ですが、Windows CE Pocket PC デバイスの性能に適したクエリのデータのみを表示することをお勧めします。図 7 は、employee テーブルのクエリから得たデータをグリッドに表示したものです。

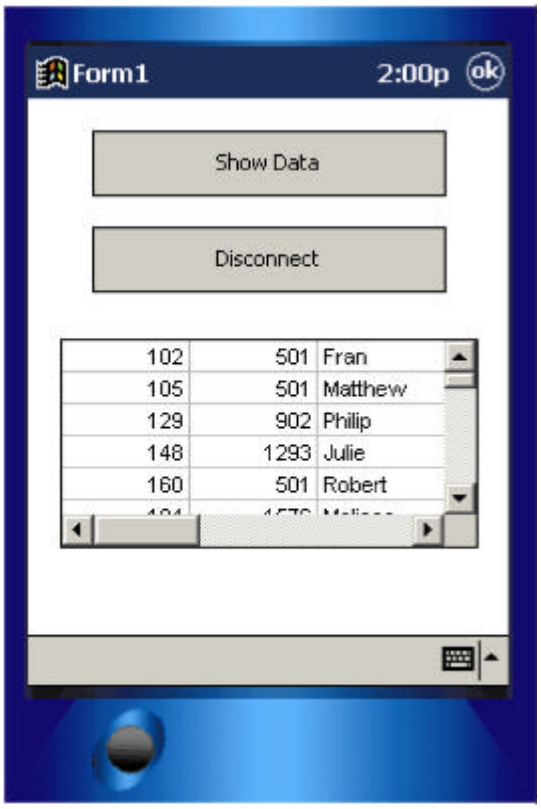


図 7：データのグリッド表示

ListBox コントロールと ComboBox コントロール

ListBox と ComboBox は、Windows CE Control のコンポーネントです。ListBox コントロールと ComboBox コントロールは、ADO の DataList コントロールと ComboList コントロールに相当します。図 8 の ListBox には一度に 1 カラムが表示されるのに対し、ComboBox にはドロップダウン・ボックスにリストが表示されます。ListBox コントロールと ComboBox コントロールは、新しいプロジェクトを作成した時点でツールボックスに含まれます。

次の DataList という例では、ListBox コントロールと ComboBox コントロールの両方を利用します。

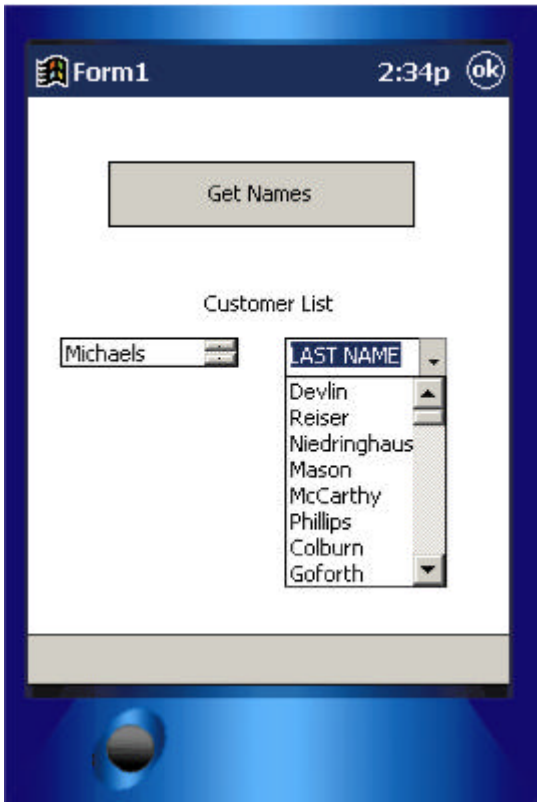


図 8：コンボ・ボックスとリスト・ボックスのデータ表示

```
Private Sub cmdFillList_Click()
   Dim Connect As ADOCE.Connection
   Set Connect = CreateObject("ADOCE.Connection.3.0")
   Dim RS As ADOCE.Recordset
   Set RS = CreateObject("ADOCE.Recordset.3.0")
   Dim SQLstring, ConnString As String

   ConnString = "Provider=ASAProv;" & _
        Data source=ASA 7.0 Sample;" & _
```

```
        "User Id = dba; Password=sql;"
    Connect.Open ConnString

    SQLstring = "select fname, lname from customer"

    RS.Open SQLstring, Connect, adOpenKeyset

    Do While Not RS.EOF
    List1.AddItem (RS.Fields("fname").Value)
    Combo1.AddItem (RS.Fields("lname").Value)
     RS.MoveNext
     Loop
End Sub
```

レコードセットとレコードセットのページのカウント

レコードセットの RecordCount プロパティは、Recordset オブジェクト内のレコード数をカウントするときに使用します。これはレコードセットを開く際に使用するカーソル・タイプに依存しており、ASAProv レコードセットでは、RecordCount プロパティを使用すると、カーソル・タイプごとの推定レコード数が得られます。ただし adOpenForwardOnly は例外で、-1 という値になります。

RecordCount プロパティの詳細については、ホワイトペーパー『ADO と Visual Basic を使用した Adaptive Server Anywhere のデータへのアクセス』の「レコードセットとレコードセットのページのカウント」を参照してください。

レコードセットの PageCount プロパティは、レコードセットのページ数をカウントする際に使用します。PageCount の動作は RecordCount と同様です。RecordCount と PageCount の構文は次のとおりです。

```
RS.RecordCount

RS.PageCount
```

これら 2 つのプロパティの詳細については、MSDN ライブラリを参照してください。

図 9 に示す AbsolutePosition プロジェクトは、WHILE ループを使用してレコードをカウントし、最大レコード数をレコードセットの RecordCount プロパティに設定します。このプロジェクトに含まれている [Support] ボタンを使用すると、指定したロック・タイプを使用した場合にブックマークがサポートされるかどうか確認できます。

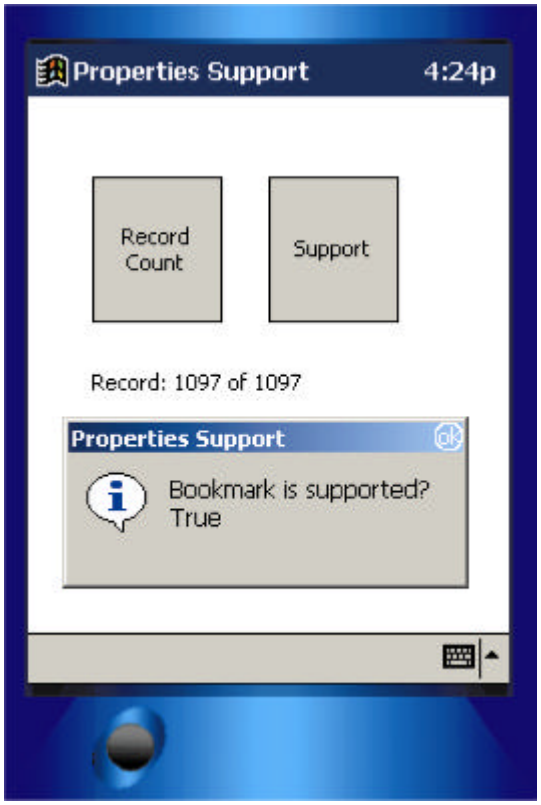


図 9：Absolute Position プロジェクト

```
Option Explicit

Dim Connect As ADOCE.Connection

Private Sub cboSupport_Click()

Call Support
End Sub

Private Sub cmdCount_Click()
Dim RS As ADODB.Recordset
Dim sMessage As String

If Connect.State <> adStateOpen Then

Set Connect = CreateObject("ADOCE.Connection.3.0")
Connect.Provider = "ASAProv"
Connect.ConnectionString = "uid=dba;pwd=sql;dbf=" & _
     App.Path & "\asademo.db"
Connect.Open
End If

Set RS = CreateObject("ADOCE.Recordset.3.0")
RS.Open "DBA.sales_order_items", Connect, & _
     adOpenKeyset, adLockReadOnly, adCmdTable

Do While Not RS.EOF
    lblRecordCount.Caption = "Record: " & RS.AbsolutePosition & _
    " of " & RS.RecordCount
   RS.MoveNext
Loop

End Sub

Private Sub Support()

Dim adoSupport As ADOCE.Recordset
Set adoSupport = CreateObject("ADOCE.Recordset.3.0")
adoSupport.Open "customer", Connect, adOpenForwardOnly, & _
     adLockReadOnly, adCmdTable
MsgBox "Bookmark is supported? " & _
     adoSupport.Supports(adBookmark), vbOKOnly + & _
     vbInformation, "Properties Support"
```

```
adoSupport.Close
Set adoSupport = Nothing

End Sub

Private Sub cmdSupport_Click()
Call Support
End Sub

Private Sub Form_Load()

Set Connect = CreateObject("ADOCE.Connection.3.0")
Connect.Provider = "ASAProv.70"
Connect.ConnectionString = "Data source = ASA 7.0 Sample"
Connect.Open

End Sub
```

レコードセット内の移動

レコードセット内の移動については、ホワイトペーパー『ADO と Visual Basic を使用した Adaptive Server Anywhere のデータへのアクセス』の「レコードセット内の移動」を参照してください。この手順は ADOCE でも ADO でも同じです。

ストアド・プロシージャ

ストアド・プロシージャは、クライアント・アプリケーションではなくデータベース内に保存されます。クライアント・アプリケーションは、ストアド・プロシージャを繰り返し呼び出して、特定の関数に自動的にアクセスできます。

Transact-SQL と Adaptive Server Anywhere の互換性については、Adaptive Server Anywhere のマニュアルの次の項を参照してください。

- Adaptive Server Anywhere バージョン 7 を使用している場合は、『ASA User's Guide』の第 6 部「The Adaptive Server Family」の第 32 章「Transact-SQL compatibility」。
- Adaptive Server Anywhere バージョン 8 を使用している場合は、『ASA SQL User's Guide』の第 12 章「Transact-SQL Compatibility」。

Connection プロジェクトは、Recordset Open メソッドでストアド・プロシージャ sp_retrieve_contacts を呼び出すクライアントの機能を示します。図 10 を参照してください。

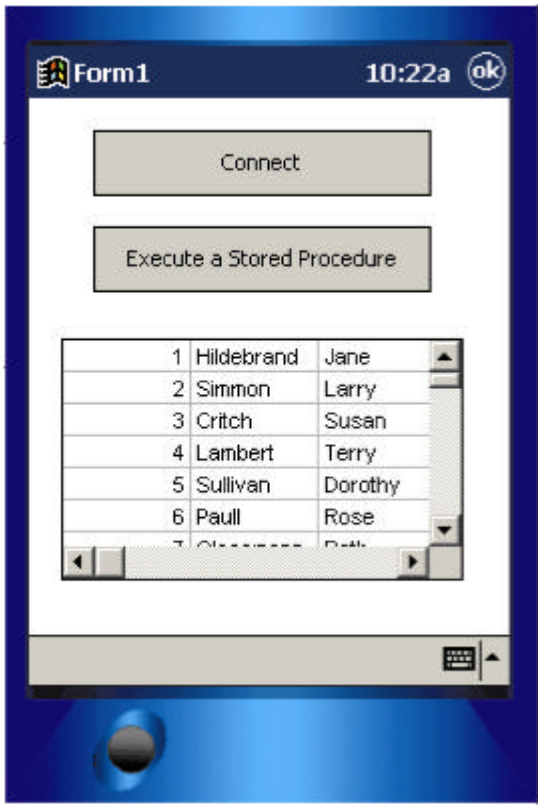


図 10：ストアド・プロシージャの実行

```
Option Explicit

Dim Connect As ADOCE.Connection
Dim RS As ADOCE.Recordset
Dim i As Integer

Private Sub cmdConnect_Click()
    Set Connect = CreateObject("ADOCE.Connection.3.0")
    Connect.Provider = "ASAProv.70"
    Connect.ConnectionString = "Data Source = ASA 7.0 Sample"
    Connect.Open
End Sub

Private Sub cmdExecuteProcedure_Click()

Set RS = CreateObject("ADOCE.Recordset.3.0")
RS.Open "call sp_retrieve_contacts()", Connect, & _
     adOpenKeyset, adLockReadOnly,adCmdText
RS.MoveFirst

    GridCtrl1.Row = 1
    GridCtrl1.Rows = 1
    GridCtrl1.Cols = 10

  Do While Not RS.EOF
       GridCtrl1.Col = 0

       'Display the fields
       For i = 0 To 8
       GridCtrl1.Text = RS.Fields(i).Value
       GridCtrl1.Col = i + 1
       Next
       'Display the last field
       GridCtrl1.Text = RS.Fields(9).Value
       'Move to the next row
       GridCtrl1.Rows = GridCtrl1.Rows + 1
       GridCtrl1.Row = GridCtrl1.Row + 1
       RS.MoveNext
   Loop

End Sub
```

```
Private Sub ErrorHandler()
Dim strErr As String
    For Each objError In Connect.Errors
        strErr = strErr & " Description : " & _
        objError.Description & vbCrLf & vbCrLf & _
        "     SQL CODE : " & objError.NativeError & _
        vbCrLf & vbCrLf & _
        "    SQL STATE : " & objError.SQLState & vbCrLf
    Next
    MsgBox strErr, vbCritical + vbOKOnly, "Error Connecting"

End Sub
```

レコードセットを閉じる

レコードセットをデータベースから閉じるには、次のコマンドを実行します。

```
RS.Close
Set RS = Nothing
```

Grid コントロールなどのコントロールにバインドしているデータは、ADO の場合とは異なり、'nothing' に設定できず、次の文は ADOCE ではサポートされていません。

```
Set GridCtrl1 = Nothing
```

このコマンドを使用するとエラーが表示されます。Recordset オブジェクトと Connection オブジェクトを閉じると、グリッドは暗黙的に閉じられます。

次の Disconnect という例は、データベースに接続し、Recordset オブジェクトを開いて、データベースから切断する適切な方法をまとめたものです。

適切な切断画面を図 11 に示します。

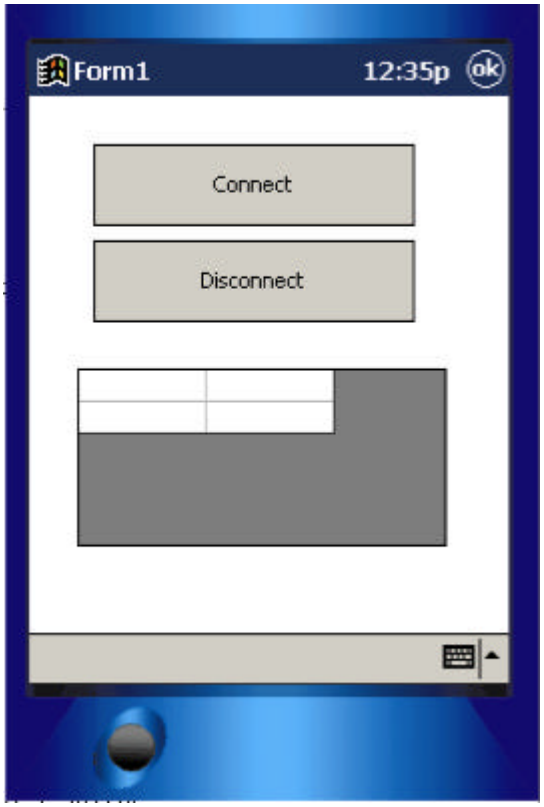


図 11：適切な切断

```
Dim Connect As ADOCE.Connection
Dim ConnString As String
Dim RS As ADOCE.Recordset
Dim SQLstring As String
Dim I As Integer

Private Sub cmdConnect_Click()
    Set Connect = CreateObject("ADOCE.Connection.3.0")
    ConnString = "Provider=ASAProv;" & _
          "Data Source=ASA 7.0 Sample;" & _
          "User Id=dba; Password=sql;"
     SQLstring = "Select * from Customer"

     'Make the connection
     Connect.Open ConnString

     'Open Recordset Object
     Set RS = CreateObject("ADOCE.Recordset.3.0")
     RS.Open SQLstring, Connect, adOpenKeyset

  'Display the table
    GridCtrl1.Row = 1
    GridCtrl1.Rows = 1
    GridCtrl1.Cols = 9

  Do While Not RS.EOF

        GridCtrl1.Col = 0
        'Display the fields
        For i = 0 To 7
       GridCtrl1.Text = RS.Fields(i).Value
        GridCtrl1.Col = i + 1
       Next
        'Display the last field
        GridCtrl1.Text = RS.Fields(8).Value
        'Move to the next row
        GridCtrl1.Rows = GridCtrl1.Rows + 1
        GridCtrl1.Row = GridCtrl1.Row + 1
       RS.MoveNext
    Loop
End Sub

Private Sub cmdDisconnect_Click()
```

```
'Close the recordset object
RS.Close
Set RS = Nothing
'Close the connection object
Connect.Close
Set Connect = Nothing
End Sub
```

Blob

ASAProv は ADOCE の blob をサポートします。次のコード例は、blob.db というデータベースで blob を使用する方法を示しています。このプロジェクトを使用するには、Blob.DSN をデバイスのデスクトップに配置するか、または Blobemul.DSN (この名前を Blob.DSN に変更しなければプロジェクトは適切に機能しません) をエミュレータのデスクトップに配置すると同時に、イメージ・ファイルを Blob プロジェクトに含める必要があります。Windows エクスプローラを使用してイメージ・ファイルをコピーする方法については、47 ページの「付録 C：Windows エクスプローラの使用による、デスクトップからデバイスまたはエミュレータへのナビゲーション」を参照してください。さらに、blob.db をエミュレータの Adaptive Server Anywhere フォルダに配置してください。

このプロジェクトでは ADOCE 3.1 Library を使用するため、このプロジェクトでは次のライブラリを参照してください。

- Microsoft CE ADO Control 3.1
- Microsoft CE ADO Ext. 3.1 for DLL

これらのライブラリの参照方法については、10 ページの「Embedded Visual Basic: ADOCE コントロールの参照」を参照してください。

Blob プロジェクトを使用するには、次の手順に従います。

1 [Connect] ボタンをクリックして Blob データベースに接続します。

2 [Display Image] ボタンの隣のフィールドにファイル・ロケーションを入力し、[Display Image] をクリックして、デバイスまたはエミュレータからイメージを表示します。ピクチャ・ボックスにイメージが表示されます。

31 ～ 3 の数字を入力して [Slow Fetch] ボタンまたは [Fast Fetch] ボタンをクリックすると、データベースにすでに保存済みのイメージを表示できます。[Slow Fetch] ボタンはネイティブの Embedded Visual Basic を使用するのに対して、[Fast Fetch] ボタンはダウンロード可能なオブジェクトを使用します。このオブジェクトのダウンロード方法は、プロジェクトのコメントで説明します。

4 [Add Using AddNew] ボタンをクリックすると、データベース内のレコードの別の部分を挿入できます。

3.0 Library を使用してこのプロジェクト全体を機能させることはできません。データベースに接続し、ファイルからイメージを表示して、データベースからイメージをフェッチすることはできます。しかし、3.0 Library を使用している場合は、新しいイメージをデータベースに追加することはできません。

図 12 は Blob プロジェクトを、また次のコードはこのプロジェクトの背景にある論理を示しています。Blob プロジェクトの冒頭にあるコメントは、Blob サンプル・ディレクトリの BlobReadme.txt ファイルにもあります。

図 12：Blob の使用

```
'How to run the Blob Program
'
'In order to run a Fast Fetch you must register and install the
'correct OSIUtil.dll
'
'The steps below outline how to do this:
'
'1) Go to
```

```
http://www.microsoft.com/mobile/developer/technicalarticles/osiuti
"l.asp
'or the actual company web site at
'http://www.odysseysoftware.com/index.shtml
'
'2) Download OSIUtil100.exe "OSI Utility Collection"
'
'3) Install this program on your computer
'
'4) Copy the OSIUtil.dll file to the windows directory of your
device or emulator.
'
'5) Now register the appropriate OSIUtil.dll from the install
'directory using regsvrCE if
'   you are going to put the dll onto a CE device. If you do not
'   have regsvrCE utility on the device, you can find it in the
'   Windows CE Tools kit folder. Copy this file to the windows
'  directory on the device, then run regsvrCE via the
'   File Explorer which will prompt for the location of the
'   osiutil.dll library.
'
'You should now be able to use the FastFetch button in the Blob
'Program

Option Explicit
Dim RecordsAffected
Dim Connect As ADOCE.Connection
Dim RS As ADOCE.Recordset

Private Sub cmdADO_Click()
    On Error Resume Next
    Dim laImage As Variant

    'get Picture to add
    ceFile.Open txtImage.Text, fsModeInput, fsAccessRead
    laImage = ceFile.InputB(ceFile.LOF)
   ceFile.Close
```

```
    'add picture
    Set RS = CreateObject("ADOCE.RecordSet.3.1")
    RS.Open "Pictures", Connect, adOpenKeyset, adLockOptimistic, _
      adCmdTable
    RS.AddNew
    RS.Fields("picture") = laImage
    RS.Update

    'error check
    If (Err.Number <> 0) Then
       Err.Clear
        MsgBox "Unable to insert value "
    Else
        MsgBox "Successfully inserted"
    End If

    RS.Close

End Sub

Private Sub cmdConnect_Click()
    On Error Resume Next

    'Create the ADOCE connection object
    Set Connect = CreateObject("ADOCE.connection.3.1")
    If (Err.Number <> 0) Then
        MsgBox "Unable to create object " & Err.Number
       Err.Clear
       Exit Sub
    End If

    'Open the ASA provider
    Connect.Provider = "ASAProv.70"
    Connect.ConnectionString = "Data Source=blob"
    Connect.Open
    If (Err.Number <> 0) Then
        MsgBox "Unable to use the Data Source! Error Code " & _
       Err.Number
       Err.Clear
    End If

End Sub

Private Sub cmdFastFetch_Click()
```

```
    Dim File, Data
    Dim strSQL As String
    Dim Factory

    'search for image number in db
    Set RS = CreateObject("ADOCE.Recordset.3.1")
    strSQL = "select picture from pictures where id = " & _
    CInt(txtImageID.Text)
    RS.Open strSQL, Connect, adOpenKeyset, adLockReadOnly

    'if no records have the id of the number specified
    If RS.BOF And RS.EOF Then
        MsgBox "No Records Found!"
        Exit Sub
    End If

    'Get image and display it
    Set Factory = CreateObject("OSIUtil.Win32")
    Set File = Factory.CreateObject("OSIUtil.File")

    Data = RS.Fields(0)

    File.VariantByteArrayToFile "\temp.bmp", Data
    pctImage.Picture = "\temp.bmp"

    If (Err.Number <> 0) Then
        Err.Clear
        MsgBox "failed to fetch image from database"
    End If

    RS.Close
    Set RS = Nothing
    Set File = Nothing

    'MsgBox "File copied." & vbCrLf & "Size = " & _
    '(UBound(Data) - LBound(Data) + 1) & " bytes."

End Sub

Private Sub cmdFetchFromDB_Click() 'THE OLD WAY
    On Error Resume Next
```

```
    Dim strSQL
    Dim blob As Variant
    Dim lsImage As String
    Dim I As Integer

    'search for picture with id number in the textbox
    Set RS = CreateObject("ADOCE.Recordset.3.1")
    strSQL = "select picture from pictures where id = " & _
    CInt(txtImageID.Text)
    RS.Open strSQL, Connect, adOpenKeyset, adLockReadOnly

    'if no record exists with the specified id
    If RS.BOF And RS.EOF Then
        MsgBox "No Records Found!"
        Exit Sub
    End If

    'load and display image from db
    ceFileOut.Open "\temp.bmp", fsModeOutput, fsAccessWrite
    blob = RS.Fields(0)

    For I = 1 To LenB(blob)
        lsImage = lsImage & Chr(AscB(MidB(blob, I, 1)))
    Next

    ceFileOut.LinePrint lsImage
    ceFileOut.Close

    pctImage.Picture = "\temp.bmp"
    If (Err.Number <> 0) Then
        Err.Clear
        MsgBox "failed to fetch image from database"
    End If

    RS.Close
    Set RS = Nothing

End Sub

Private Sub cmdLoadImage_Click()
    'display image
    pctImage.Picture = txtImage.Text
End Sub
```

```
Private Sub Form_Load()
End Sub

Private Sub Form_OKClick()
    On Error Resume Next

    'close the Connect connection if it is open
    If (Connect.State = adStateOpen) Then
       Connect.Close
    End If

    App.End
End Sub
```

Fields コレクション

Fields コレクションは Field オブジェクトと混同しないでください。これまでも、Fields コレクションを使用してデータを表示していました。Recordset オブジェクトがこのコレクションを使用します。このコレクションには、レコードセット内の 1 カラムにつき 1 つの Field オブジェクトが含まれています。次のように、カラム番号または名前を基準として、特定のフィールド・オブジェクトを参照できます。

```
RS.Fields(1)
RS.Fields("manager_id")
```

Fields コレクションは Count プロパティをサポートするため、レコードセット内のフィールド数をカウントできます。これはレコードセットを表示する際に便利です。次のコマンドは、Count プロパティを使用してレコードセット内のフィールド数を表示します。

```
MsgBox RS.Fields.Count
```

レコードセットをループしてカラム名を表示するときは、Count プロパティを上限値に使用することもできます。次のコードはループに使用します。

```
For n = 0 To RS.Fields.Count - 1
     MsgBox RS.Fields(n).Name
Next
```

Field オブジェクト

Field オブジェクトにはイベントはなく、2 つのメソッドと多数のプロパティがあるだけです。Value と ActualSize を例外として、Field オブジェクトは読み込み専用です。Field オブジェクトは常に既存のレコードセットの Fields コレクション内にあるフィールドを参照するため、直接作成されることはありません。

Field オブジェクトのインスタンスを作成するには、次のコマンドを使用します。

```
Dim f As ADOCE.Field
Set f = RS.Fields(カラム番号またはカラム名)
```

Name プロパティの使用

図 13 のように、特定のフィールドが所属するカラムをテキストボックスにヘッダとして表示するときは、カラムの名前を読み込む方が簡単です。Field オブジェクトの値をメッセージ・ボックスに表示するとき、Field オブジェクトが所属するカラムの名前のボックスのヘッダに表示できます。

次の NameProperty というプロジェクトでこれを示します。

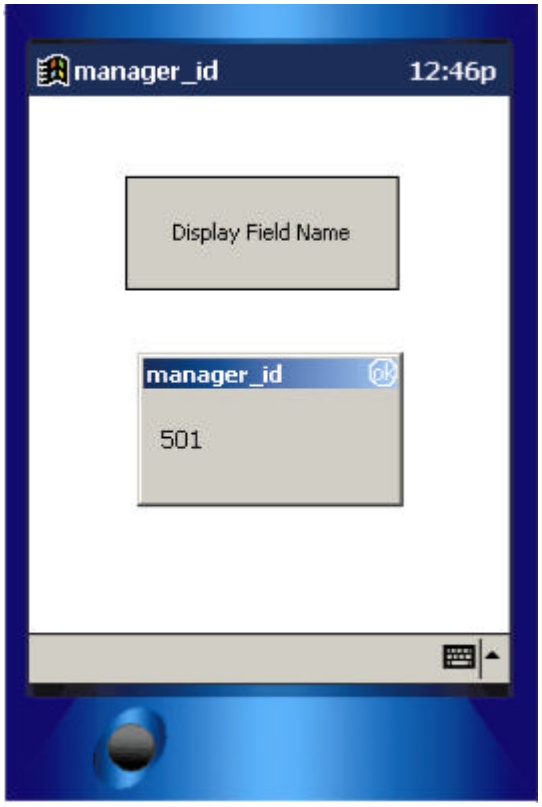


図 13：Name プロパティ

```
Option Explicit

Private Sub Form_OKClick()
   App.End
End Sub

Private Sub Command1_Click()

Dim Connect As ADOCE.Connection
Dim RS As ADOCE.Recordset
Dim fld As ADOCE.Field

'Make the connection
Set Connect = CreateObject("ADOCE.Connection.3.0")
Connect.Provider = "ASAProv"
Connect.ConnectionString = "Data Source = ASA 7.0 Sample"
Connect.Open

'Open the recordset
Set RS = CreateObject("ADOCE.Recordset.3.0")
RS.Open "select * from employee", Connect, & _
    adOpenKeySet, adLockReadOnly, adCmdText

'Set the field object to a field in the recordset
Set fld = RS.Fields(1)
'Display the field object's value and the column to which it

'belongs
MsgBox fld.Value, , fld.Name

End Sub
```

Field オブジェクトを使用したデータ表示

DisplayData プロジェクトを再び取り上げて Field オブジェクトを使用すると、Field オブジェクトの機能を十分に確認することができます。Field オブジェクトを使用すると、ActualSize プロパティを利用して、各フィールドに必要な大きさが特定できます。このテストを完了すると、次のコマンドを使用してカラムを正しいサイズに設定できます。

```
GridCtrl1.ColWidth(column #) = ActualSize * 100
```

次の 2 つの例は、レコードセット内の各フィールド・オブジェクトの ActualSize を表示する方法と、それに応じてグリッドのカラム・サイズを操作する方法を示しています。2 つのうちの 1 つ目は ActualSize というプロジェクトです。これを図 14 に示します。

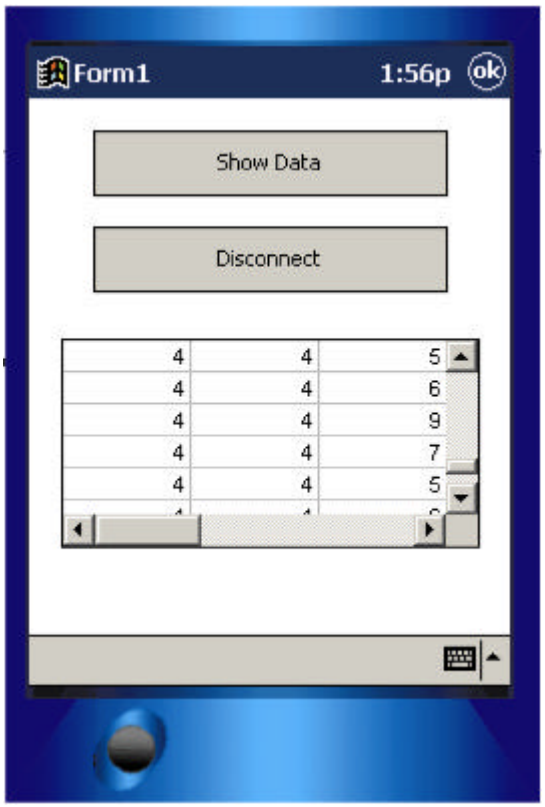


図 14：ActualSize プロジェクトの使用

```
Option Explicit
Dim Connect As ADOCE.Connection
Dim RS As ADOCE.Recordset
Dim SQLstring As String
Dim i As Integer
Dim fld As ADOCE.Field

Private Sub cmdshowdata_Click()
   'Make the connection
   Set Connect = CreateObject("ADOCE.Connection.3.0")
   Connect.Provider = "ASAProv"
   Connect.ConnectionString = "Data Source=ASA 7.0 Sample"
   Connect.Open

   SQLstring = "Select * from employee"

   'Connect the recordset to the database
   Set RS = CreateObject("ADOCE.Recordset.3.0")
   RS.Open SQLstring, Connect, adOpenKeySet, _
                   adLockOptimistic, adCmdText

  GridCtrl1.Row = 1
  GridCtrl1.Rows = 1
  GridCtrl1.Cols = 14
  'The column sizes to be specified later
  'GridCtrl1.ColWidth(0) =
  'GridCtrl1.ColWidth(1) =
  'GridCtrl1.ColWidth(2) =
  'GridCtrl1.ColWidth(3) =
  'GridCtrl1.ColWidth(4) =
  'GridCtrl1.ColWidth(5) =
  'GridCtrl1.ColWidth(6) =
  'GridCtrl1.ColWidth(7) =
  'GridCtrl1.ColWidth(8) =
  'GridCtrl1.ColWidth(9) =
  'GridCtrl1.ColWidth(10) =
  'GridCtrl1.ColWidth(11) =
  'GridCtrl1.ColWidth(12) =
  'GridCtrl1.ColWidth(13) =

    Do While Not RS.EOF
        GridCtrl1.Col = 0
        'Display the fields
```

```
        For i = 0 To 12
        Set fld = RS.Fields(i)
        GridCtrl1.Text = fld.ActualSize
        GridCtrl1.Col = i + 1
       Next
        'Display the last field
        Set fld = RS.Fields(13)
        GridCtrl1.Text = fld.ActualSize
        'Move to the next row
        GridCtrl1.Rows = GridCtrl1.Rows + 1
        GridCtrl1.Row = GridCtrl1.Row + 1
       RS.MoveNext
    Loop
End Sub

Private Sub cmdDisconnect_Click()

    RS.Close
    Set RS = Nothing

    Connect.Close
    Set Connect = Nothing
End Sub
```

上の図からも、グリッドでレコードセット全体をスクロールしてもわかるように、すべてのカラムには、実際のサイズとして設定可能な範囲があります。この情報をもとに、カラムの最大サイズに合わせてカラムのサイズを指定できます (ActualSize*100 が適しているようです)。たとえば、第 1 カラムは、どのフィールドにも 4 という値が表示されています。この第 1 カラムのサイズは 400 に設定できます。ここで、この新しい情報をコードに入力し、Value プロパティを使用して Field オブジェクトの値を表示することができます。

全く同じ結果になるため、この更新後のプロジェクトは、元の DisplayData プロジェクトと類似していますが、わずかに異なる方法を使用しています。図 15 と図 7 を比較すると、第 2 の方法を使用する効果がわかります。図 15 では、グリッド内の限られた領域がより効率的に使用されています。この更新後のプロジェクトのタイトルは ImprovedDD です。

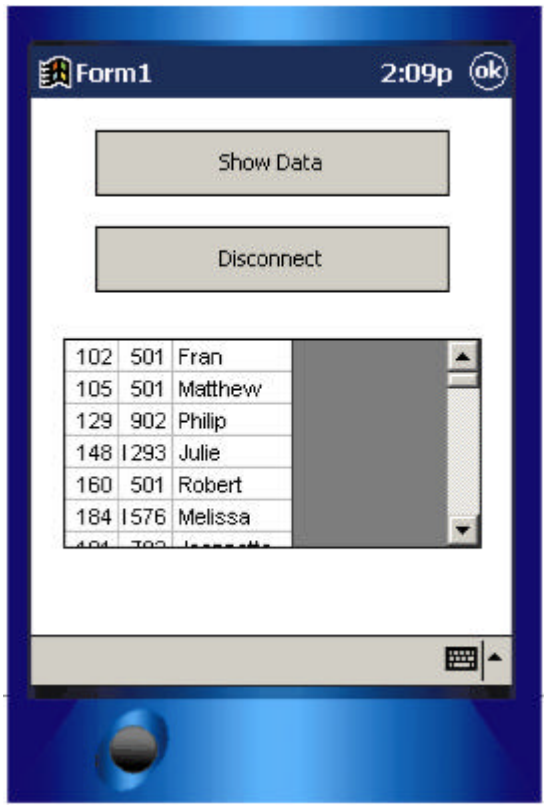


図 15：Field オブジェクトを使用したデータ表示

図 15 からわかるように、すべてのデータが各カラムに収まり、すべてのカラムを左右にスクロールせずに一度に表示できます。これにより、グリッドから無駄な領域を削除し、一度により多くのカラムを表示できます。これは、多数のカラムを含むレコードセットを表示するときに大きな違いが出ます。

Error オブジェクト

Error オブジェクトはそれぞれ Errors コレクション内に存在します。コードでエラーが発生すると、エラーに固有の Error オブジェクトが Errors コレクションに配置されます。Errors コレクションは、Connection オブジェクトのプロパティです。ADOCE では、Error オブジェクトは読み込み専用です。関連付けられているメソッドやイベントはありません。一方で、Errors コレクションは、Clear メソッドを使用してクリアできます。次の呼び出しは、ADOCE ではできません。

```
On Error Go To ErrorHandler
```

ADOCE でサポートされている On Error 文は、次の文だけです。

```
On Error Resume Next
```

エラーが発生する前は、すべてのプロパティが NULL を返します。すべてのデータ・ソースに、Error オブジェクトを返す機能があります。

ASAProv でサポートされているプロパティ

Description ：発生したエラーについてユーザに説明するテキスト。

NativeError ：使用しているプロバイダに固有のエラー番号。

Error Number ： 発生したエラーの番号を表わす長い整数 (ErrorValueEnum 定数によって表示される)。

Source ：エラー発生の原因となったアプリケーションまたはオブジェクトの名前を特定。

SQLState プロパティは、ADOCE ではサポートしていません。SQLState プロパティを呼び出そうとすると、ブランクのメッセージ・ボックスが表示されます。SQLState プロパティを含めると、他のプロパティが消去されます。

プロバイダ・エラーに対する Count プロパティの使用

Count プロパティが 1 以上の場合は、Error オブジェクトが Errors コレクションに追加されたため、確認が必要です。Errors コレクション内に存在する Error オブジェクトを見つけるには、コレクション内を検索し、Error オブジェクトのプロパティの形式で情報を取り出す必要があります。

次の呼び出しを行うと、このエラーをテキスト・ボックスに表示できます。

```
MsgBox <エラー文字列>
```

次のコードは、存在しないテーブルを変更するコードを実行したときに発生するプロバイダ・エラーの例です。次の Error というプロジェクトでは、エラー番号、説明、nativeError 番号、報告元のプロバイダの情報がテキスト・ボックスに表示されます (図 16 を参照)。

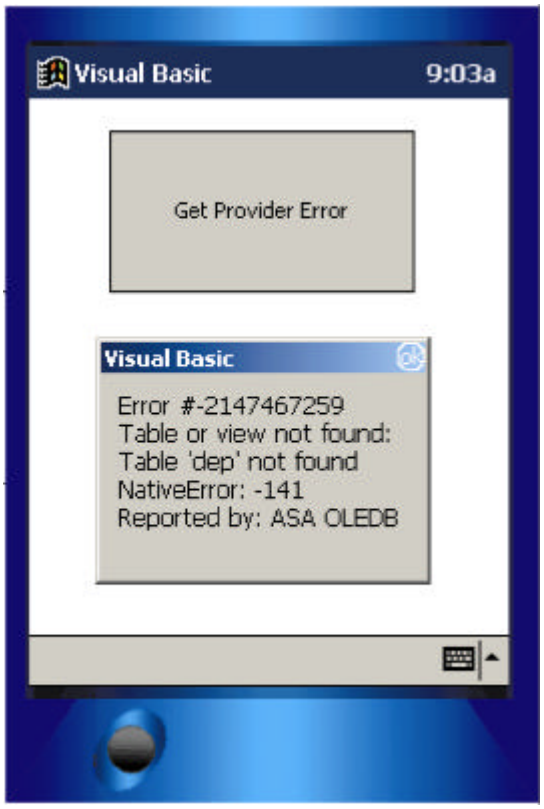


図 16：プロバイダ・エラー

```
Option Explicit

Private Sub Command1_Click()

    Dim Connect As ADOCE.Connection
    Dim RS As ADOCE.Rescordset
    Dim objError As ADOCE.Error

    On Error Resume Next
    Set Connect = CreateObject("ADOCE.Connection.3.0")
    Connect.ConnectionString = "Provider=ASAProv;" & _
         "Data Source = ASA 7.0 Sample"
    Connect.Open

    Set RS = CreateObject("ADOCE.Recordset.3.0")
    Set RS.ActiveConnection = Connect
    ' Create a provider error by trying to update a non-existent
    'table
    RS.Open "update dep set stuff = & _
         " 'asfdsadfsadfdsasd' " & _
         "where id =4", Connect, adLockOptimistic

    'Check if the errors collection has any errors in it
    If Connect.Errors.Count > 0 Then
        'Iterate through and display the errors found
        For Each objError In Connect.Errors
            MsgBox "Error #" & objError.Number & _
                "  " & objError.Description & vbCrLf & _
               "NativeError: " & objError.NativeError & vbCrLf & _
              "Reported by: " & objError.Source & vbCrLf

       Next

    End If
End Sub
```

Errors コレクションのクリア

Connection オブジェクトを使用して別のメソッドを呼び出すときは、必ず Connection オブジェクトの Errors コレクションをクリアしてから、別のメソッドを呼び出してください。これを行うには、次のコマンドを使用します。

```
Connect.Errors.Clear
```

付録 A：プロジェクト

次の表に、このホワイトペーパーで使用したすべてのプロジェクトを示します。iAS_ADOCEprojects を作成して、これらのプロジェクトを zip ファイルから抽出したときに iAS_ADOCEprojects ディレクトリに配置することができます。

プロジェクト名	ファイル名
RSConnect	iAS_ADOCEprojects\RSConnect\RecordSet.ebp
UpdateRS	iAS_ADOCEprojects\UpdateRS\UpdateRS.ebp
DisplayData	iAS_ADOCEprojects\DisplayData\DisplayData.ebp
DataList	iAS_ADOCEprojects\DataList\DataList.ebp
AbsolutePosition	iAS_ADOCEprojects\AbsolutePosition\AbsolutePosition.ebp
Connection	iAS_ADOCEprojects\Connection\Connection.ebp
Blobs	iAS_ADOCEprojects\Blob\Image.ebp
NameProperty	iAS_ADOCEprojects\NameProperty\NameProperty.ebp
ActualSize	iAS_ADOCEprojects\ActualSize\ActualSize.ebp
Improved DD	iAS_ADOCEprojects\ImprovedDD\ImprovedDD.ebp
Error	iAS_ADOCEprojects\Error\Error.ebp
SyncSample	iAS_ADOCEprojects\SyncSample\SyncSample.ebp

付録 B：同期の実行

Embedded Visual Basic アプリケーション内から dbmlsync を使用して同期を実行する方法について、これまでに多くの質問が寄せられました。同期を正常に実行するための 1 つの方法を説明するために、次のコード・サンプルを掲載します。

```
Option Explicit

Public Declare Function TerminateProcess Lib "Coredll" _
    (ByVal hProcess As Long, ByVal uExitCode As Long) As Long
Public Declare Function CreateProcess Lib "coredll.DLL" _
    Alias "CreateProcessW" _
    (ByVal lpApplicationName As String, _
    ByVal lpCommandLine As String, _
    ByVal lpProcessAttributes As Long, _
    ByVal lpThreadAttributes As Long, _
    ByVal bInheritHandles As Long, _
    ByVal dwCreationFlags As Long, _
    ByVal lpEnvironment As Long, _
    ByVal lpCurrentDirectory As Long, _
    ByVal lpStartupInfo As Long, _
    ByVal lpProcessInformation As String) As Long

Public Function MemStringToLong(StringIn As String) As Long
      On Error Resume Next
      Dim hWorkVal As String
      '
       ' Convert the String back to Long Integer.
       ' Converting back to Big Endian format.

     Dim i As Long
     For i = 4 To 1 Step -1
      hWorkVal = hWorkVal & Hex(AscB(MidB(StringIn, i, 1)))
     Next I
      '
       ' Return Long Integer value.
       MemStringToLong = CLng("&H" & hWorkVal)
  End Function

Public Sub getPROCESS_INFORMATION( _
   ByVal sPROCESS_INFORMATION As String, _
   ByRef hProcess As Long, ByRef hThread As Long, _
   ByRef dwProcessId As Long, ByRef dwThreadId As Long)
```

```
   '
   ' Convert memory-formatted String back to Long Integer.
   hProcess = MemStringToLong(MidB(sPROCESS_INFORMATION, 1, 4))
   hThread = MemStringToLong(MidB(sPROCESS_INFORMATION, 5, 4))
   dwProcessId = MemStringToLong(MidB(sPROCESS_INFORMATION, 9, 4))
   dwThreadId = MemStringToLong(MidB(sPROCESS_INFORMATION, 13, 4))

End Sub

Public Function LongToMemoryString( _
    ByVal lInputValue As Long) As String

    Dim hWorkVal As String
    Dim n As Long
    Dim i  As Long
    '
    ' Convert to HEX value.

    hWorkVal = Hex(lInputValue)

    '
    ' Check to see if it is not zero.
    If hWorkVal <> "0" Then
       '
       ' Convert to memory storage format (Little
       ' Endian).  For example, 0000A411 would convert
       ' to 11A40000.
       '
       ' Place leading zeros in 8 character sequence
       ' to maintain consistent character count
       n = Len(hWorkVal)
       If n < 8 Then
           hWorkVal = String(8 - n, "0") & hWorkVal
       End If
       '
       ' Use ChrB to rebuild Bytes.
       For i = 7 To 1 Step -2
           LongToMemoryString = LongToMemoryString & _
              ChrB(CInt("&H" & Mid(hWorkVal, i, 2)))
       Next I
```

```
    Else
        ' Just return zeros.
        ' Use ChrB to build Bytes.
        LongToMemoryString = ChrB(CInt("&H00"))
        LongToMemoryString = LongToMemoryString & _
           ChrB(CInt("&H00"))
        LongToMemoryString = LongToMemoryString & _
           ChrB(CInt("&H00"))
        LongToMemoryString = LongToMemoryString & _
           ChrB(CInt("&H00"))

    End If
End Function
Public Function PROCESS_INFORMATION(hProcess As Long, _
    hThread As Long, _
    dwProcessId As Long, _
    dwThreadId As Long) As String
    '

    ' Convert inbound Long Integers to a memory storage
    ' String format.
    PROCESS_INFORMATION = LongToMemoryString(hProcess) & _
          LongToMemoryString(hThread) & _

          LongToMemoryString(dwProcessId) & _
          LongToMemoryString(dwThreadId)

End Function
Private Sub Command1_Click()
    Dim lRet As Long
    Dim sPROCESS_INFORMATION As String
    Dim hProcess As Long
    Dim hThread As Long
    Dim dwProcessId  As Long
    Dim dwThreadId  As Long
    '
    ' Initialize PROCESS_INFORMATION memory string.
    ' Convert initial Rect values to String to pass
    ' into CreateProcess API.
    sPROCESS_INFORMATION = PROCESS_INFORMATION(0, 0, _
       0, 0)
    '
    ' Call CreateProcess.
```

```
    lRet = CreateProcess(CStr(Text1.Text), CStr(Text2.Text), _
        0, 0, 0, 0, 0, 0, 0, sPROCESS_INFORMATION)
    '
    'convert string back to long integer
    getPROCESS_INFORMATION sPROCESS_INFORMATION, hProcess, _
      hThread, dwProcessId, dwThreadId
    '
    'The handle to the process is returned in the
    'sPROCESS_INFORMATION string when CreateProcess is called.
    'This hProcess value can be passed to TerminateProcess.
    'Uncomment the 3 lines below to terminate the process.
    'MsgBox "Click to terminate process"
    'Dim x As Long
    'x = TerminateProcess(hProcess, 0)

End Sub

Private Sub Form_Load()
    ' Text may require to be changed based on your file locations.
    Text1.Text = "\Program Files\Sybase\ASA\dbmlsync.exe"
    Text2.Text = "-c ""uid=DBA;pwd=SQL;" & _
       "dbf=\Program Files\Sybase\ASA\asademo.db"" -k "

End Sub

Private Sub Form_OKClick()
   App.End
End Sub
```

付録 C：Windows エクスプローラの使用による、デスクトップからデバイスまたはエミュレータへのナビゲーション

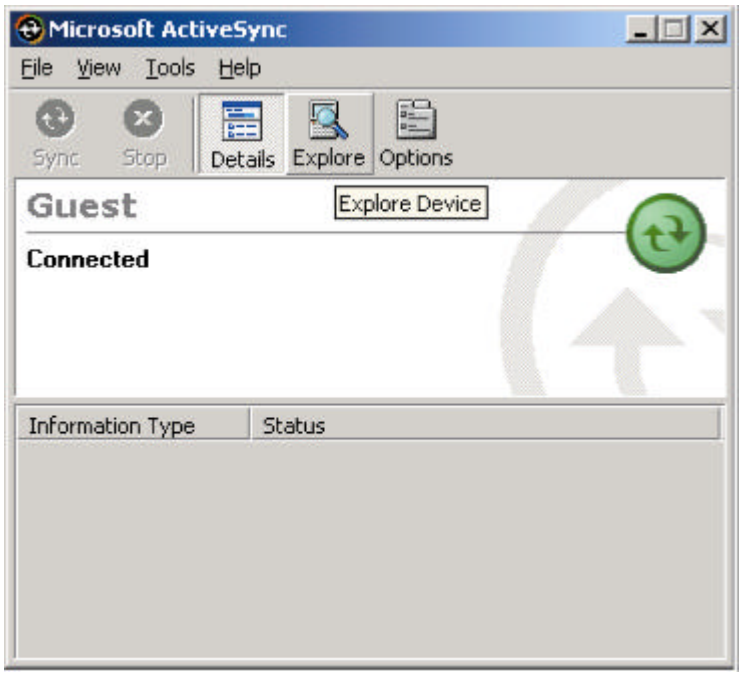
ファイル検索では、多くの人がWindows エクスプローラに慣れています。

デスクトップからデバイスまたはエミュレータへナビゲーションする際には、次の手順を役立ててください。

また、Windows エクスプローラをデバイスで使用するには、次の手順に従います。

1 System Tray の  アイコンをダブルクリックします。

2 次のウィンドウが表示されます。

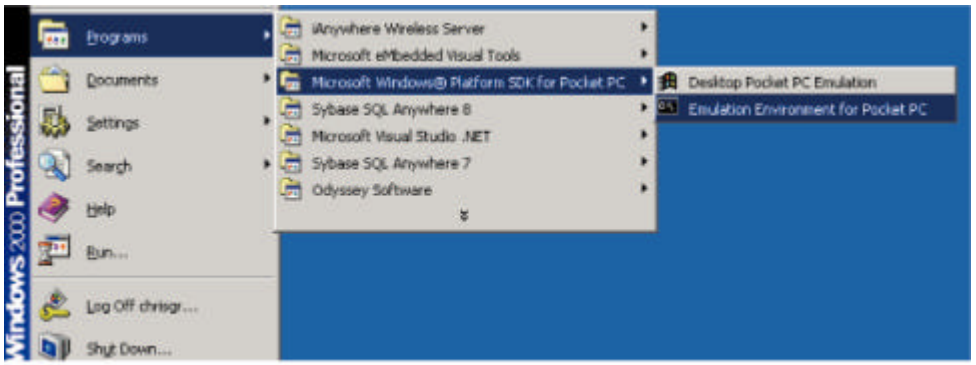


3 このウィンドウで [Explore] ボタンをクリックします。これで、デバイスのディレクトリ間でコピー・アンド・ペーストするときに Windows エクスプローラを使用できます。

Windows エクスプローラをエミュレータで使用するには、次の手順に従います。

1 [スタート] - [プログラム] - [Microsoft Windows Platform SDK For XXX] (XXX は使用しているデバイスのタイプ) を選択します。このプログラム・グループは通常、Microsoft Embedded Tools と共にインストールされます。

2 プログラム・グループから、下図に示すようにエミュレーション環境を選択します。



3 コマンド・プロンプトに、"explorer ." と入力します。これで、エミュレータのディレクトリ間でコピー・アンド・ペーストするときに Windows エクスプローラを使用できます。

法的注意

Adaptive Server、iAnywhere、iAnywhere Solutions、SQL Anywhere、iAnywhere Solutions のロゴは、IAnywhere Solutions, Inc.またはSybase, Inc.とその系列会社の米国または日本における登録商標または商標です。その他の商標はすべて各社に帰属します。

Mobile Linkの技術には、Certicom,Inc.より供給を受けたコンポーネントが含まれています。これらのコンポーネントは特許によって保護されています。

本書に記載された情報、助言、推奨、ソフトウェア、文書、データ、サービス、ロゴ、商標、図版、テキスト、写真、およびその他の資料(これらすべてを"資料"と総称する)は、iAnywhere Solutions, Inc..とその供給元に帰属し、著作権や商標の法律および国際条約によって保護されています。また、これらの資料はいずれも、iAnywhere Solutions, Inc..とその供給元の知的所有権の対象となるものであり、iAnywhere Solutions, Inc..とその供給元がこれらの権利のすべてを保有するものとします。

資料のいかなる部分も、iAnywhere Solutions の知的所有権のライセンスを付与したり、既存のライセンス契約に修正を加えることを認めるものではないものとします。

資料は無保証で提供されるものであり、いかなる保証も行われません。IAnywhere Solutions は、資料に関するすべての陳述と保証を明示的に拒否します。これには、商業性、特定の目的への整合性、非侵害性の黙示的な保証を無制限に含みます。

IAnywhere Solutions は、資料自体の、または資料が依拠していると思われる内容、結果、正確性、適時性、完全性に関して、いかなる理由であろうと保証や陳述を行いません。iAnywhere Solutions は、資料が途切れていないこと、誤りがないこと、いかなる欠陥も修正されていることに関して保証や陳述を行いません。ここでは、「iAnywhere Solutions」とは、米国iAnywhere Solutions, Inc.またはSybase,Inc.とその部門、子会社、継承者、および親会社と、その従業員、パートナー、社長、代理人、および代表者と、さらに資料を提供した第三者の情報元や提供者を表します。

＊本書は、米国iAnywhere Solutions, Inc.が作成・テストしたものを日本語に翻訳したものです。

アイエニウェア・ソリューションズ株式会社
http://www.ianywhere.jp

1019003