SYBASE®

SYBASE® POCKETBUILDER®

THE ENTERPRISE. UNWIRED.

REVIEWER'S GUIDE

TABLE OF CONTENTS

SYBASE
POCKETBUILDER

INTRODUCTION

Thank you for evaluating Sybase PocketBuilder, the new RAD tool for creating mobile and wireless enterprise applications. Based on the acclaimed PowerBuilder® product used by hundreds of thousands of developers worldwide, PocketBuilder has a solid foundation for extending enterprise application development out to Pocket PC devices.

This Reviewer's Guide provides an introduction to PocketBuilder, its features, examples of live PocketBuilder applications, and some key differentiators between PocketBuilder and other mobile application development.

For more in-depth technical information on PocketBuilder, please consult the comprehensive product documentation or visit the Sybase Forums:

Product Information:	**http://www.sybase.com/pocketbuilder**
Sybase Forums:	**http://www.sybase.com/support/newsgroups**
Product Manuals:	**http://www.sybase.com/support/manuals**

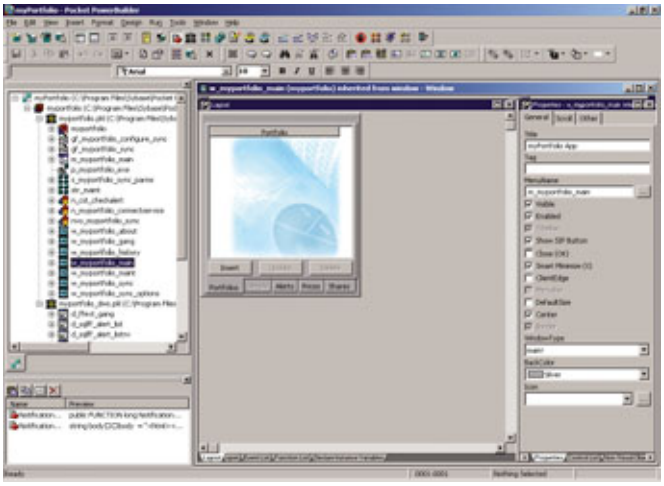
Name Change Note: The original name for the Sybase PocketBuilder product was "Pocket PowerBuilder". Some screenshots used in this document were taken off version v1.5. The name has been changed to PocketBuilder since version 2.0.

Note: A fully functional 60-day evaluation version of PocketBuilder is available for download from the Sybase Web site.

http://www.sybase.com/pocketbuilder

POCKETBUILDER OVERVIEW

Sybase PocketBuilder is a new rapid application development tool that speeds the creation of mobile and wireless enterprise Pocket PC applications. It combines the popular PowerBuilder IDE and patented DataWindow® technology with the market-leading mobile database and enterprise synchronization suite. As an evolution of the proven PowerBuilder product, PocketBuilder provides a reliable platform for wireless commerce, customer, and business applications.



KEY FEATURES

Pocket PC developers can quickly create robust applications with the easy-to-use and intuitive PocketBuilder environment, which:

- Delivers the first highly productive 4GL IDE for mobile development
- Extends the patented Sybase DataWindow to mobile environments, enabling dynamic data access with display formatting and data manipulation capabilities—all without coding
- Tightly integrates with SQL Anywhere® Studio, the market-leading mobile database and enterprise synchronization solution
- Provides PowerBuilder developers with the same best-in-class rapid application development (RAD) platform they rely upon today
- Supports Microsoft INK Controls to add signature verification, handwritten recognition on text fields or images to existing applications
- Supports SMS, Smart Phone, GPS, digital cameras, bar codes, biometric scanners and different new device technologies to really free-up your mobile workers' time
- Supports Printing from a PocketBuilder Application in conjunction with the FieldSoftware Printer CE SDK, available from the FieldSoftware Web site at **http://www.fieldsoftware.com**

PRODUCTIVE 4GL IDE

- Support for the complete development cycle
- Rich component set (detailed below)
- Visual development with minimal coding (PowerScript detailed below)

DATAWINDOW CONTROL

- Point-and-click component for sophisticated data presentation
- Built-in SQL for selects, updates, inserts, and deletes
- Data presentation in many formats
    - Freeform
    - Grid
    - Tabular
    - Graph
    - Group
- Printing from within a PocketBuilder Application

POWERSCRIPT

- Powerful OO language
- Event-driven development
- Roots in Basic/Pascal
- Support for user events, functions, and objects (visual and non-visual)
- Ability to send mail through a Microsoft ActiveSync connection configured to synchronize mail files with a desktop mail client

RICH SET OF COMPONENTS

Examples:

- Command button
- Picture button
- Check box
- Radio button
- Static text
- List view
- Tree view
- List box
- Drop-down list box
- Edit mask
- Single-line edit
- Multi-line edit
- Progress bar
- Scroll bars
- Line, oval, rectangle, and tabs
- Native objects that encapsulate operating system and device specific APIs to further simplify programming.
    - Pocket Outlook Object Model
    - Rich Ink
    - Notification bubble
    - HP 5400 & 5500 series iPAQ Biometric Scanner
    - Symbol Barcode Scanner

SOURCE CODE CONTROL

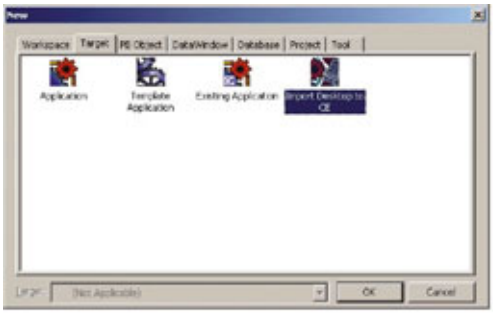
Source code is now available within the PocketBuilder environment through the industry standard SCC API, making team development easier and more manageable.

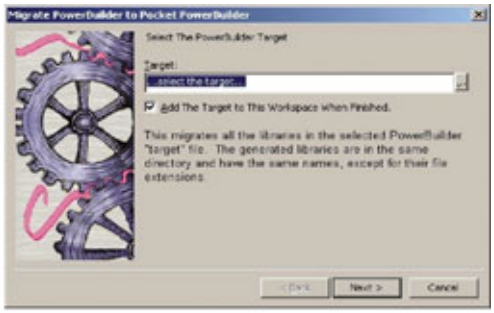
ORCA SCRIPT

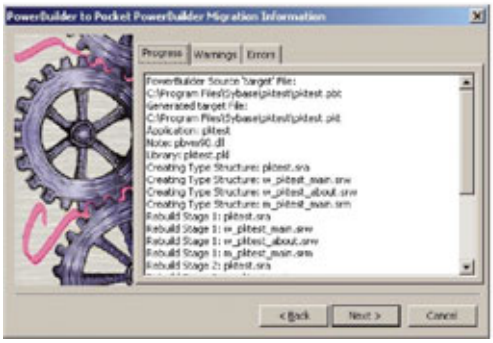
With ORCA Script comes the ability to build your application in a batch processing way. Ideal for overnight builds.

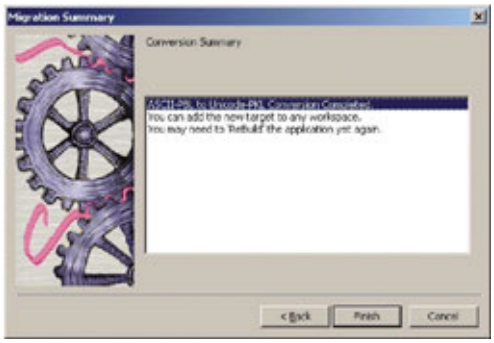
MIGRATION

Existing PowerBuilder developers can easily leverage their expertise to create new applications or extend existing solutions using the PocketBuilder IDE. PocketBuilder aids the developer with a migration wizard for existing PowerBuilder applications.









DEPLOYED APPLICATIONS

Following are details of some of the first deployed PocketBuilder applications, real-world examples of the innovative mobile applications made possible by the product.

TECHWAVE 2003 CONFERENCE SCHEDULER

Company:	Sybase, Inc.
First deployment location:	Sybase TechWave Conference, Orlando, FL
Date of deployment:	August 2003

APPLICATION DESCRIPTION

The TechWave 2003 conference scheduler was available to all attendees who used Pocket PC devices (potential deployment 1500 –2000 attendees). The application allowed attendees to search sessions by track, instructor, product, and other parameters. Users could drill down for track description, date, time, and location. They could then add the session to their personal session agenda, stored locally on the Adaptive Server Anywhere database. The application provided session reminders, detailed maps of the conference facilities, and keynote session information.

TECHNICAL DETAILS

The TechWave 2003 conference scheduler contains an Adaptive Server Anywhere runtime database. The database was initially loaded with all of the conference information and did not require any synchronization.

TechWave 2003 was tested on a Toshiba e370, Dell Axim X5, iPAQ1940 (running Windows Mobile 2003), iPAQ 3950, iPAQ 3970, and iPAQ 5450. All platforms ran Adaptive Server Anywhere and the TechWave 2003 conference scheduler.

The application was available on the conference CD and is still accessible online at **http://www.sybase.com/detail?id=1025516**.

The users that attended TechWave 2003 and installed the application gave very positive feedback with regard to the speed and functionality of the application. One specific point raised was the inability of the application to place session in the Pocket Outlook Diary. This is now possible with PocketBuilder.

AIRMAN MOBILE

Company:	Software Tool & Die
Name of Application:	AIRMAN Mobile (an extension to the AIRMAN application)
First deployment location:	JFK Airport, New York, NY
Date of deployment:	August 12, 2003

APPLICATION DESCRIPTION

The original AIRMAN (Airport Information Report Manager) application was designed to streamline airport operations and establish a safer travel environment. AIRMAN helps manage airport wildlife, perform FAR139 inspections, and ensure personnel are up-to-date on training. Developed in PowerBuilder 9 by Winfield Solutions, AIRMAN is now used at airports throughout the United States and Canada.

AIRMAN Mobile is the mobile data entry extension for AIRMAN users. Designed to improve productivity, it enables users to input data continually and at a fraction of the time required in the original paper–to-PC process. The application was developed by Software Tool & Die.

TECHNICAL DETAILS

AIRMAN Mobile contains a remote Adaptive Server Anywhere (ASA) database that synchronizes with the central AIRMAN ASA database. The application uses the TCP/IP and ActiveSync configuration to connect the main PowerBuilder application (utility) to the AIRMAN Mobile database on the Pocket PC device. It then opens another connection to the central AIRMAN database on the NT network. After both connections are established, PowerBuilder Data Pipelines move new data from the mobile to the central database (using auto increment to assign new ID) and then move new and changed static code table data back from the central to the mobile database.

AIRMAN Mobile has been tested on a Toshiba e370, Dell Axim X5, and IPAQ 3950. All platforms run ASA and AIRMAN smoothly. JFK Airport is initially using Axim X5s.

MARKET POTENTIAL

Attendees of the North American Airport Management conference (held in August 2003) saw a demonstration of AIRMAN, a solution well known by a large segment of the audience. That same month, JFK became the first airport to deploy AIRMAN Mobile. Since then, airport officials in Miami, Toronto, Portland, and other cities, as well as various U.S. Air Force locations, have begun their own evaluations of the mobile solution.

BASIC SAMPLE APPLICATION

This tutorial, will walk you through a basic "Hello World" PocketBuilder application. In this application there will be no database access, just the creation of a sample window. This will help you familiarize yourself with the IDE and some of the wizards.

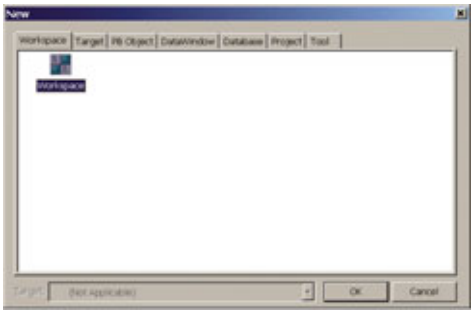
BEFORE YOU BEGIN

Before going through the tutorial, please make sure you have the PocketBuilder Virtual Machine (VM) as well as the Adaptive Server Anywhere (ASA) database and MobiLink Client installed on your deployment device or emulator. Instructions for accomplishing this are provided in the PocketBuilder Installation Guide.

To set up the workspace:

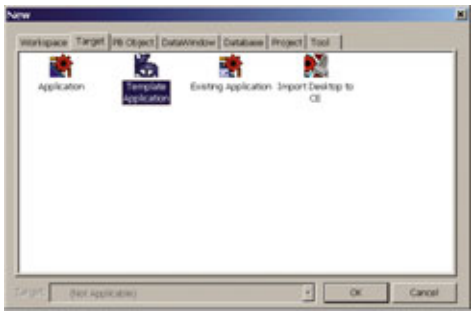
1. Select **File -> New**.

2. In the Workspace tab, select **Workspace** and click **OK**.



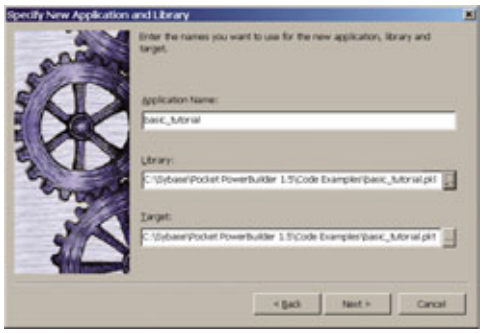
3. Save the new workspace as **basic_tutorial** under the …**Code Examples** directory.

4. Select **File -> New**. **Target Tab** - Template Application.

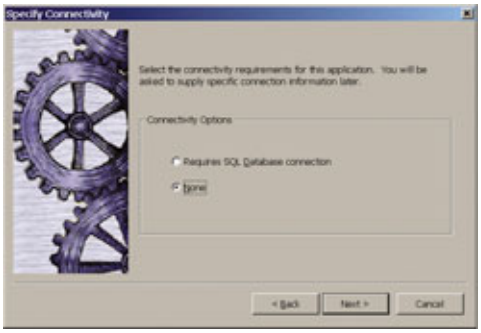


If not specified please take the system generated defaults in the wizard.

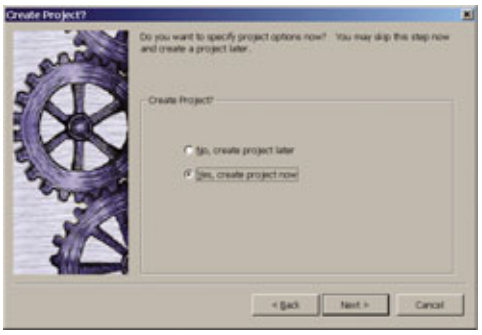
5. Specify the name of the Application as **basic_tutorial**.



6. This tutorial does not require any Database connectivity.

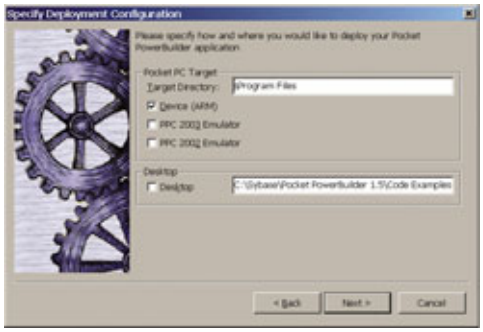


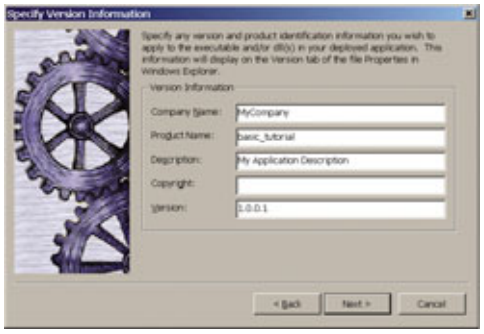
7. Create a Project Object. If you are going to run this sample just from the PocketBuilder IDE, then this step can be ignored (and continue from step #9).



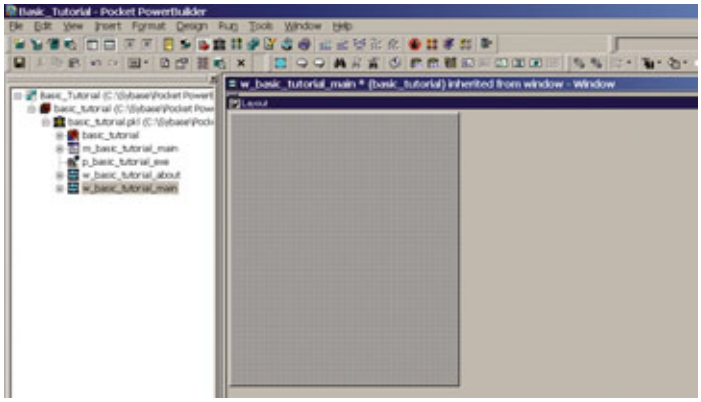
Project name & Executable name can be left as default.

8. This is where you can specify the Deployment target.



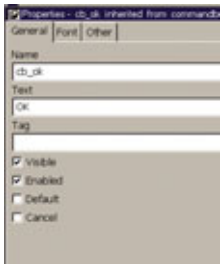


9. At the end of the wizard confirm Finish and PocketBuilder will generate you a template application that we will work on, as seen below.



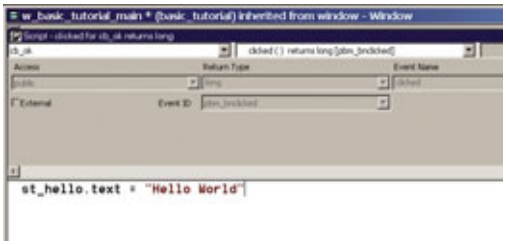
10. In turn select a **Command Button** and a **Static Text**, from the component palette and place them on the form.

11. Click on the **Command Button** and change some of its properties in the properties tab. Change the default name to **cb_ok** and make its initial text property to **OK.**

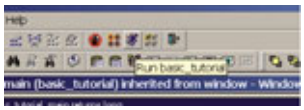


12. Click on the **Static Text Field** and change some of its properties in the properties tab. Change the default name to **st_hello** and make its initial text property to blank.

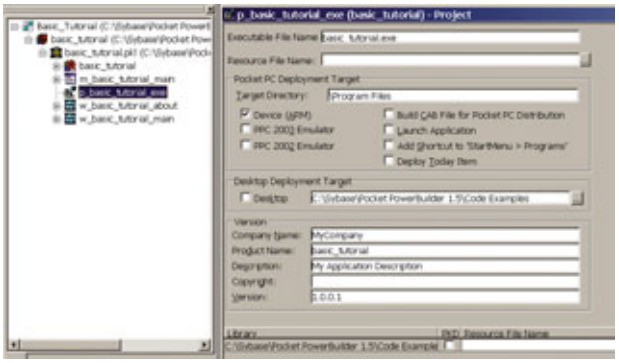
13. Double click on the **Control Button** to code the clicked event. Within this event we will make the change to the static text fields, text property.



14. Run the application from the IDE by clicking on the **running man icon**.



This application could also be deployed to a device or emulator by running the project object.



BASIC DATAWINDOW APPLICATION

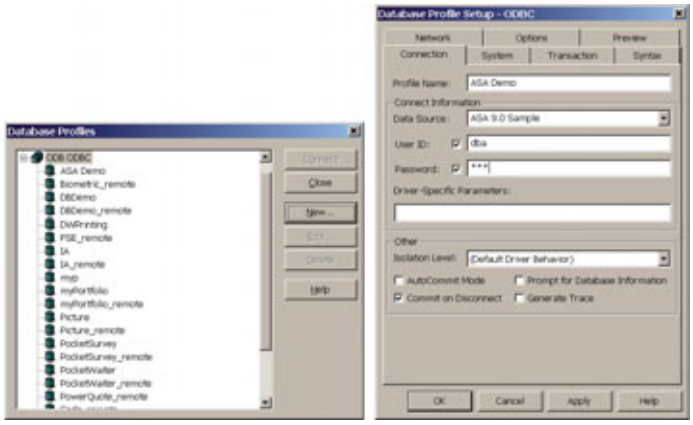
This tutorial will walk you through a basic PocketBuilder application that uses a DataWindow to select data from the ASA demo database. This will facilitate your usage of the DataWindow Painter within the IDE.

BEFORE YOU BEGIN

Before going through the tutorial, please make sure you have the PocketBuilder Virtual Machine (VM) as well as the Adaptive Server Anywhere (ASA) database and MobiLink Client installed on your deployment device or emulator. Instructions for accomplishing this are provided in the *PocketBuilder Installation Guide*.

Also make sure you have the ASA Sample Database in the **DB profile.**

If it does not appear click on the **new** button and set up as shown.

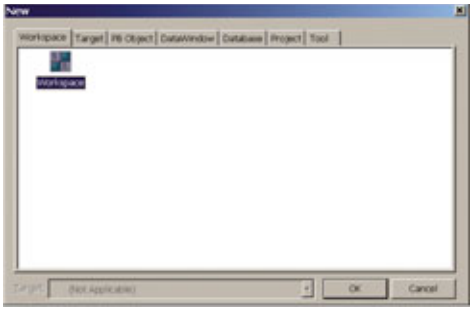


The password for the ASA database is 'sql'.

To set up the workspace:

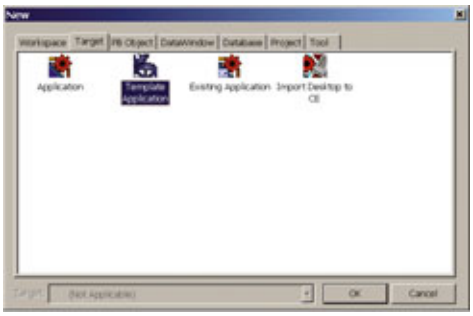
1. Select **File -> New**.

2. In the **Workspace** tab, select **Workspace** and click **OK**.



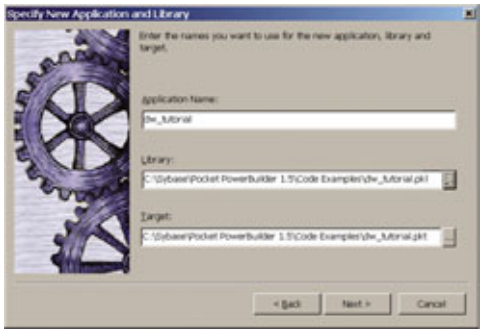
3. Save the new workspace as **dw_tutorial** under the …**Code Examples** directory.

4. Select **File -> New**. **Target Tab - Template Application.**

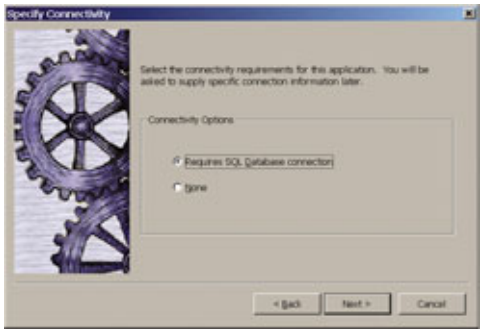


If not specified please take the system generated defaults in the wizard.

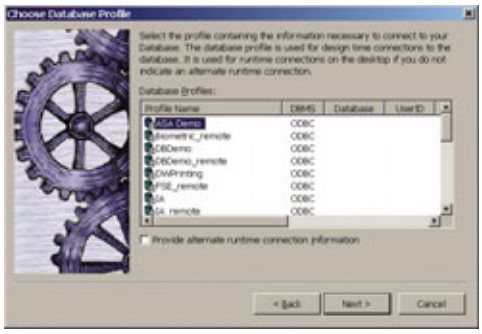
5. Specify the name of the Application as **dw_tutorial**.



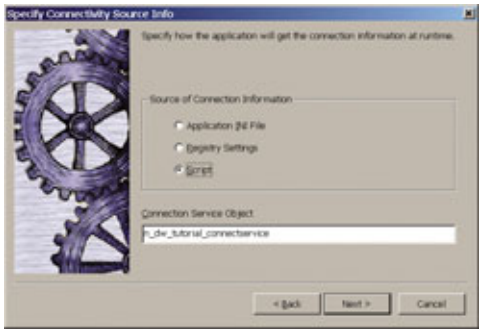
6. This tutorial does require Database connectivity.



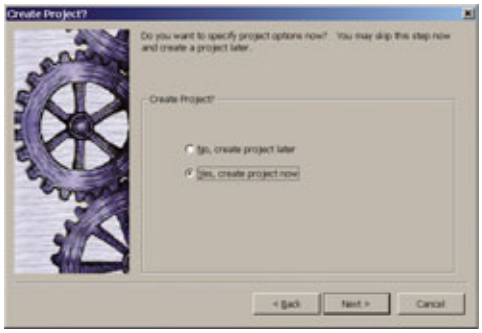
7. Select the ASA Demo Database profile.



Specify that the application will get the connection information via script.

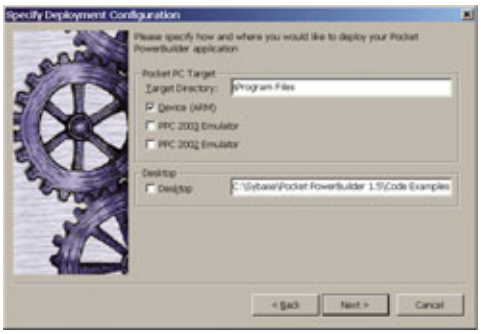


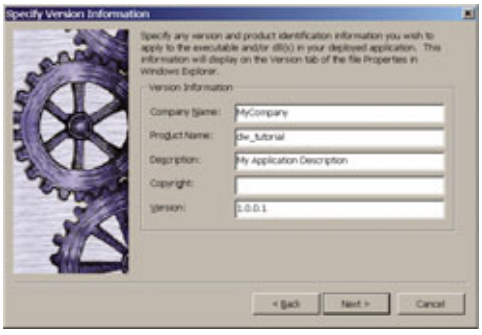
8. Create a Project Object. If you are going to run this sample just from the PocketBuilder IDE, then this step can be ignored (and continue from step #9).



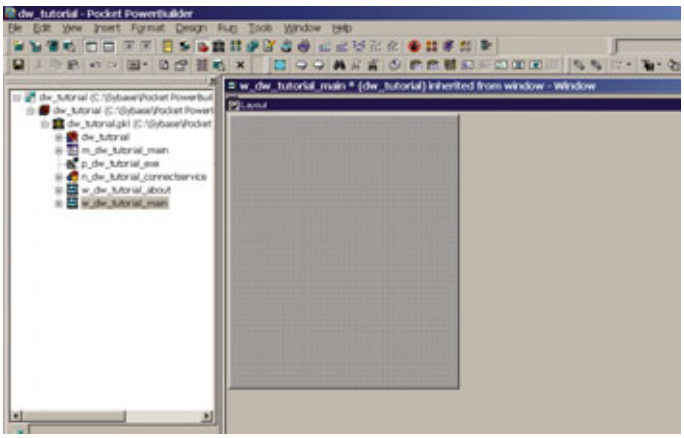
Project name & Executable name can be left as default.

9. This is where you can specify the Deployment target.

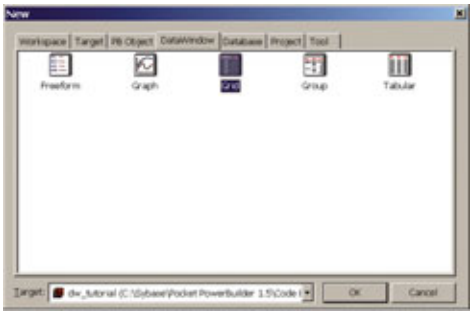




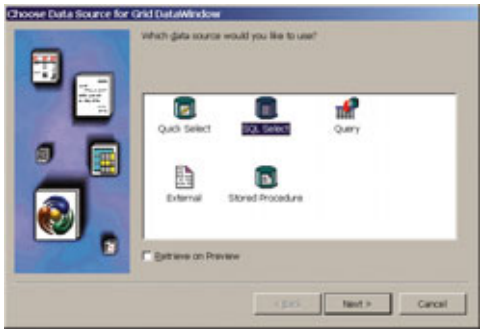
10. At the end of the wizard confirm Finish and PocketBuilder will generate you a template application that we will work on, as seen below:



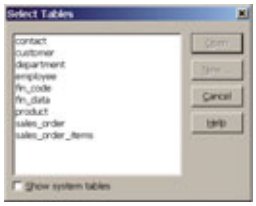
11. Select **File -> New**. **DataWindow** – This will open the DataWindow Painter.



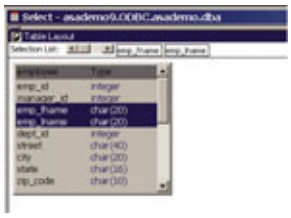
12. Select the **DataSource** as SQL Select.



13. The DataWindow Painter will display the tables in the ASA Demo Database.



14. Select the **employee table** and click on **Open**. The DataWindow Painter displays the SQL Painter. Select the columns **emp_fname** and **emp_lname.**



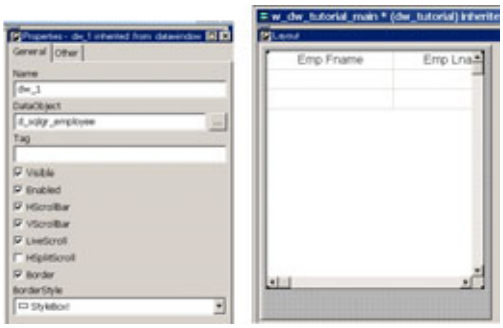
Now click on the **return** button.

15. Allow all the defaults and the DataWindow will be displayed in the DataWindow Painter.

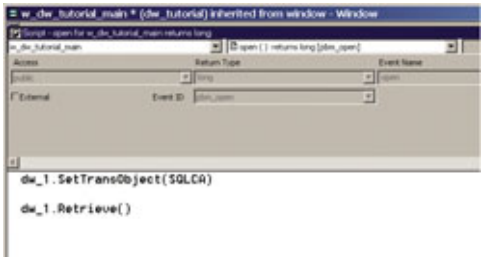
16. Save the DataWindow as **d_sqlgr_employee.**

17. Place a DataWindow Control on the main window.

18. Associate the DataWindow Control with the DataWindow Object you created. This is achieved via the **property sheet** for the DataWindow object dw_1.



19. Double click the window to code in the Open event.



We code the SetTransObject for the DataWindow **dw_1**, this associates the transactionobject SQLCA that is connected to the database, with the datawindow.

Next we code the Retrieve function for **dw_1**. This makes the datawindow object associated with **dw_1**. to retrieve the data from the Database.

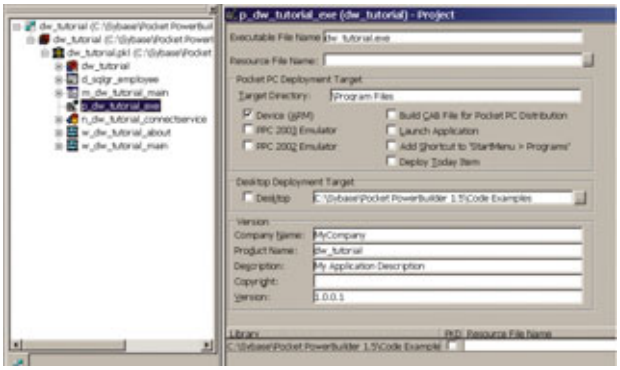
20. The application can be run in the IDE.

When deploying the application to a device or emulator we also need a .DSN file for the application to pick up certain values so that it can connect to the DB.

```
[ODBC]
uid=dba
pwd=sql
enginename=ASADemo
databasename=ASADemo
databasefile=\Program Files\Sybase\ASA\asademo.db
start=\Program Files\Sybase\ASA\dbsrv9.exe -q
driver=dbodbc9.dll
```

The .DSN file should be placed in the root folder of the Pocket PC Device.

This application could also be deployed to a device or emulator by running the project object.



SALESDB SAMPLE APPLICATION

This tutorial is included on the product CD. It will walk you through all the steps in creating and deploying a PocketBuilder application. A skeleton application is provided. You will add more functions to complete the application during this tutorial.

Note: You can download an evaluation edition of PocketBuilder, including this tutorial, at ***http://www.sybase.com/pocketbuilder***

BEFORE YOU BEGIN

Before going through the tutorial, please make sure you have the PocketBuilder Virtual Machine (VM) as well as the Adaptive Server Anywhere (ASA) database and MobiLink Client installed on your deployment device or emulator. Instructions for accomplishing this are provided in the PocketBuilder Installation Guide.

OVERVIEW

SalesDB is a simple sales-status application. It uses MobiLink synchronization technology to synchronize data between the Pocket PC and server databases. Both the remote and consolidated databases are either ASA 8 or ASA 9. To test the application on the Pocket PC, you must have either an ARM/XScale device or a Pocket PC emulator. The PocketBuilder Installation Guide contains more information on installing the Pocket PC emulator. The instructions for both Pocket PC devices and emulators are the same unless stated otherwise.

The process of creating the SalesDB tutorial application includes the following steps:

1. Create the Adaptive Server Anywhere remote and consolidated databases.
2. Add features to the tutorial application in the PocketBuilder IDE.
3. Preview the application in the PocketBuilder IDE.
4. Deploy the application to a Pocket PC device or emulator.

The goal of this tutorial is to provide a basic introduction to PocketBuilder and MobiLink synchronization. It should take about 60 minutes to complete.

Some of the common questions are listed in the Troubleshooting section at the end of the tutorial.

PART 1: SETTING UP THE ASA DATABASES

The SalesDB application uses one consolidated database on the PC to store all data. Different remote databases are placed on the Pocket PC devices and their data are transferred to the consolidated database via MobiLink synchronization.

To set up the databases:

1. Run **MakeDB.cmd** located under the …**Code Examples\SalesDB\db directory**. This will create the remote and consolidated databases. The corresponding Data Source Name (DSN) entries will also be created.

Note: *MakeDB.cmd will automatically detect the version of ASA running on your machine and use the most recent version installed. (It will use ASA 9 if both ASA 8 and ASA 9 are installed.) You can explicitly decide to use ASA 8 by using the MakeDB8.cmd file to create the database.*

2. Scan the output of the script to make sure no error has occurred. Then press any key to exit.

To verify the databases are created correctly:

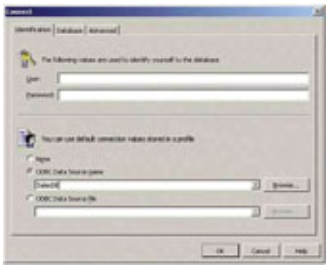
1. From the **Start Menu**, select **Programs -> Sybase SQL Anywhere 8 -> Sybase Central**.

Note: *If you are using Adaptive Server Anywhere 9, SQL Anywhere Studio is located at Programs -> SQL Anywhere 9 in the start menu.*

2. Select **Tools -> Connect….**

3. Select **Adaptive Server Anywhere 8/9**.

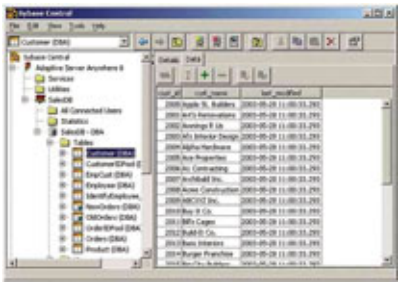
4. Select the **ODBC Data Source name** radio button, click **Browse…** and select **SalesDB**.



5. Select the **Advanced tab** and select the **JDBC-ODBC bridge** (in ASA 8) or the **iAnywhere JDBC driver** (in ASA 9) radio button.

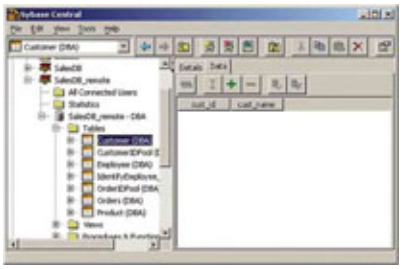
6. Click **OK**.

7. Verify that the SalesDB tables are created and populated with sample data.



8. Repeat steps 2-7 to create the SalesDB_remote remote database.

Note: *This database will not be populated with data. Data will be transferred from the consolidated database during the first synchronization.*

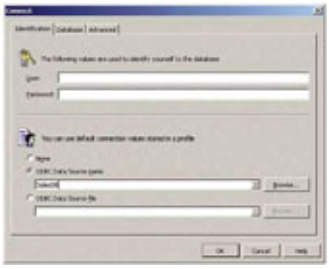


The consolidated database contains the synchronization scripts used by MobiLink. To verify these scripts are created properly:

1. From Sybase Central, select **Tools -> Connect….**

2. Select **MobiLink Synchronization**.

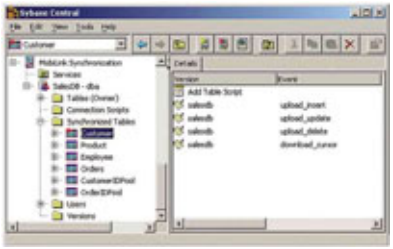
3. Check **ODBC Data Source** name and select **SalesDB**.



4. Select the **Advanced tab** and select the **JDBC-ODBC bridge** (in ASA 8) or the **iAnywhere JDBC driver** (in ASA 9) radio button.

5. Click **OK**.

6. Expand the Synchronized Tables node and select **Customer**.



7. Double-click the **salesdb** under Version, next to the download_cursor event. The script should read:

```
SELECT cust_id, cust_name FROM Customer WHERE last_modified > ?
```

This process downloads data that has changed since the last synchronization from the consolidated database to the remote database.

8. The upload stream uses three events: upload_insert, upload_update, and upload_delete. Double-click on the **salesdb** under Version, next to upload_insert. The script should read:

```
INSERT INTO Customer( cust_id, cust_name ) VALUES( ?, ? )
```

This process inserts any new customers created in the remote database into the consolidated database.

9. Repeat the previous step with the upload_update and upload_delete events and view the related SQL synchronization scripts.

Note: *To keep the tutorial focused on the application development capabilities of PocketBuilder, we will not explore further how these scripts are made. For more information about synchronization scripts, please refer to the MobiLink documentation.*

10. Close Sybase Central.

PART 2: MODIFYING THE SALESDB APPLICATION

CREATING THE DATABASE PROFILE

Before we can start developing with the PocketBuilder IDE, we must tell it how to connect to the database. As SalesDB is a Pocket PC application, only the remote database needs to be setup inside PocketBuilder.

To create a database profile:

1. From the **Start Menu**, select **Programs -> Sybase -> PocketBuilder >Pocket Builder** to start PocketBuilder.
2. Select **Tools -> Database Profile…**
3. Select **ODB ODBC** and click **New**.
4. Enter **SalesDB_remot**e as the Profile Name.
5. Select **SalesDB_remote** as the Data Source.

6. Uncheck User ID and Password. (They are provided by the DSN file on the Pocket PC device.)
7. Click **OK** to accept all other defaults.
8. Expand ODB ODBC.
9. Select **SalesDB_remote** and click **Connect**.

The ASA remote database will start and a connection will be established.

SETTING UP THE SALESDB WORKSPACE

The SalesDB tutorial comes with a **salesdb_tutorial.pkl file**. This contains the skeleton code which you will build upon.

To set up the SalesDB workspace:

1. Select **File -> New**.

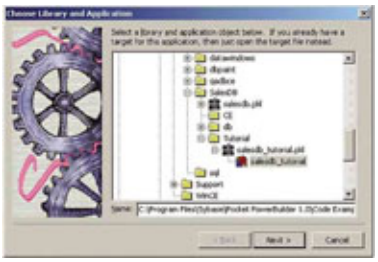
2. In the **Workspace tab**, select **Workspace** and click **OK**.

3. Save the new workspace as **SalesDB_tutorial** under the …**Code Examples\SalesDB\tutorial** directory.

4. Select **File -> New**.

5. In the **Target tab**, select **Existing Application** and click **OK**.

6. Locate and expand **salesdb_tutorial.pkl**.



7. Select **salesdb_tutorial** application and click **Next**.

8. Click **Next** again then click **Finish** to accept the default locations.

The SalesDB tutorial skeleton application is now ready to be modified.

CREATING THE F_CONN FUNCTION

Before the application can access the database, it must connect to it. We use a function call f_conn to handle the connection code. To create the f_conn function:

1. Select **File -> New**.

2. In the **PB Object tab**, select **Function** and click **OK**.

3. Enter the following data:

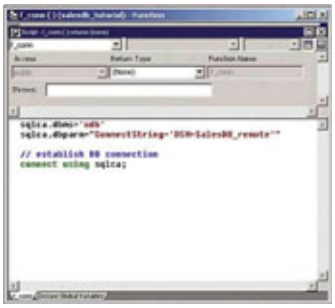
∑ ▪ Return Type: (None)
∑ ▪ Function Name: f_conn

In the script view (the code editor below the function properties), enter the following code:

```
sqlca.dbms='odb'
sqlca.dbparm="ConnectString='DSN=SalesDB_remote'"

// establish DB connection
connect using sqlca;
```

This code provides the connection parameter and establishes a connection to the ASA database. The same parameter can be used to initiate a connection on the PC or the Pocket PC environment.



4. Select **File -> Save** to save the function.

5. Accept the default name.

6. Expand the System Tree in the **salesdb_tutorial.pkl file**. You should see the new f_conn f function you created.

The f_conn function is used when the application first starts. It is called from the ue_postopen event inside the salesdb application. The database connection is established in this event and any connection error is reported to the user. When the application terminates, the database connection is closed by the f_disconn function provided for you.

CREATING A DATAWINDOW

A DataWindow (DW) is a powerful PocketBuilder component that allows you to manipulate data visually in a variety of ways. The DataWindow we build will display sales information in a scrollable window.

To build the d_orders DataWindow object:

1. Select **File -> New**.

2. In the **DataWindow tab** select **Freeform**.

3. Select **SQL Select** and check **Retrieve on Preview**.

4. Click **Next**.

5. In the Select Tables window, click on customer, orders, and product.

6. Click **Open**.

7. In the **Table Layout window**, click on (in this order) **order_id from orders**, **cust_name from customer**, **prod_name from product**, **quant from orders, price from product**, and **disc**, **status**, and **notes from orders**.

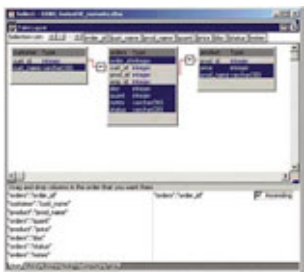
The sequence of your selections determines the order the column appears in the DataWindow. It is also possible to rearrange the order later.

The Syntax tab at the bottom shows the query that PocketBuilder will use to retrieve the data. We also want to sort the result according to order_id. To sort the result:

1. Select the **Sort tab**.

2. Drag “orders”.”order.id” from the left to the right.

3. Make sure **Ascending** is checked.



The Syntax tab confirms the columns we want to select as well as the sorting criteria. Notice that all of the SQL was generated for you automatically.

This completes the data selection step. Next, we will visually manipulate the physical location of the data displayed in the DataWindow object.

1. Select **File -> Return to DataWindow Painter**.

2. Click **Next** to accept the default for the Color and Border Settings.

3. Click **Finish** to generate the DataWindow.

4. Select **File -> Save** to save the DataWindow.

5. Enter **d_orders** as the name and click **OK**.

6. Since the order_id is for internal use and not shown to the user, in the Header section, select **Order Id:** and **orders_order_id** and press the **Delete** key.

7. Select the "Cust Name:" label and change the Text property to "Customer:"

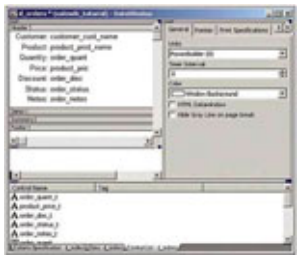
8. Repeat the previous step and rename these labels.

   Prod Name: to Product:

   Quant: to Quantity:

   Disc: to Discount:

9. Drag the labels to match the screen shown below. You can select multiple labels and move them at once by holding the **Ctrl key** and selecting each label.



10. You can also align multiple labels via **Format -> Align**.

11. Save the DataWindow again by sekecting **File -> Save**.

MODIFYING THE MENU

Next, we will add new items to the menu.

1. Double-click **m_salesdb** in the **System Tree** on the left hand side of the IDE.

2. In the tree menu view, right click **Order** and select **Insert Menu Item**.

3. Enter "File" as the menu's text.

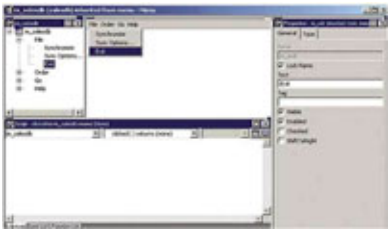
4. Right click on **File** and select **Insert Submenu Item**.

5. Enter "Synchronize" as the menu's text.

6. Repeat steps 4-5 and create submenu items with "Sync Options…" and "Exit" as the text on the menus.

7. Make sure the Name properties are m_file, m_synchronize, m_syncoptions, and m_exit, respectively. To modify the menu name, uncheck **Lock Name** and modify.

8. The menu painter should look like the following figure:



9. Double click on **Exit**.

10. Make sure the drop-down menu on top of the script view displays the clicked() event for the m_file.m_exit object.

11. Enter the following code in the script view.

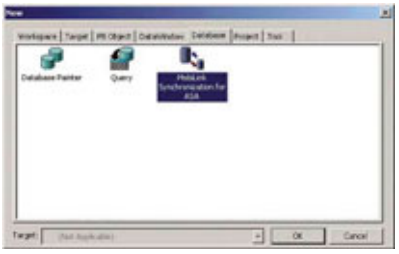
```
// Terminate application
f_disconn()
Halt Close
```

12. Select **File -> Save** to save the changes.

We will now generate a MobiLink connection for the remote application using the MobiLink Synchronization Wizard.

1. Select **File -> New…**

2. In the **Database tab**, select **MobiLink Synchronization for ASA** and click **OK**.



3. Read the first two introduction screens to learn about what the wizard will provide. Click **Next** on each once you have read them.

4. Click **Next** to accept the default of **salesdb_tutorial.pkl** as the package to store the generated MobiLink objects.

5. Select **SalesDB_remote** from the drop-down menu, and click **Test Connection** to ensure the DSN is in working order.

6. A 'Connection successful' message should appear in the lower-left corner of the dialog, If it does not, verify the DSN is correctly configured using the ODBC Administrator. Click **Next**.

7. Choose **Browse**, then locate and open **SalesDB_remote.DSN** (typically found in the …Code Examples\SalesDB directory).

8. Click **Next**.

9. The follwing screen shows the publications present in the ASA remote database. Highlight the **salesapi publication,** and click **Next**.

10. Click **Next** to accept the default names for the generated MobiLink objects.

11. You can now decide how you would like the MobiLink synchronization status messages to be displayed. Click **Next** for the wizard to create a synchronization status window for your application.

12. The Runtime Configuration Objects dialog gives you the ability to allow users to change synchronization settings at runtime. Typically, you would disable this option, or deploy a minimal subset of the generated windows. For now, let's demonstrate the full functionality of the windows—ensure 'Prompt user for password and runtime changes' is checked, and click **Next** to generate the synchronization options objects.

13. Check **Show all except –vc and –vp (-v+)**. This dialog determines the amount and type of information shown in the MobiLink status window during synchronization. Click **Next**.

Note: *This option will show a lot of status messages which will affect the performance of the synchronization. In a deployed application it may be better to show less information to improve performance.*

14. The next page in the wizard allows you to specify any additional MobiLink command line or extended options, including the MobiLink host and port. Here is where you could enter default values for the MobiLink server location. Click **Next** to leave them empty.

15. The final page in the wizard shows the complete list of options you selected. You can go back and change any that may be incorrect.

16. Click **Finish** to generate the MobiLink synchronization components in your project.

The MobiLink Synchronization Wizard has created the following synchronization objects:

- **nvo_salesdb_tutorial_sync** – a non-visual user object that controls the MobiLink synchronization client
- **gf_salesdb_tutorial_sync** – a global function that creates the user object and initiates synchronization requests
- **s_salesdb_tutorial_sync_parms** – a structure that stores the MobiLink command line parameters (all of the variables accessible from the wizard are MobiLink command line options)
- **gf_salesdb_tutorial_configure_sync** – a global function that handles a user request to change the synchronization options, then stores the values into the Windows/Windows CE registry

Note: *If you are using Adaptive Server Anywhere 9, you will need to change the ASA_REGPATH constant in the nvo_salesdb_tutorial_sync object. You can change this value by going into the script editor for the nvo_salesdb_tutorial_sync object and selecting the 'Declare Instance Variables' tab. The value generated in the wizard is:*

```
constant string ASA_REGPATH =
"HKEY_CURRENT_USER\Software\Sybase\Adaptive Server Anywhere\8.0"
```

For ASA 9 clients, this value needs to be changed to:

```
constant string ASA_REGPATH =
"HKEY_CURRENT_USER\Software\Sybase\Adaptive Server Anywhere\9.0"
```

The MobiLink Synchronization Wizard also creates two windows:

- **w_salesdb_tutorial_sync** – displays the synchronization status information
- **w_salesdb_tutorial_sync_options** – sets synchronization options at runtime

Now, let's enable the **Sync Options…** window from the menu. Keep in mind that in a production environment, giving the end user a high degree of control over the synchronization parameters (as we've done in this tutorial) is usually inadvisable.

1. Double click **m_salesdb** in the **System Tree**.

2. In the tree menu view, expand **File**, and double click on **Sync Options…**

3. Make sure the drop-down menu on top of the script view displays the clicked() event for the m_file.m_syncoptions object.

4. Enter the following code in script view:

```
// Open the Sync Options window
gf_salesdb_tutorial_configure_sync()

// Fetch data
f_refresh_orders(-1)
```

Finally, to enable synchronization, we must add a call to **gf_salesdb_tutorial_sync** when the user clicks **Synchronize** from the **File** menu.

5. In the tree menu view, double click on **Synchronize**.

6. Make sure the drop-down menu on top of the script view displays the clicked() event for the m_file.m_synchronize object.

7. Enter the following code in script view:

```
// Start synchronization
if gf_salesdb_tutorial_sync(string(::g_emp_id), "") <> 0 then
    MessageBox("Error", " MobiLink Synchronization Error. ");
End if

// Fetch data
f_refresh_orders(-1)
```

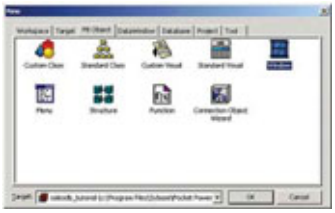
8. Select **File -> Save** to save the changes.

CREATING A WINDOW

The last part of this tutorial creates the w_orders window.

1. Select **File -> New**.

2. In the **PB Object tab**, select **Window** and click **OK**.



3. Change the window properties as follows:

- Title: SalesDB tutorial
- MenuName: m_salesdb
- DefaultSize: checked (check to make sure)

4. Select **File -> Save** to save the window.

5. Enter **w_orders** as the name of the window. Click **OK**.

6. Select **Insert -> Control -> DataWindow** and click inside the **Pocket PC** area on the **Layout tab** to insert a DataWindow control onto the **w_orders** window.

7. Change the DataWindow control properties as follows:

- Name: dw_orders
- DataObject: d_orders
- HScrollBar: checked
- VScrollBar: checked

You should now see the static labels inside the DataWindow control.

8. Select **Insert -> Control -> CommandButton** and click on the window below the DataWindow control to add a button to the window.

9. Name it **cb_prev** and give it the text "< Previous".

10. Repeat the previous two steps with the following settings:

- cb_next Next >
- cb_approve Approve…
- cb_deny Deny…

Rearrange the position of the DataWindow control and buttons so the resulting window looks like this:

Next, we will associate events with the buttons.

1. Double click the **cb_prev button**.

2. Make sure the top drop-down menu shows the clicked() event.

3. Enter the following code in the script view.

```
f_scroll(-1)
```

This function scrolls the data in the DataWindow backwards by one row.

4. Repeat steps 1-3 with the following code segments in the corresponding buttons.

- cb_next: f_scroll(1)
- cb_approve: f_approve_deny(APPROVE)
- cb_deny: f_approve_deny(DENY)

5. Select **File -> Save** to save the changes.

Now we must tell the application to open this window when it starts.

1. In the **System Tree**, double-click the **salesdb_tutorial** application entry.

2. In the top drop-down for the script view, select the **open event**.

3. Uncomment the section following "Uncomment the following section after creating w_orders".

4. In the top drop-down, select the **ue_postopen event**.

5. Uncomment the section following "Uncomment the following section after creating w_orders".

6. Select **File -> Save** to save the changes.

7. In the **System Tree**, double click the **m_salesdb** menu entry.

8. Double click the **Order -> Delete** submenu item.

9. Uncomment the section following "Uncomment the following section after creating w_orders".

10. Select **File -> Save** to save the changes.

11. In the **System Tree**, double-click the **f_scroll** function.

12. Uncomment the section following "Uncomment the following section after creating w_orders".

13. Select **File -> Save** to save the changes.

14. Repeat steps 11-13 for **f_scroll_last**, **f_approve_deny**, **f_refresh_orders** and **f_set_dir_btn_enabled**.

15. Take this opportunity to examine the code in the provided functions. You will find many embedded SQL calls and MobiLink related operations.

The SalesDB application is ready to be tested.

PART 3: TESTING SALESDB IN THE IDE

Before you run the application, the MobiLink server has to be started. To do this, execute the **StartML.cmd** file in the …**Code Examples\SalesDB\db** directory. This should start the MobiLink server and the SalesDB consolidated database on your desktop.

Note: *StartML.cmd will automatically detect the version of ASA running on your machine and use the most recent version installed. (That is, it will use the MobiLink Server from version 9 if both ASA 8 and ASA 9 are installed.) You can explicitly decide to use the MobiLink Server from version 8 by using the StartML8.cmd file to start the MobiLink Server.*

The application can be previewed inside the IDE.

1. Close all the open painters in the IDE.

2. Select **Run -> Run.**

3. The SalesDB application will connect to the local copy of the SalesDB_remote remote database.

4. The application will open the Sync Options dialog. Here you may enter synchronization information, including MobiLink user and password settings. For now, enter "50" in the MLUser text field, and leave all other options as default. When you hit OK, the application will automatically synchronize the data. This is the default behavior of this window, but it can be changed for your own applications.

5. If you use MLUser 50 now, you will want to use another MLUser (51 or 52), when you run the application on your device or emulator.

6. Take this opportunity to explore the layout of the application if you wish. With the ability to run PocketBuilder applications in the IDE, it is possible to develop and test applications with the absence of a device or emulator.

7. Select **File -> Exit** to exit the application.

PART 4: DEPLOYING SALESDB TO A DEVICE

CREATING A PROJECT

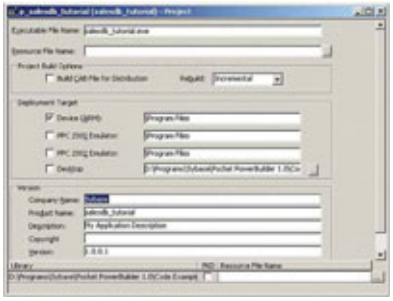
Before an application can be deployed, a project is needed to specify how the application is deployed. To create a project:

1. Select **File -> New**.

2. In the **Project tab**, select **Application** and click **OK**.

3. Enter **SalesDB_tutorial.exe** as the Executable File Name. This is the default.

4. Select **Device (ARM) in Deployment Target** if you have an ARM device or select one of the emulators if you plan to use the emulator. The default location for deployment on the device is the **\Program Files** directory. For your own applications, you may want to deploy into an application-specific directory that you create. For the purpose of this tutorial, leave the default deployment location.



5. Select **File -> Save** to save the project.

6. Enter **p_salesdb_tutorial** as the project name and click **OK**.

7. Select **Run -> Deploy Workspace** to deploy SalesDB to the device or emulator.

Before SalesDB can be run on the device or emulator, the remote database must be set up. There are different steps for devices and emulators.

SETTING UP SALESDB_REMOTE

In each deployment, we copy both the .db and .log file to the device/emulator. This is because synchronization subscriptions were added to the remote database during the initialization phase, for simplicity. We recommend you add synchronization subscriptions either at deployment or during the first execution of the application. When adding the subscriptions at deploy/run time, you only need to deploy the .db file (and not the .log file) to the device.

Note: *If you are using Adaptive Server Anywhere 9, you will have to modify the SalesDB_remote.DSN file. Using a text editor, change the "start=…" line in the file to point to dbsrv9.exe rather than dbsrv8.exe. Also add a line that reads: driver=dbodbc9.dll to the SalesDB_remote.DSN file.*

To set up the remote database on a physical device:

1. On your desktop machine, open **SalesDB_remote.DSN** located in the **Code Examples\SalesDB** directory with a text editor and make sure database file and start have the right paths for the Pocket PC device. They are set to the default location.
2. Open **ActiveSync** while a connection to the device is established.
3. Click **Explore**.
4. Double click "My Pocket PC".
5. Copy **SalesDB_remote.DSN** from your desktop to "My Pocket PC".
6. Go to the **Program Files -> Sybase -> ASA directory**.
7. Disconnect from the SalesDB_remote database from Sybase Central and PocketBuilder. If the database is still running (it will show up in the System Tray), stop the SalesDB_remote ASA server on the PC.
8. Copy **SalesDB_remote.db** and **SalesDB_remote.log** located in the **SalesDB\db\fresh** directory on your desktop to **\Program Files\Sybase\ASA** on the device.

To set up the remote database on the PPC 2002 emulator:

1. On your desktop machine, open **SalesDB_remote.DSN** located in the …**Code Examples\SalesDB** directory with a text editor and make sure database file and start have the right paths for the Pocket PC emulator. They are set to the default location.
2. Run **CEFileVw.exe**. By default it is located under **c:\Program Files\Windows CE Tools\Common\Platman\bin\cefilevw.exe**.
3. Copy **SalesDB_remote.DSN** from your desktop to "\".
4. Stop the SalesDB_remote ASA server on the PC if it is running (it will show up in the System Tray).
5. Copy **SalesDB_remote.db** and **SalesDB_remote.log** located in the SalesDB\db\fresh directory on your desktop to **\Program Files\Sybase\ASA**.

To set up the remote database on the PPC 2000 emulator:

1. On your desktop machine, open **SalesDB_remote.DSN** located in the **Code Examples\SalesDB** directory with a text editor and make sure database file and start have the right paths. The correct paths will likely be similar to "\Program Files\ASA\…"
2. Copy **SalesDB_remote.DSN** to the root directory of the emulator. The default location is **C:\Program Files\Windows CE Tools\wce300\MS Pocket PC\emulation\palm300**.

3. Copy **SalesDB_remote.db** and **SalesDB_remote.log** located in the SalesDB\db\fresh directory on your desktop to **\Program Files\Sybase\ASA** of the emulator. The default location of this is **C:\Program Files\Windows CE Tools\wce300\MS Pocket PC\emulation\palm300\Program Files\ASA**.

Note: *The SalesDB_remote.DSN file sets the DSN on the device and contains the path to the remote database. Note that the path to ASA is different between an actual device and the Pocket PC 2000 emulator. Emulator users will have to edit the DSN file manually as discussed above (remove \sybase in the paths). With this setup, the SalesDB_remote database will start automatically on the Pocket PC when a connection is initiated, and terminate automatically when all connections are closed.*

STARTING MOBILINK SYNCHRONIZATION SERVER

In order for synchronization to occur, the MobiLink synchronization server must be started.

1. Run **StartML.cmd** located in the …**Code Examples\SalesDB\db directory**.

The SalesDB consolidated database and the MobiLink synchronization server will start. If you already started the MobiLink Server in an earlier step, you can skip it now.

Note: *StartML.cmd will automatically detect the version of ASA running on your machine and use the most recent version installed. (That is, it will use the MobiLink Server from version 9 if both ASA 8 and ASA 9 are installed.) You can explicitly decide to use the MobiLink Server from version 8 by using the StartML8.cmd file to start the MobiLink Server.*

RUNNING THE SALESDB SAMPLE APPLICATION

1. Tap on the **Start Menu** on the Pocket PC and select **PocketBuilder**

2. Tap on **SalesDB_tutorial.exe** to start the SalesDB application.

3. When SalesDB is first run, the Sync Options window will appear, allowing you to enter MobiLink user and password values. For the tutorial, enter "51" into the MLUser text field, leave MLPassword blank, and leave all other options as default.

Note: *The databases we have generated using the makeDB.cmd batch file are intended for single-user devices only. You will notice that the remote database rows will appear to overlap when synchronizing multiple users' records. Adaptive Server Anywhere does have the capability to handle this issue, but it is beyond the scope of this tutorial.*

4. The application will remind the end user to synchronize. If the MobiLink server is not located at localhost or PPP_PEER, select **File -> Sync Options**… to enter the MobiLink host and port. If you are using an emulator, set the host entry to the numeric IP address of the machine on which the MobiLink server is running.

5. Click **OK**. This will automatically launch a synchronization request.

6. Data relevant to the employee will be downloaded to the Pocket PC after first synchronization. Changes made within the SalesDB application will be updated to the consolidated database during next synchronization.

7. You can now browse the sales data, as well as add and remove new sales orders.

This completes the tutorial. It demonstrated the foundation of building a PocketBuilder application using Adaptive Server Anywhere databases and MobiLink synchronization technology. For more information, please refer to the PocketBuilder and SQL Anywhere Studio documentation related to these technologies.

PART 5: TROUBLESHOOTING

DATABASE CONNECTION

Question: When I start the SalesDB application, the “Connect to Adaptive Server…” dialog shows up. The application cannot establish a connection to the database.

Answer: This is most likely due to an incorrect DSN file. Check SalesDB_remote.DSN located at the root directory of the device/emulator and make sure the databasefile and start properties points to the correct locations.

SYNCHRONIZATION

Question: When I initialize synchronization from the device/emulator, the MobiLink window shows the error, “Error: Protocol version mismatch”.

Answer: Check the version of ASA engine on the device/emulator and the version of MobiLink server. Both components need to have a version of 9.0.0.1108 or higher. ASA 9.0.0 build 1108 is bundled with PocketBuilder.

Question: When I initialize synchronization from the device/emulator, the MobiLink window shows the error "Communication error occurred while receiving data from the MobiLink server".

Answer: Make sure the MobiLink server is running and check the MobiLink server log for more details.

Question: When I initialize synchronization from the device, MobiLink stalls while displaying the message “Connecting to MobiLink server at ‘’ using ‘dbsock8.dll’”.

Answer: Inside the SalesDB application on the device, select File -> Sync Options… Click on the ML Server tab, then enter the host and port of the MobiLink you wish to connect to and try synchronizing again.

Question: When I initialize synchronization from the device/emulator, a dialog with the title "ASA MobiLink Synchronization" appears with the message "Error in command near "-pd"".

Answer: Make sure ASA 9.0.0 build 1108 or above is installed on the device/emulator. Previous builds do not support this switch. ASA 9.0.0 build 1108 is bundled with PocketBuilder.

SYBASE®

Sybase Incorporated
Worldwide Headquarters
One Sybase Drive
Dublin CA, 94568 USA
T 1.800.8.SYBASE
www.sybase.com