Release Bulletin
PocketBuilder™ 2.0.1

Document ID: DC50063-01-0201-01

Last revised: February 24, 2005

1. Accessing current release bulletin information

A more recent version of this release bulletin might be available on the Web. To check for critical product or document information added after the release of the product CD, use the Sybase® Technical Library Product Manuals Web site.

❖ **Accessing release bulletins at the Technical Library Product Manuals Web site**

1 Go to Product Manuals at http://www.sybase.com/support/manuals/.

2 Follow the links to the appropriate Sybase product.

3 Select the Release Bulletins link.

4 Select the Sybase product version from the Release Bulletins list.

5 From the list of individual documents, select the link to the release bulletin for your platform. You can either download the PDF version or browse the document online.

2. Product summary

Sybase PocketBuilder™ is a smart-client application development tool for Windows CE platforms. It enables developers to build handheld applications in an object-centric, graphical, desktop environment, then deploy the same applications to a supported device or emulator.

PocketBuilder version 2.0.1 (Build 688) is compatible with the following desktop platform and operating system configurations:

- Microsoft Windows 2000 with Service Pack 2 or later
- Microsoft Windows XP

The principle runtime platform for applications that you develop with PocketBuilder is Windows Mobile 2003 Second Edition (Windows Mobile 2003 SE) on Pocket PC and Smartphone devices. Support is also maintained for the Windows CE 3.0 platform on Pocket PC devices. Device and emulator support is listed in “Device and emulator support” on page 4.

New features are described in “Changed functionality in this version” on page 6. For a more comprehensive product summary, see the *Introduction to PocketBuilder*.

2.1 Version contents

2.1.1 Software used for database connection

PocketBuilder 2.0.1 has been tested with SQL Anywhere Studio® 9.0.1. The SQL Anywhere Studio setup includes Adaptive Server® Anywhere (ASA), a transaction-based, relational SQL database for the desktop and for Windows CE platforms. It also includes UltraLite®, a relational database expressly designed for small mobile and embedded devices.

The developer edition of SQL Anywhere Studio is part of the PocketBuilder installation.

2.1.2 Software used for database synchronization

PocketBuilder is integrated with MobiLink software, which can be used for two-way database synchronization between a mobile remote database and a central consolidated database. You can install MobiLink from the SQL Anywhere Studio 9.0.1 installation program.

2.2 Deployment and debugging support

2.2.1 Device and emulator support

PocketBuilder includes virtual machines (VMs) for deployment to Pocket PC and Smartphone devices and emulators with any of the following operating systems:

- arm for ARM- and XScale-based devices supporting either Microsoft Windows CE 2002 or the Windows Mobile 2003 SE operating system

  Supported devices include HP iPaq, Dell Axim, Toshiba e740, and Symbol PPT-8800.

- sparm for Smartphone 2003

  Supported devices include Audiovox 5600, SPV C500, and Motorola MPx.

- x86 for the Microsoft Pocket PC 2002 and Pocket PC 2003 emulators
- spx86 for Microsoft Smartphone 2003 emulators

You can download the supported emulators from links on the Microsoft Mobile and Embedded Application Developer Center at http://msdn.microsoft.com/mobility/downloads/sdks/default.aspx.

2.2.2 Debugging support

Debugging support for PocketBuilder applications is currently available only for the desktop.

3. Special installation instructions

3.1 Installing on the desktop

The PocketBuilder 2.0.1 setup program is packaged in a zip file that you can obtain from the Sybase Web site and save to your hard drive.

You must install PocketBuilder 2.0.1 on top of an existing installation of PocketBuilder 2.0. For information on installing PocketBuilder 2.0, see the *Installation Guide* for PocketBuilder 2.0. Use the following procedure to install PocketBuilder 2.0.1:

❖ **Installing PocketBuilder 2.0.1 on the desktop**

1. Go to the Sybase product download site at http://www.sybase.com/downloads
2. Click the EBFs/Maintenance link under the list of downloads.

   If you are not logged in to MySybase, a login page will display. The login page includes a Sign Me Up button link to open a MySybase account if you do not already have one. In some cases you are required to click Yes in a Securities Alert dialog box to continue to the download page after you log in. The EBFs/Maintenance download page displays a list of Sybase product families.
3. Click PocketBuilder in the list of Sybase product families.
4. Click PocketBuilder 2.0.1 (688) EBF Release.
5. Select the I Agree check box and click Continue on the license agreement page.
6. Click the *PocketBuilder201_688.ZIP* file in the File list.
7. Click Save in the File Download dialog box, then select a directory on your hard drive where you want to save the zip file and click Save.
8. Extract the *PocketBuilder201_688.ZIP* file to a temporary directory.

   The extracted files include the setup program file, *Setup.exe*.
9. Run the *Setup.exe* file.
10. Select Install and click Next.

    The setup program installs (or reinstalls) PocketBuilder 2.0.1.
11. Click Finish.
12. (Optional) Delete all the files you extracted to the temporary directory in Step 8.

3.2 Installing on a device or emulator

If you do not install PocketBuilder directly to a Pocket PC or Smartphone device when you run the setup program, you can manually copy and run CAB files that are installed in the *PocketBuilder 2.0\WinCE* directory, or you can use the Start menu item for installing PocketBuilder to a Windows CE device.

The setup program places the following CAB files in the *PocketBuilder 2.0\WinCE* directory, overwriting earlier versions of the CAB files if necessary:

- *PocketBuilder.ARM.CAB* for Pocket PC 2002 or 2003 devices
- *PocketBuilder.X86.CAB* for Pocket PC 2002 or 2003 emulators
- *SP_PocketBuilder.ARM.CAB* for Smartphone Windows Mobile 2003 devices
- *SP_PocketBuilder.X86.CAB* for Smartphone Windows Mobile 2003 emulators
- *BW_PocketBuilder.ARM.CAB* for WinCE 3.x and 4.x, and WinCE .Net devices
- *BW_PocketBuilder.X86.CAB* for WinCE 3.x and 4.x, and WinCE .Net emulators

To manually install PocketBuilder 2.0.1 to a device or emulator, you must copy the appropriate CAB file to the device or emulator, then run the CAB file.

4. Changed functionality in this version

4.1 SMS receive functionality

PocketBuilder 2.0 added the ability to send SMS messages from an application. With PocketBuilder 2.0.1, you can now receive SMS messages in PocketBuilder applications running on Windows Mobile 2003 platforms.

Deploying a DLL with hooks into the SMS processor

PocketBuilder provides a shim DLL, *PKSMS20.DLL*, that contains a COM object that hooks into the SMS processor. The DLL should be copied to the *\Windows* directory of a Pocket PC device or the *\Storage\Windows* directory of a Smartphone device. You must register the DLL with the operating system, then perform a soft reset on the device.

You can deploy the DLL along with registry settings in the CAB file you create for a customer application. The DLL is not part of the standard CAB file that you generate from the PocketBuilder Project painter. You can use the Enhanced CAB Generation tool to add the *PKSMS20.DLL* file to the CAB file, or you can add it manually.

Registering the shim DLL

To register the DLL, you can use the Enhanced CAB Generation tool (see "Support for SMS receiving in generated CAB files" next) or either of the following approaches:

- Call DLLRegisterServer from your PocketBuilder application
- Modify the INI for the CAB file to add the required registry keys manually

First you register the server:

```
[HKEY_CLASSES_ROOT\CLSID\{CA08D891-1E24-4c69-A313-
453B1120E558}\InProcServer32] @="PKSMS20.dll"

[HKEY_CLASSES_ROOT\CLSID\{CA08D891-1E24-4c69-A313-
453B1120E558}]ReadOnly = dword: #
```

The ReadOnly value (#) can be 0 or 1. If you register the DLL by calling DLLRegisterServer, this value is automatically set to 0 (false).

When `ReadOnly = dword: 0` and you return true for an SMSSession IncomingMessage event, PocketBuilder attempts to prevent an incoming SMS message from displaying in the SMS inbox. PocketBuilder can fail in this attempt when other phone-related activity interferes with the request for nondisplay of the incoming message. In some configuations, it is possible that the shim DLL will be prevented from operating unless the ReadOnly attribute is set to 1.

When `ReadOnly = dword: 1`, the application will be notified of incoming SMS messages, but PocketBuilder will not be able to delete the incoming messages from the SMS inbox regardless of what the IncomingMessage event returns.

Next you register the inbox processor:

```
[HKEY_LOCAL_MACHINE\Software\Microsoft\Inbox\Svc\
SMS\Rules]{CA08D891-1E24-4c69-A313-453B1120E558} =
dword:1
```

Unregistering a DLL
You can unregister the DLL by calling DLLUnregisterServer from your application. You must perform a soft reset after you unregister a DLL.

Support for SMS receiving in generated CAB files

You access the Enhanced CAB Generation tool on the Tool tab of the New dialog box. The PocketBuilder Options tab page in the Enhanced CAB Generation tool has check boxes that let you include the following in a CAB file that you deploy to a device:

- The SMS reception DLL, *PKSMS20.DLL*

  The Deploy SMS Reception DLL check box is disabled by default. To enable this option, you must select the check box to include PocketBuilder support DLLs.

- Registry entries for SMS reception in a PocketBuilder application

  If you select the Insert Registry Entries for SMS Reception check box and generate a CAB file with the Enhanced CAB Generation tool, running the CAB file on the deployment device automatically inserts the required entries in the device registry.

- A modification to the `ReadOnly` string of the SMS reception DLL registry entry

  If the check box labeled SMS Reception is Read Only is selected when you generate a CAB file with the Enhanced CAB Generation tool, unzipping the CAB file on the deployment device automatically assigns a `ReadOnly` value of 1. This makes PocketBuilder applications unable to delete incoming messages from the SMS inbox.

New values for Open function argument

In PocketBuilder 2.0.1, the *msgmodes* argument of the Open function of the SMSSession object accepts values that enable the object to receive notification of incoming messages. The syntax for the Open function remains the same as in PocketBuilder 2.0:

SMSSessionname.Open (*smsproto*, *msgmodes*)

Valid values for the *msgmodes* argument are now:

- **0** send only (retained to support PocketBuilder 2.0 code)
- **1** send only
- **2** receive only
- **3** send and receive

In order to receive SMS messages in an application, you must set the *msgmodes* argument in the Open call for the SMSSession object to 2 or 3.

SMSSession IncomingMessage event

In PocketBuilder 2.0.1, an IncomingMessage event has been added to the SMSSession object. This event occurs when an SMSSession object that is open for receiving messages is notified of an incoming SMS message.

All instantiated SMSSession objects listening for messages are notified of an incoming SMS message. PocketBuilder can prevent the message from displaying in the SMS inbox if you set the IncomingMessage event to return true. If any SMSSession object requests deletion of the message by returning true for the IncomingMessage event, PocketBuilder will attempt to delete the message, but only after all SMSSession objects have processed the message.

Deletion of messages is dependent on registry setting
PocketBuilder applications can receive notification of an SMS message only after you register the shim DLL, *PKSMS20.DLL*, on your device. The ReadOnly registry attribute for the DLL must be set to 0 before PocketBuilder can delete an SMS message. For information on registering the DLL and the ReadOnly registry attribute, see "Registering the shim DLL" on page 7.

The IncomingMessage event is synchronous with the operating system processing SMS messages. Therefore you should not include code that prompts for user input, or perform any lengthy operation in the script for this event.

Example

The following code in the IncomingMessage event determines whether an incoming SMS message is placed in the SMS inbox or is deleted. The determination is based on whether or not the message contains the text "top secret":

```
// If the message contains "top secret" it will deleted.
// Otherwise, it is placed in the inbox.

if POS(SMSMsg.text, "top secret") <> 0 then
    // delete this e-mail and notify user of deletion
    //  in a MultiLineEdit control
    mle_status.text += "~r~nWill be deleted.~r~n"
    return TRUE
end if

// allow to go into the inbox
return FALSE
```

4.2 Enhanced support for VGA screens

Two new system functions allow you to zoom in, zoom out, and obtain the current zoom factor of controls in an application running on a Windows CE device or emulator. Table 1 describes the two new system functions. These functions are not designed to work on the desktop.

Table 1: System functions for enhanced VGA support

Function syntax	Description
integer GetDisplayZoom ( )	Returns the current zoom factor.
integer SetDisplayZoom (integer *izoom*, boolean *rebuild*)	Sets the zoom factor that you enter in the *izoom* argument. To resize existing controls, the *rebuild* argument must be true. If *rebuild* is false, only newly instantiated controls are sized by the zoom factor in the *izoom* argument.

The current zoom factor that you set in a SetDisplayZoom call applies to all controls (when *rebuild* = true) or all newly instantiated controls (when *rebuild* = false) in an application. However, when *rebuild* = true, the sizes of the bitmaps for radio buttons, check boxes, and the edit boxes of drop-down lists are not changed by a SetDisplayZoom call unless they are used as display formats for columns in a DataWindow. When the same controls are placed on application windows, the sizes of these controls' bitmaps are fixed by the Windows CE operating system and can be modified only by a SetDisplayZoom call before they are loaded.

Setting the zoom factor for windows with DataWindow controls
The zoom value should be set before any dynamic changes are made to the DataWindow content since changing the display zoom value resets the DataWindow content.

The zoom factor is a percent of the size of the controls at design time. SetDisplayZoom works best for devices that have a VGA screen, such as the ASUS MyPal A730. The permissible zoom factor range is 10 to 500 percent. If you set a zoom factor outside of this range, PocketBuilder automatically resets the zoom factor to 100.

Although horizontal and vertical scroll bars are resized based on the zoom factor that you set in a SetDisplayZoom call, there is a threshold below which these controls cannot be painted. The threshold depends on the device resolution. For example, on a Dell Axim device with a 240 x 320 screen resolution, a scroll bar is not visible if its height is less than 28 pixels.

Drawing objects, such as lines and ovals, are automatically repainted with the current zoom factor when an action causes the application window to be refreshed. This occurs even if you called SetDisplayZoom with the *rebuild* argument set to false.

4.3 Support for additional Windows CE devices

PocketBuilder 2.0 added support for Smartphone devices and emulators and Pocket PC devices and emulators using the Windows Mobile 2003 platform. With PocketBuilder 2.0.1, you can deploy applications to Windows CE devices that use only the "bare" Windows CE or Windows CE .NET platform, such as the HP Jornada. The bare Windows CE platform does not include SIP functionality and other features and dependencies of the Pocket PC platform.

Menu functionality on bare Windows CE and CE .NET platforms
Menus that work correctly on Pocket PC or Smartphone platforms can cause unusual behavior on the deployment devices. Fixes to the menu functionality for these platforms will be implemented in future releases. [CR 378642]

Separate CAB files for loading the PocketBuilder VM to a bare Windows CE platform have been added to the PocketBuilder 2.0\WinCE directory. These CAB files are:

- BW_PocketBuilder.Arm.Cab (for a bare Windows CE device)
- BW_PocketBuilder.x86.Cab (for a bare Windows emulator)

CAB files that you generate for the Pocket PC Device (ARM) platform from the Project painter or the Enhanced CAB Generation tool are also valid for bare Windows CE platforms.

4.4 Support for certificate management on Windows CE devices

The Device Certificate Management tool lets you deploy certificates and review the certificate status of an attached device using the Windows Mobile 2003 platform. In this tool, you can select one of the following options as the function of an XML file to download to a connected device:

- Install PocketBuilder Default Certificates
- Query for Unprivileged Certificates
- Query for Privileged Certificates
- Custom WAP Command File

PocketBuilder provides an XML file (*PK_AddCerts.xml* in the PocketBuilder *Support>TestCertificates* directory) that it assigns as the default source to install to a connected device when the Install PocketBuilder Default Certificates option is selected. Clicking the Install Certificates button deploys this XML file which contains references to all the default PocketBuilder test certificates.

Additional XML files installed with PocketBuilder (*QueryStore_priv.xml* and *QueryStore_unpriv.xml*) can be used to query the status of the privileged and unprivileged certificates currently residing on a connected device. One of these XML files is automatically selected as the default source to download when a query option is selected for privileged or unprivileged certificates. The label for the Install Certificates button automatically changes to "Query for Certificates" when one of these options is selected.

The Device Certificate Management tool also lets you deploy a custom XML file containing information for any certificates that you want to download to a connected device. The label for the Install Certificates button automatically changes to "Execute Custom WAP File on the Device" when the Custom WAP Command File option is selected. The file selection dialog box also opens automatically when the Custom WAP Command file selection is made.

4.5 Support for device skins at design time

PocketBuilder 2.0.1 adds the ability to display windows inside a device skin at design time in the Window painter. You can select an XML file containing a device skin definition on the General tab of the Options dialog box. (You open this dialog box from the Window painter's Design>Options menu.) Clicking the browse button on the General tab displays a file selection dialog box. The file you select here is automatically entered in the Current Skin text box on the General tab of the Options dialog box.

Smartphone_Emulator_Skin.xml is an example of an XML file containing a device skin definition. This file is installed with the Smartphone 2003 SDK that you can obtain from the Microsoft Mobile and Embedded Application Developer Center at http://msdn.microsoft.com/mobility/downloads/sdks/default.aspx.

After you select a skin and click the Show Skin button, all windows that you subsequently open in the Window painter are displayed in the selected skin, and the windows are repositioned inside the skin. You can remove the skin by clicking the Hide Skin button.

4.6 Support for tracing and profiling

Creating a trace file

In PocketBuilder 2.0.1, you can use tracing and profiling functions to monitor application performance on the desktop or on a handheld device. Tracing and profiling can help you debug and tune an application. You can create a trace file when you run an application from the design-time environment or from a standalone executable at runtime. You use the trace file to create a profile of your application. The trace file must have a PKP extension to be viewed in profiling tools included with PocketBuilder.

When you run an application with tracing turned on, PocketBuilder records a timer value in a data file every time a specific activity occurs. You control when logging begins and ends and which activities are recorded.

PKDebug tracing

PKDebug tracing was available in earlier builds of PocketBuilder and Pocket PowerBuilder. You can still generate a simple text trace file without timer values by checking Enable PKDebug Tracing in the System Options dialog box. Trace files created with PKDebug tracing have a DBG extension.

For information about PKDebug tracing, see the *User's Guide* in the online Help.

Timer kinds

There are three kinds of timer: clock, process, and thread. If you do not specify a type of timer, the time at which each activity begins and ends is recorded using the clock timer, which measures an absolute time with reference to an external activity, such as the machine's start-up time. The clock timer measures time in microseconds.

To trace activities on the desktop, you can use process or thread timers, which measure time in microseconds with reference to the start of the process or thread being executed. Clock timers and thread timers are the only kinds of timers supported on handheld devices. If you select the process timer for an application running on a device, the thread timer is used instead.

Both process and thread timers exclude the time taken by any other running processes or threads so that they give you a more accurate measurement of how long the process or thread is taking to execute, but both have a lower resolution than the clock timer.

Creating profiles

After you have generated a trace file, you can create several different profiles (views) of the application by extracting different types of information from the trace file. A profile shows you:

- Which functions and events were called by which other functions and events
- How often functions and events were called
- When garbage collection occurred
- When objects were created and destroyed
- How long each activity took to complete

This information helps you find errors in the application's logic and identify areas that you should rewrite to improve performance.

PocketBuilder provides three profiling tools that create views of the application for you, but you can also create your own analysis tools. Examining the profiles generated by the profiling tools tells you where the application is spending the most time. You can also find routines that are being called too often, being called unexpectedly, or not being called at all.

Viewing profiles from a handheld device

The profiling tools included with PocketBuilder work only on the desktop. If you use the profiling tools to view a trace file from a deployment device, you must copy the executable for the application that created the trace file to a desktop directory. The desktop directory must have the same path name—with the exception of the initial drive letter—as the path where the executable resides on the device. For example, for an application residing in the \Program Files\Tests directory on a device, you must copy the executable to \Program Files\Tests on the current drive on the desktop.

You must also copy the PKP trace file from the handheld device. Although you can copy the PKP file to any location on the desktop, when you select the PKP file in a profiling tool, you also set the current drive; therefore you should copy the PKP file and the executable file to the same drive on the desktop.

Collecting trace information

There are three ways to collect trace information. You can use:

- The Profiling tab on the System Options dialog box
- A window similar to the Profiling tab
- Trace objects and functions

Whichever method you use, you can specify:

- The name and location of the trace file and optional labels for blocks of trace data
- The kind of timer used in the trace file
- The activities you want recorded in the trace file

For more information on collecting and analyzing trace information, see the chapter on tracing and profiling in the *User's Guide* in the online Help.

Tracing functions

You can use the PowerScript system functions listed in Table 2 to collect information in a trace file. Each of these functions returns a value of type ErrorReturn, an enumerated datatype.

Table 2: PowerScript trace functions

Use this PowerScript function	To do this
TraceOpen	Open a named trace file and set the timer kind.
TraceEnableActivity	Enable logging of the specified activity.
TraceBegin	Start logging all enabled activities. You can pass an optional label for the trace block.
TraceError	Log a severity level and error message to the trace file.
TraceUser	Log a reference number and informational message to the trace file.
TraceEnd	Stop logging all enabled activities.
TraceDisableActivity	Disable logging of the specified activity.
TraceClose	Close the open trace file.

Profiling functions

You use the PowerScript functions and PocketBuilder objects listed in Table 3 to analyze the performance of an application; however, this analysis must be performed on the desktop. Each of these functions returns a value of the enumerated datatype ErrorReturn. The objects listed in the table contain information such as the number of times a line or routine was executed, and the amount of time spent in a line or routine and in any routines called from that line or routine.

Table 3: Desktop-only functions for analyzing performance

Use this function	With this object	To do this
SetTraceFileName	Profiling	Set the name of the trace file to be analyzed.
BuildModel	Profiling	Build a call graph model based on the trace file. You can pass optional parameters that let you track the progress of the build.
RoutineList	Profiling and ProfileClass	Get a list of routines in the model or in a class.
ClassList	Profiling	Get a list of classes in the model.
SystemRoutine	Profiling	Get the name of the routine node that represents the root of the model.
IncomingCallList	ProfileRoutine	Get a list of routines that called a specific routine.
OutgoingCallList	ProfileRoutine and ProfileLine	Get a list of routines called by a specific routine or from a specific line.
LineList	ProfileRoutine	Get a list of lines in the routine in line order.
DestroyModel	Profiling	Destroy the current performance analysis model and all the objects associated with it.

4.7 Modifications to the Enhanced CAB Generation tool

The Enhanced CAB Generation tool was included in PocketBuilder 2.0 as the CAB Configuration tool. PocketBuilder 2.0.1 adds a PocketBuilder Options tab page and incorporates a number of changes to the other tab pages in this tool.

You can use the Enhanced CAB Generation tool as an alternative to the Project painter to generate CAB files for application distribution. The tool lets you include additional items in the generated CAB without making manual modifications to the generated file.

The tool includes check boxes for the selection of files that you want to include with the CAB file you generate. These selections include DLLs for the PocketBuilder VM, executables for the ASA runtime engine, files for ASA database connections, and registry entries to enable SMS reception capability in your PocketBuilder applications.

The Enhanced CAB Generation tool also lets you generate a setup file that can install the CAB on a connected Windows CE device. The setup file generation requires the EZSetup executable that you can download from the Spb Software House Web site at http://www.spbsoftwarehouse.com/products/ezsetup/?en. Because of EZSetup requirements, there must not be spaces in the names of the paths that you use to specify files for inclusion in the setup file.

When you generate a CAB file with the Enhanced CAB Generation tool, a log file, *err.log*, is also created in the output directory. The *err.log* file logs warnings and errors that occur during generation of the CAB or setup files.

Table 4 lists the fields available on the different tab pages of the Enhanced CAB Generation tool.

Table 4: Tab pages of the Enhanced CAB Generation tool

Tab page	Field or control, and description
Cab Config	• **Application Name** Required field for saving profile settings; can also be used in the application deployment path • **Company Name** Optional field that can be used as a directory name in the application deployment path • **Reset Deployment Path Using Company Name** Button that adds company and application name to deployment path for the CAB file • **Deployment Path** The default deployment path for the application is %CE1% (\Program Files) • **Create CABs for** Radio button options for selecting the device or emulator where you want to deploy the generated CAB file • **Directory for Application Files Included in Deployment** Directory containing the application files you want to deploy • **File Types to Include** Comma-separated list of the types of files to be added to the CAB from the appplication files directory; file extensions should be listed with a period (such as ".pkd, .dll") although files can also be specified with a standard file name • **Executable** The name of the executable file for the application • **Add Shortcut to Start Menu** Check box for deploying a shortcut for the application; this must be selected to enable the Shortcut Link field • **Shortcut Link** Displays where the shortcut is added; the shortcut must use the %CE11% (\Windows\Start Menu\Programs) special folder variable; you can add a subdirectory to this path when the field is enabled

Tab page	Field or control, and description
Database Options	• **Deploy Application Database** Check box that must be selected to enable other options on the Database Options tab page • **DSN-File** The name and desktop location of a DSN file you want to use for the application database connection • **DB-File** The name and desktop location of a database file • **DB-Log File** The name and desktop location of a database log file • **DB Directory on Device** Directory on the device for the database file when you include one in the deployment CAB • **Include ASA Support DLLs and EXEs** Check box for adding ASA DLLs and Exes to the CAB file and enabling other runtime ASA options • **ASA Source** The location of the desktop ASA • **ASA Executables Path on Device** The path on the device where you want to deploy ASA executables • **ASA Language** Two-letter code that determines the language you want to use with the database • **ASA Version** Version of the ASA database
PocketBuilder Options	• **Include PocketBuilder Support DLLs** Check box for including PocketBuilder DLLs in the CAB file you generate and enabling other options on the PocketBuilder Options tab page • **PocketBuilder Source** Desktop location of PocketBuilder support DLLs for the deployment platform you want to use • **Deploy AppList.exe** Check box for deploying the AppList utility • **AppList Location on Device** Location on the device where you want to install the AppList utility • **Deploy Remote Debugging Server** Check box for deploying the PKDebug remote debugging executable • **Deploy SMS Reception DLL** Check box for deploying the SMS reception DLL; you must select Include PocketBuilder Support DLLs to enable this check box • **Insert Registry Entries for SMS Reception** Includes registry entries for the SMS reception DLL in the CAB; the entries are added to the device registry when the CAB is unzipped • **SMS Reception is Read Only** Includes a registry entry string value that prevents deletion of an incoming SMS message by a PocketBuilder application

Tab page	Field or control, and description
Preview	• **Generate CAB Info File** Button that displays the INF file contents in the Preview window • **Copy into Clipboard** Button that copies the INF file contents to the desktop clipboard • **Save Settings** Button that saves your settings in a CAB enhancement tool profile; the profile takes the name you selected in the Application Name field on the CAB Config tab page
Build	• **Output Path** Path on the desktop where you want to generate a CAB or setup file, and INF and CMD files • **CAB-INF Name** Name of the INF file you generate for inclusion in the CAB file • **CMD File (for CAB generation)** The command file for generating the CAB • **EZSetup** Desktop location of the EZSetup executable that is required to create a setup file; you can download this executable at no charge from the Spb Software House Web site at http://www.spbsoftwarehouse.com/products/ezsetup/?en • **INF Name** Name of the INF file to generate with the setup file • **CMD File (for setup executable file creation)** Name of the command file that is automatically created and used to generate the setup file • **EULA File** End-user license agreement file that EZSetup requires for inclusion with the setup file it generates • **ReadMe File** Readme file for inclusion with the setup file; this is a required file for the setup file generation • **Language** Drop-down list of languages you can use in the setup file for the initial CAB installation screens • **Save Settings** Button that saves your settings in a CAB enhancement tool profile; the profile takes the name you selected in the Application Name field on the CAB Config tab page • **Create CAB File** Button that generates the CAB file • **Create Setup Executable** Button that generates the setup file

4.8 Support for automatic picture scaling on runtime toolbars

In PocketBuilder 2.0.1, you can control picture size in runtime toolbars by setting the PictureWidth and PictureHeight properties. The toolbar shrinks or grows to fit the height, and the toolbar buttons shrink or grow to fit the width and the height you set. Pictures you select for the buttons are automatically scaled to the picture width and height settings. This feature is particularly useful for VGA devices.

For the Toolbar control, you can set the PictureWidth and PictureHeight properties in script whether or not there are images in the PictureName property array. For the TreeView control, the PictureName array must be empty before you can set the PictureWidth and PictureHeight properties.

4.9 Fixed issues in PocketBuilder 2.0.1

The current release of PocketBuilder resolves multiple issues discovered in earlier releases.

For information about fixed change requests (CRs) in this release, see the fix list HTML file contained in the *PK201_688..ZIP* file you download from the PocketBuilder download site at http://downloads.sybase.com. The fix list HTML file is also available from the PocketBuilder download site as a separate file.

5. Known problems

5.1 Emulator issues

5.1.1 Default shutdown of Pocket PC or Smartphone emulators

By default, when you shut down a Pocket PC 2002, Pocket PC 2003, or Smartphone 2003 emulator, the emulator does not save its state. You lose any files you exported to the emulator prior to the shutdown, including the PocketBuilder VM.

To avoid this, you must select Save Emulator State after you click the Emulator>Shut Down menu item. If you shut down the emulator without saving its state, you need to re-export the CAB file with the PocketBuilder VM to the Windows directory on the emulator, then run the CAB file again.

5.1.2 Default memory configuration on the supported Pocket PC emulators

If you run ASA applications on the Pocket PC 2002 emulator, you must change the default memory configuration. You can increase the memory available to the emulator by increasing the Memory value for the following registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows CE Tool\
Platform Manager\
{F384D888-F9AA-11D1-BB9E-00A0C9C9CCEE}\
{DE9660AC-85D3-4C63-A6AF-46A3B3B83737}\
{F384D894-F9AA-11D1-BB9E-00A0C9C9CCEE}\
{67C8D913-F0CF-486A-8CF0-CE7D116225E8}
```

There is a similar, but different, setting for the Pocket PC 2003 emulator. Sybase recommends that you increase this value from 16MB to 64MB.

5.2 PocketBuilder issues

5.2.1 Enabling Manual Result Set with a Stored Procedure data source

During the creation of a DataWindow using the Stored Procedure data source, selecting the Manual Result Set option in the Select Stored Procedure wizard page produces a wizard page that is blank except for an unlabeled command button on the top left corner of the page. Clicking Next on this blank wizard page can cause the wizard to fail or PocketBuilder to crash. [CR 380430]

5.2.2 Selecting functions in Profiling Class View tool

Selecting a function in the Profiling Class View tool when the function script exceeds approximately 1000 characters (Windows 2000) or 1800 characters (Windows XP) can cause PocketBuilder to crash. After viewing a function in the Profiling Class View tool, modifying or adding a comment to the same function in the Function painter and attempting to save the modified function can also cause PocketBuilder to crash. [CR 380153 and CR 380154]

5.2.3 Creating a DataWindow with QuickSelect data source and a new database

Creating a new DataWindow using the Quick Select data source and a newly created ASA database causes the DataWindow creation wizard to fail. The error message indicates a syntax error or access violation. [CR 380151]

5.2.4 Deleting a custom visual object associated with a tab control

If you insert a custom visual object on a tab control and subsequently delete the window containing the tab control, you must restart PocketBuilder before you can successfully delete the custom visual object. If you attempt to delete the custom visual object without restarting PocketBuilder, an error message indicates that the custom visual object or one of descendants is open in another painter—even if no painters are open at this time. [CR 370801]

5.2.5 DialingDirectory object functions

The UpdateEntry function currently works for a SIM directory only, not for a POOM directory. The AddEntry function is not currently implemented either for a SIM or a POOM directory. [CR 371011]

5.2.6 Smartphone CAB files do not include DLL for ODBC connections

Because Smartphone applications typically use UltraLite, the CAB files you generate from the Project painter for deploying applications to a Smartphone device or emulator do not include the *PKODB20.DLL* file required for ODBC database connections. If you use an ODBC connection to an ASA database in an application, you must manually copy the *PKODB20.DLL* file or use the Enhanced CAB Generation tool to create the CAB file.

If you create the CAB file with the Project painter, you can copy the *PKODB20.DLL* file from the PocketBuilder 2.0\WinCE\sparm directory to the Smartphone device where you deploy your application, or from the PocketBuilder 2.0\WinCE\spx86 directory to a Smartphone emulator. If you use the Enhanced CAB Generation tool, you must select the Include PocketBuilder Support DLLs check box on the PocketBuilder Options tab page.

UltraLite connections do not require the *PKODB20.DLL* file. The *PKUL20.DLL* file required for UltraLite connections is included automatically in the CAB files you generate from the Project painter. [CR 371146]

5.2.7 Calling a deeply recursive function can cause a stack overflow

The Windows CE platform does not have any built-in protection to manage memory during calls to functions that call themselves (recursive functions). Calls to deeply recursive functions can cause the operating system to crash due to stack overflow.

5.2.8 Writing to a file with the SetProfileString function

You can write to a file with the SetProfileString function, but only if the file name you pass in the first function argument refers to a file that already exists and is a valid Unicode file. A valid Unicode file must have at least one character written to the file.

The following code sample uses regular file functions to create a valid Unicode file by adding a comment to the top of the file:

```
integer li_file
integer li_ret
li_file = FileOpen(gs_FileName,  LineMode!,  Write!, &
    LockWrite!,  Replace!)
li_ret = FileWrite( li_file, "; my comment" )
FileClose(li_file)
```

The ProfileInt and ProfileString functions also require references to valid Unicode files. [CR 328631]

5.2.9 Restarting a debugging session

When you debug an application for the second time in the same PocketBuilder session, certain actions, such as changing a watch variable, can cause PocketBuilder to crash. You can avoid crashes by closing and restarting PocketBuilder before running the debugger a second time. [CR 325369]

6. Product compatibilities

6.1 SQL Anywhere Studio

PocketBuilder 2.0.1 has been tested with Sybase SQL Anywhere Studio 9.0.1; however, you can use Sybase SQL Anywhere Studio 8.0.2 Build 4229 or later for database and MobiLink integration support. UltraLite support is available only with version 9 or greater.

6.2 ActiveSync software

Microsoft ActiveSync is required for transferring applications to your mobile device. ActiveSync 3.8 is available as a free download on the Microsoft Web site at http://www.microsoft.com/mobile/pocketpc/downloads/default.asp.

6.3 Afaria software

If you want to make an application available to an Afaria® software channel, you must copy the CAB file for the PocketBuilder target to a Universal Naming Convention (UNC) location where the software for the channel resides. You should refresh the channel contents after copying the CAB file.

If you want to deploy EXE and PKD files, you must clear the Delete Temporary Files After Build check box of the Select Platform to Build section of the Project object before you build your application. (This check box is selected by default.) These files must also be copied directly to the UNC location.

Future versions of PocketBuilder will have Afaria-specific application and package deployment features.

7. Migration information

Migration of a Pocket PowerBuilder 1.x application occurs automatically when you open the application in PocketBuilder 2.0.1. No migration is required for PocketBuilder 2.0 applications.

7.1 Conversion of PowerBuilder applications

You can use the Import Desktop to CE wizard to convert an existing PowerBuilder 7, 8, 9, or 10 application to a PocketBuilder 2.0.1 application. For more information, see Appendix B in the PocketBuilder *User's Guide*.

8. Technical support

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you have any questions about this installation or if you need assistance during the installation process, ask the designated person to contact Sybase Technical Support or the Sybase subsidiary in your area.

9. Other sources of information

Use the Sybase Getting Started CD, the SyBooks™ CD, and the Technical Library Product Manuals Web site to learn more about your product.

- The Getting Started CD contains release bulletins and installation guides in PDF format and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD you need Adobe Acrobat Reader, which is downloadable at no charge from the Adobe Web site, using a link provided on the CD.
- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access technical information about your product in an easy-to-use format.
- The Technical Library Product Manuals Web site is an HTML version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to the Technical Documents Web site (replacement for the Tech Info Library), the Solved Cases page, and Sybase newsgroups.

  To access the Technical Library Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

9.1 Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ **Finding the latest information on product certifications**

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Select Products from the navigation bar on the left.
3. Select a product name from the product list and click Go.
4. Select the Certification Report filter, specify a time frame, and click Go.
5. Click a Certification Report title to display the report.

❖ **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Click MySybase and create a MySybase profile.

9.2 Sybase EBFs and software maintenance

❖ **Finding the latest information on EBFs and software maintenance**

1 Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.

2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.

3 Select a product.

4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the “Technical Support Contact” role to your MySybase profile.

5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.