

Satellite Forms KnowledgeBase



© 2010 Thacker Network Technologies Inc.

Table of Contents

Introduction	6
Known Issues	9
Bitmap buttons behave differently on different PalmOS versions	9
DLL error when installing application at Hotsync	12
Error compiling PocketPC target on PC with ActiveSync 4.0	13
Error locating third party PocketPC extensions when loading project	16
Error: Referenced control not on current page	17
Find In Project does not always work	18
Known Issues for Satellite Forms 6.1	19
PocketPC device crashes when Symbol control used in app on non-Symbol	21
PPC device	
Problem with alternate-shape checkboxes in PocketPC targets created from	22
Palm target	
Error C016: Method 'Controls.SetPosition' takes 4 param(s) : 1 specified	23
SocketScan PalmOS control does not work with laser SDIO scanner	25
Bug in Binarysearch function with PocketPC PDB	26
PromptCustom may cut off some text on PocketPC	27
PocketPC target requires icon bitmap to compile even though it is not used	28
Problems printing scripts in App Designer	29
PocketPC app is relaunched after it is closed	30
An invisible 'hotspot' button is still clickable on PocketPC	31
Ink control does not allow bitmap overlay on PocketPC	32
PocketPC CDB application cannot handle more than 127 fields per table	33
PocketPC PDB listbox problems with more than 127 table fields	34
Pen tap on Text or Lookup control generates 2 OnPenDown events on	35
PocketPC	
Cannot set droplist caption in code on PocketPC	36
Form Scrollbars do not Appear Automatically in PocketPC	37
Form needs to be tapped by pen before accepting keyboard input on PocketPC	38
Deleting a record causes 'Error 30: Table won't open or invalid' on PocketPC	39
PDB	
Janam XP20/XP30 scanner powers off but needs to be reset to turn back on	40
Symbol MC50 scanner shows Error Enabling Scanner Library after scanning for	41
a while	
PocketPC PDB problems filtering a record using a 10 digit numeric field	42
Binarysearch function is not case sensitive on PocketPC PDB	43
Binarysearch function returns incorrect row number when no match on	44
PocketPC PDB	
PocketPC RemoveFilter function does not remove filter on droplist table	45
PocketPC project settings automatically change to MDB desktop DB format	46
when project is loaded	
Windows Vista Compatibility Issues	47
How-To Guides	49
How To use SatSync to send data to the Palm device	49
How To use different platform targets for PocketPC applications	50

How To use Global Functions & Subs to replace extension functions not available on the current target platforms	52
How To Install the SatForms Runtime for PocketPC Programmatically	54
How To make Satellite Forms 6.1 PocketPC applications launch faster	56
How To use color bitmaps in your application	58
How To use SatSyncPPC to send data to the PocketPC device	62
How To use the Ink View OCX to display uploaded signatures on the PC	63
How To Install SatForms PocketPC Runtime to multiple handhelds	73
How To Install the SatForms Runtime for Palm silently	77
How To use the SatForms ActiveSync control with Delphi	79
How To Create an Installer for your SatForms 6.x PocketPC Applications	83
How To support multiple languages using build targets	94
How To Change a control color using SFControlMagic	96
How To use the PocketPC Emulator with Satellite Forms	97
How To use PalmDB (PDB) tables in a PocketPC application	100
How To use High Density Bitmaps in PalmOS applications	103
How To use High Density Icons for your PalmOS applications	105
How To support Expandable Screens in PalmOS applications	107
How To Move and Resize Controls at Runtime	118
How To use the SFConvertPDB utility	120
How To use SatSyncPPC to sync PocketPC data	127
How To Change Control Fonts at Runtime	129
How To Enable a User to Interrupt a Closed Loop	131
How To Bundle the SatForms PocketPC runtime engine with your app	133
How To Create an Installer for your SatForms 7 PocketPC Application	135
How To Create a shortcut to your PocketPC application	149
How To Implement a Quick Find feature	152
How To Use OnPenDown/OnPenUp Scripts to Detect Pen Taps on Controls	154
How To Limit Edit Control Input to Numeric Only	156
How To Force Input To ALL CAPS	158
How To Dial a Phone Number Using the LaunchURL Extension	160
How To Send an Email Message Using the LaunchURL Extension	161
How To Commit Table Data to Storage Immediately	162
How To Make PocketPC PDB Apps Close Faster	166
How to Beam Files via IR or Bluetooth on PocketPC	169
How To Sync Satellite Forms Data to a Linux Server With jSyncManager	171
How To Use the MSR Attachment with Janam XP Scanners in Satellite Forms	174
How To Insert New Records Into a Sorted Table	175
How To Use Google Maps for Windows Mobile from your Satellite Forms application	177
How To Use a Specific Connection using ConnectionMgr	180
How To Enable Newer 2D Barcode Types on Janam Scanners	182
How To Change The CreatorID of a Palm Extension PRC To Match Your App	185
QuickStart Guides	189
Barcode Scanning on Janam XP PalmOS Scanners	189
Barcode Scanning on Symbol Windows Mobile/PocketPC Scanners	196
Barcode Scanning on Aceeca Meazura PalmOS Scanners	203
Barcode Scanning on Intermec Windows Mobile/PocketPC Scanners	209
Barcode Scanning on Unitech Windows Mobile/PocketPC Scanners	215

Introduction



The Satellite Forms KnowledgeBase is a searchable collection of known problems and solutions, as well as How-To guides for Satellite Forms. It covers Satellite Forms 6.x, 7.x and 8.x.

Known problems and suggested solutions are listed in the Known Issues category. How-To guides are listed in the How-To category.

You can search the KnowledgeBase by keyword, via the Index tab, or search the full text via the Search tab.

The KnowledgeBase is available online at <http://www.satelliteforms.net/knowledgebase.htm> and can also be downloaded as a Windows help file for offline use.

If you encounter a suspected bug or other issue with Satellite Forms, and cannot locate a solution in the KnowledgeBase, we encourage you to report it to our support department, via the technical support request form on our website: <http://www.satelliteforms.net/supportform.htm>

Revision History

Version	Updated Date	Description of changes
V2.3	June 29, 2010	Added How To Use Google Maps for Windows Mobile from your Satellite Forms application Added How To Use a Specific Connection using ConnectionMgr Added How To Enable Newer 2D Barcode Types on Janam Scanners Added How To Change The CreatorID of a Palm Extension PRC To Match Your App Updated several articles to mention new information for Satellite Forms 8
V2.2	June 26, 2008	Added How To Insert New Records Into a Sorted Table Added How to Beam Files via IR or Bluetooth on PocketPC to revision history (article added in V2.1)
V2.1	May 16, 2008	Added Windows Vista Compatibility Issues Added How To Sync Satellite Forms Data to a Linux Server With jSyncManager Added How To Use the MSR Attachment with Janam XP Scanners in Satellite Forms Updated Barcode Scanning on Symbol Windows Mobile/PocketPC Scanners with information about compatibility with the Janam XM60 scanner
V2.0	Dec 3, 2007	Added new QuickStart Guides for Barcode Scanning with Janam XP PalmOS scanners , Symbol Windows Mobile/PocketPC scanners , Aceeca Meazura PalmOS scanners , Intermec Windows Mobile/PocketPC scanners , and Unitech Windows Mobile/PocketPC scanners Updated several How To and Known Issues articles with new information relating to improvements in Satellite Forms 7.1

V1.9	Sept 13, 2007	Added How To Commit Table Data to Storage Immediately Added How To Make PocketPC PDB Apps Close Faster
V1.8	July 11, 2007	Updated several How To and Known Issues articles including How To use the SFConvertPDB utility Added How To Implement a Quick Find feature Added How To Use OnPenDown/OnPenUp Scripts to Detect Pen Taps on Controls Added How To Limit Edit Control Input to Numeric Only Added How To Force Input To ALL CAPS Added How To Dial a Phone Number Using the LaunchURL Extension Added How To Send an Email Message Using the LaunchURL Extension Added PromptCustom may cut off some text on PocketPC Added PocketPC target requires icon bitmap to compile even though it is not used Added Problems printing scripts in App Designer Added PocketPC app is relaunched after it is closed Added An invisible 'hotspot' button is still clickable on PocketPC Added Ink control does not allow bitmap overlay on PocketPC Added PocketPC CDB application cannot handle more than 127 fields per table Added PocketPC PDB listbox problems with more than 127 table fields Added Pen tap on Text or Lookup control generates 2 OnPenDown events on PocketPC Added Cannot set droplist caption in code on PocketPC Added Form Scrollbars do not Appear Automatically in PocketPC Added Form needs to be tapped by pen before accepting keyboard input on PocketPC Added Deleting a record causes 'Error 30: Table won't open or invalid' on PocketPC PDB Added Janam XP20/XP30 scanner powers off but needs to be reset to turn back on Added Symbol MC50 scanner shows Error Enabling Scanner Library after scanning for a while Added PocketPC PDB problems filtering a record using a 10 digit numeric field Added Binarysearch function is not case sensitive on PocketPC PDB Added Binarysearch function returns incorrect row number when no match on PocketPC PDB Added PocketPC RemoveFilter function does not remove filter on droplist table Added PocketPC project settings automatically change to MDB desktop DB format when project is loaded
V1.7	February 1, 2007 January 26, 2007	Updated How To use the PocketPC Emulator with Satellite Forms Added Bug in Binarysearch function with PocketPC PDB Updated Error compiling PocketPC target on PC with ActiveSync 4.0 Added SocketScan PalmOS control does not work with laser SDIO scanner Added How To Implement a Quick Find feature Added How To Create a shortcut to your PocketPC application Updated How To Create an Installer for your SatForms 7 PocketPC Application

V1.6	November 3, 2006	Added How To Bundle the SatForms PocketPC runtime engine with your app Added How To Create an Installer for your SatForms 7 PocketPC Application
V1.5	October 2, 2006 September 18, 2006	Added Error C016: Method "Controls.SetPosition" takes 4 param(s) : 1 specified Added How To Change Control Fonts at Runtime Added How To Enable a User to Interrupt a Closed Loop Added How To use PalmDB (PDB) tables in a PocketPC application Added How To use High Density Bitmaps in PalmOS applications Added How To use High Density Icons for your PalmOS applications Added How To support Expandable Screens in PalmOS applications Added How To Move and Resize Controls at Runtime Added How To use the SFConvertPDB utility Added How To use SatSyncPPCPDB to sync PocketPC data
V1.4	February 17, 2006 February 16, 2006	Added How To use the PocketPC Emulator with Satellite Forms Reinstated How To make Satellite Forms 6.1 PocketPC applications launch faster "SYS_POSIDX" article Added How To support multiple languages using build targets Added Problem with alternate-shape checkboxes in PocketPC targets created from Palm target Added How To Change a control color using SFControlMagic
V1.3	December 7, 2005	Added How To Create an Installer for your PocketPC Application article
V1.2	December 5, 2005	Added article on using SF ActiveSync control with Delphi Removed the " SYS_POSIDX " article
V1.1	November 17, 2005	Added articles on bitmap buttons , ActiveSync 4.0 , PocketPC targets , and PalmOS Runtime silent installation
V1.0	November 4, 2005	First release

Copyright (C) 2005, 2010 Thacker Network Technologies Inc.

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Known Issues

Bitmap buttons behave differently on different PalmOS versions

Problem: Bitmap buttons behave differently on different PalmOS versions. To create graphical button controls, developers often use a "clear" button control (a visible button with no label and no border) with a bitmap control overlaid on top, referred to herein as a bitmap button. This enables you to design more visually appealing applications. However, the behaviour of bitmap buttons unfortunately differs depending on the version of PalmOS on the device.

With PalmOS versions prior to 4.0, a clear button placed underneath a bitmap control would receive the pen tap on the form, thus enabling you to take action when the user tapped on the bitmap image. However, starting with PalmOS 4.0, PalmSource changed this default behaviour so that the foreground control receives the pen tap. In this case, the pen tap would be directed to the bitmap control, which does not respond to click events, thus nothing would happen when the user tapped on the bitmap image. For PalmOS 4.0 and higher devices, including PalmOS 5.x devices, the clear button must be placed overtop of the bitmap control in order to receive the pen taps from the user.

Solution: If your application is designed solely for devices running PalmOS 3.5, you can place clear buttons underneath bitmap controls in order to implement bitmap buttons in your application. It is unlikely, however, that many applications are intended solely for PalmOS 3.5 devices.

If your application is designed solely for devices running PalmOS 4.0 or higher, for example PalmOS 5.x devices only, then you can implement bitmap buttons by placing the clear button controls overtop of the bitmap controls.

If you wish to support all PalmOS devices running PalmOS 3.5 and higher, there are four options for implementing bitmap buttons:

1. "Button Sandwich" option

In order to support bitmap buttons on PalmOS 3.5 and higher devices, you can place clear button controls both underneath the bitmap control and overtop the bitmap control. We use the term "button sandwich" as the bitmap control is "sandwiched" between two clear button controls. Often, developers will have one of the button controls in sandwich call the .execution method of the other button control, instead of duplicating the action or OnClick script on both buttons. This makes code maintenance easier since you do not need to make changes to both button controls.

This is the simplest and most efficient method to implement bitmap buttons that behave the same regardless of the PalmOS version.

2. OnPenDown/OnPenUp handling

Rather than implementing bitmap buttons with a clear button control, you can handle the pen down and pen up events on the form directly. This enables you to process pen taps within the bounds of the bitmap control without using a clear button at all. While this method offers the advantage of working the same across all PalmOS versions, it does have some distinct disadvantages that may make it impractical in your application. In particular, the OnPenDown/OnPenUp scripts needed to handle the pen taps are much more complex than using a clear button or button sandwich, especially when you have more than one bitmap button on a

form. As well, the bitmap control is not visually inverted when the pen is touching the screen, as it is when clear buttons are used, so there is less visual feedback to the user that tapping on the bitmap will make it act like a button.

Part of the complexity comes from the fact that in order to properly behave like a button, you must track where the pen touches the screen (in the OnPenDown event) and follow it until it is lifted from the screen (the OnPenUp event), and only take action if the pen both touched and was lifted from the correct screen coordinates. If the pen touches down in the desired screen location, but the user slides it away out of the bitmap location before lifting, you should not consider that a valid pen "tap".

A sample set of OnPenDown and OnPenUp scripts to handle a "home" icon in the upper left corner of the screen is presented below. They use a pair of global variables named gPenX and gPenY.

```
'OnPenDown track pen for home icon
GetPenStatus(gPenX, gPenY)
if (gPenX >= 146) and (gPenY <= 12) then
    while GetPenStatus(gPenX, gPenY) = true
        'do nothing except update the gPenX and gPenY location vars for OnPenUp checking
    wend
endif

'OnPenUp track pen for home icon
if (gPenX >= 146) and (gPenY <= 12) then
    beep(7)           'click
    forms("Main").show   'go home!
endif
```

As you can see, adding support for multiple bitmap buttons using this method will soon become unwieldy, as you will need to track multiple bitmap location boundaries.

3. SFOnClick custom control

SFOnClick is a custom control extension available from PalmDataPro. See the SatForms Solutions Guide entry for SFOnClick here: <http://www.satelliteforms.net/solutions/?10054>

SFOnClick can be used to implement bitmap buttons that behave the same on PalmOS versions 3.5 and higher. However, this is again more complex than using the button sandwich approach, and requires the use of a commercial extension. In addition, the bitmap control is not inverted visually when the pen is touching it, thus there is less visual feedback to the user. See the sample application that comes with SFOnClick for implementation details.

4. SFBitmapView/SFJPGView custom control

SFBitmapView (<http://www.satelliteforms.net/solutions/?10061>) and SFJPGView (<http://www.satelliteforms.net/solutions/?10035>) are custom controls available commercially from PalmDataPro that present an alternative to the standard bitmap control in Satellite Forms. One of the options that these controls have is the ability to respond directly to pen taps, by providing an OnClick event for the control. The disadvantage of using this approach is that it requires the purchase of a custom control extension.

Keywords: bitmap, button, sandwich, onclick, icon

KB ID: 10023

Updated: 2006-10-02

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

DLL error when installing application at Hotsync

Problem: Your PalmOS application uses third party extensions. When trying to install the application you receive the following error message: Satellite Forms DLL error 1050: Couldn't copy file to install directory.

Solution: Your issue with the third party extension and DLL error is most likely the result of the extension file location changing with SatForms V5 and higher.

In versions older than 5.x, you would have placed the INF and PRC and possible BMP files in \Satellite Forms\Extensions\{manufacturer}

For Satellite Forms 5.x and higher, they need to go in a new location (as a result of PocketPC support in SatForms).

INF and BMP file:

\Satellite Forms\Extensions\{manufacturer}

PRC file (for a PalmOS extension):

\Satellite Forms\Extensions\{manufacturer}\Palm

SFX file (for a PocketPC extension):

\Satellite Forms\Extensions\{manufacturer}\PPC_ARM

Close App Designer, move the files, then restart App Designer and hotsync again to correct the error and install the application.

Keywords: DLL, error, Hotsync, extension, 1050

KB ID: 10011

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Error compiling PocketPC target on PC with ActiveSync 4.0

Problem: On a development PC with ActiveSync 4.0, attempting to compile a PocketPC target with CDB databases fails with an error message "SFrmUt Error - Error - Unable to delete \Temp\SFTempTable.cdb. Path: \Temp\SFTempTable.cdb". It is followed by a few more error messages.

Solution: Microsoft ActiveSync 4.0 comes with new Windows Mobile 5 based devices, while ActiveSync 3.x comes with Windows Mobile PocketPC 2002/2003/2003SE devices. This problem occurs when trying to compile an application with a PocketPC target (PocketPC 2002, PocketPC 2003, WinMobile 5, WinCE.NET) that uses the Microsoft CDB device database format to any PocketPC device. ActiveSync 4.0 is required to synchronize WinMobile 5 devices.

This article is updated Feb 1, 2007 with an easier solution to this problem.

There are known problems using Microsoft CDB databases on Windows Mobile 5 devices with ActiveSync 4.x. These problems are caused by changes to ActiveSync that Microsoft made in version 4.x, which is required for WM5 devices. With Satellite Forms, one way that this problem manifests itself is when you go to compile a PocketPC project that includes CDB database tables, with a connected WM5 device/emulator. App Designer attempts to use the CDB database conversion services of ActiveSync to generate the device database tables, and ActiveSync 4 will error out, causing the device db creation to fail. This results in a somewhat cryptic error in App Designer: "SFrmUt Error - Error - Unable to delete \Temp\SFTempTable.cdb. Path: \Temp\SFTempTable.cdb". That is usually followed by a few more error messages. The end result is that the compile process fails due to the device database creation failure.

This is not an issue with Satellite Forms 7 and higher which supports [PDB database files on PocketPC](#). PDBs offer numerous advantages over CDBs and we highly recommend that solution, but if you're using SF 6.x then the PDB solution will not be available to you.

[Old Information - Obsolete with Satellite Forms 8]

The original KB article on this topic discussed a possible solution of uninstalling AS4, installing AS 3.8, and then installing AS4 over that without uninstalling. Sometimes this works, sometimes it does not. It is also difficult to locate the older ActiveSync 3.8 installer these days.

Now, a fellow Satellite Forms developer Tim Fischer informed us of a new, simpler way to solve this problem. We've tested this solution here, and it appears to work for all of our tests. If you are faced with this issue, we encourage you to try this suggestion and report your results.

The CDB compile problem can be solved by applying some registry changes to the PC that the PocketPC device is connected to. Create a text file named AS4CompileRegFix.reg with these contents:

--- cut here ---

Windows Registry Editor Version 5.00

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows CE Services\SharedDLLs]
"C:\Program Files\Microsoft ActiveSync\cefstore.dll"= dword:00000000
"C:\Program Files\Microsoft ActiveSync\adofiltr.dll"= dword:00000000
```

```
"C:\\Program Files\\Microsoft ActiveSync\\inetrepl.dll"= dword:00000000  
"C:\\Program Files\\Microsoft ActiveSync\\inkreg.dll"= dword:00000000  
"C:\\Program Files\\Microsoft ActiveSync\\pwdreg.dll"= dword:00000000  
"C:\\Program Files\\Microsoft ActiveSync\\pwiofcnv.dll"= dword:00000000  
"C:\\Program Files\\Microsoft ActiveSync\\pwireg.dll"= dword:00000000  
"C:\\Program Files\\Microsoft ActiveSync\\pwoffcnv.dll"=dword:00000000
```

--- cut here ---

Disconnect the PocketPC device. Apply these registry changes by double-clicking on the .reg file you just created. You will be asked to confirm the registry changes. Once they have been confirmed, you can reconnect the PDA, and you should be able to successfully compile your PocketPC project without any CDB creation errors!

If this tip helps you out, be sure to let Tim know:

Tim Fischer
Enlighten Development LLC
Secure Solutions Since 1991

edev@hotmail.com
<http://www.enlighten-dev.com>

The older KB article information is archived below.

This error only occurs on development PCs that did not have a version of Microsoft ActiveSync 3.x installed prior to version 4.0.

A workaround to solve this problem on the development PC is to do the following:

1. Uninstall Microsoft ActiveSync 4.0 using the Add/Remove control panel. Restart the PC if prompted.
2. Install Microsoft ActiveSync 3.8 downloaded from the Microsoft PocketPC support website:

<http://www.microsoft.com/windowsmobile/downloads/activesync38.mspx>

3. Without uninstalling ActiveSync 3.8, install ActiveSync 4.0 (i.e. upgrade to AS 4.0). The installer should detect the current version and ask if you wish to replace the current installation of ActiveSync. Click Next to proceed with the upgrade. Restart the PC if prompted.

You should now be able to successfully compile PocketPC targets using Microsoft ActiveSync 4.0, to PocketPC, WinMobile 5, and WinCE.NET targets. You should be able to successfully sync with all of your PocketPC devices.

Keywords: ActiveSync, Windows Mobile 5, WinMobile 5, PocketPC, SFTempTable.cdb, SFrmUt

KB ID: 10022
Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Error locating third party PocketPC extensions when loading project

Problem: You may receive an error locating third party PocketPC extensions when loading a project that uses the extensions, or you may see the extension listed in the extension manager but it is greyed out and unavailable for use.

Solution: Starting with Satellite Forms 6.0, there has been a minor change in the format of the extension INF file for PocketPC extensions. SatForms 6.x will not read the INF file properly for older PocketPC extensions, and therefore cannot load the extension when the project loads. The extension INF files for extensions included with Satellite Forms were updated to use the new INF format, so they are loaded properly.

Fortunately, it is easy to fix this problem for older third party extensions:

1. Close App Designer.
2. Make a backup of your PocketPC extension INF files. They are located in \Satellite Forms\Extensions\{manufacturer} where {manufacturer} would be something like PalmDataPro. Set the file extension of the backup to something other than .INF.
3. Open the INF file in a text editor, locate the line that starts with "Target=".
4. Change the string "PPC_2002_ARM" to "PPC_ARM".
5. Save the modified INF file.
6. Rename the \Extensions\{manufacturer}\PPC_2002_ARM folder to PPC_ARM.
7. Restart App Designer and your project should find the extension properly.

Keywords: extension, PocketPC, INF, error

KB ID: 10012

Updated: 2005-11-17

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Error: Referenced control not on current page

Problem: Your application displays an error message Error: Referenced control not on current page

Solution: What the error means is that a script on that form is accessing a control that is not on the current page of the form. For example, if you have a multipage form with a control named editName on the second page of the form, and you try to access that control when you are on the first page of the form, that error message will be displayed.

You must take care to ensure you are on the same page as the control you are accessing, by using the forms().currentpage property, eg:

```
if forms().currentpage = 1 then
  'we are on second page (zero indexed) so we can access control
  editName = "Jim"
endif
```

This applies to all scripts on the form, such as AfterOpen, AfterLoad, OnValidate, etc., as well as global subs and functions.

Keywords: error, control, page, currentpage

KB ID: 10013

Updated: 2006-10-02

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Find In Project does not always work

Problem: Find In Project does not always work. If you select 'Find in Project' from the Edit menu, and enter a string, the program will search the current script page. If the string is found on the page, it will then continue if you select 'Find Next'. If it is not on that page, it will go no further & say it is not found. This is not the desired behaviour: it should go on and search the next script, and continue on throughout the project.

Solution: A workaround is to use the Edit | Replace function to find without replacing. Select the "Replace In: Entire project" option, but click on Find Next instead of Replace. A keyboard shortcut to the Replace function is CTRL-H.

Status: OPEN This bug has not been resolved yet.

Keywords: Find, Replace, Search, Edit

BugID: SF-00091

KB ID: 10017
Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Known Issues for Satellite Forms 6.1

Known Issues for Satellite Forms 6.1

- If a SatForms application has custom menubars defined, users must add at least one item to the custom menubar before downloading the application on Palm devices. Otherwise, a Fatal Exception is displayed when launching the menu on Palm. (24393, 24037)
- A limitation of the ActiveX control used for the App Designer table editor allows only tables with fewer than 514,100 characters to be displayed or saved. Table data that exceeds 514,100 characters will be truncated and saved in a temporary file. (25115)
- For the PPC platform, there is a known issue with the Symbol Control barcode scanner extension. When an application containing that extension is run on a non-Symbol device, an error message is displayed and the application will not run. See the Satellite Forms KnowledgeBase for the solution.
- Due to a known issue with Pocket PC 2002 devices, you must close an application on the device before you can synchronize the application.
- Due to limitations in the Palm OS, clear buttons must be implemented differently, depending on the Palm OS version. For Palm OS 3.5 or earlier, the clear button must be set to the back. For Palm OS 4.0 or greater, the clear button must be set to the front. You can place clear buttons both behind and in front of the overlapping control in order to work on both Palm OS 3.5 and Palm OS 4.0 and higher. (14514)
- The application definition database (ExxxxMMnn\$AppName.pdb) contains definitions of objects that runtime engines will use to render your Satellite Forms application on the Palm. Because the Palm OS limits the size of individual records to 64K, individual records in this database cannot exceed 64K in size. App Designer will display an error message when this limit is exceeded. If your application exceeds this limitation, you will need to separate the information into multiple fields.

Objects that are contained in this database are: form attributes, control attributes, table & column attributes, and compiled scripts.

- The Bitmap and resource extensions will not work with Palm OS 5.0.
- The Bitmap extension will not compile with Palm OS 5.0 SDKs. The SDKs for Palm 4.1 and lower should be used.
- SF 3.1 applications can use a lot more fields in a single application than allowed by SF 3.5 and later. This situation may be corrected in future Satellite Forms versions.
- You must specify all 4 icon files in your project for Palm OS (large and small monochrome and color icons). Missing icons will appear as "hearts" on the device.

- Applications or databases installed using HotSync Manager 4.0.0 will always have the backup bit set. This overrides the settings in your project properties. Palm has not issued a fix for this problem.
- Importing Tables containing Number fields that use "Replication ID" as field sizes does not work. This problem may be corrected in future Satellite Forms versions.
- Checkboxes on Pocket PC 2002 targets will sometimes revert to a height of 11 from their default height of 18. To reset the checkbox height to the default, click the form and then double-click the checkbox.
- The Symbol Control Extension's Aim() function is not supported on Symbol's Pocket PC 2002-based devices.
- Using "Download App & Tables to Handheld" will not rebuild the app if you just saved your project. If you have saved since your last change you must use "Rebuild All" to incorporate your recent changes into the app you wish to download.
- Adding a new target to your project will not inform App Designer that it needs to save when it closes. After adding a new target to an App Designer project always remember to save.
- The Color Graphics extension for the Pocket PC platform uses a different INF file than Palm OS Color Graphics extension. Palm targets using this extension will not automatically transfer this extension to a derivative Pocket PC target.
- Changing the data source a radio button is linked to without first specifying an index value for the radio button will result in errors. Avoid errors by specifying an index value for your radio button.
- When compiling an application, if the App Designer stops responding and the following error message appears: "AppDesigner: Out of virtual memory", more total memory (RAM + Virtual Memory) is required to compile the application. Any ratio of physical memory to virtual memory can be used; for example, either 128MB RAM/570 MB Virtual Memory or 256 MB RAM/440 MB Virtual Memory.

Keywords: bugs, known issues

KB ID: 10019

Updated: 2005-11-17

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

PocketPC device crashes when Symbol control used in app on non-Symbol PPC device

Problem: PocketPC device crashes when Symbol control used in app on non-Symbol PPC device

Solution: There is a workaround available: our technical support department can supply you with a "fake" version of the Symbol Integrated Scanner extension for Pocket PC that returns a "FALSE" value for the IsSymbolUnit function. You can install this SFX file on non-Symbol devices, and check the IsSymbolUnit function before using any of the other scanner control methods. In this way, you can use the same application (with the special extension SFX file) on non-Symbol devices without crashing.

We expect to resolve this in a future revision to allow the inclusion of this barcode scanner control on non-Symbol PocketPC devices.

Status: RESOLVED This problem has been resolved with the release of an updated SFE_Symbol_Control.sfx extension file with Satellite Forms 7.1. The SF 7.1+ versions of the Symbol scanner control do not cause the app to crash on non-Symbol devices. Simply use this updated SFX file instead of older ones, to prevent the crash problem.

Keywords: Symbol, PocketPC, barcode, scanner

BugID: SF-00019

KB ID: 10006

Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Problem with alternate-shape checkboxes in PocketPC targets created from Palm target

Problem: If you create a new PocketPC build target from a Palm build target, and the Palm target includes alternate-shape checkboxes or radio buttons, those controls are not migrated properly to the PocketPC target. The migrated alternate-shape controls become randomly sized and located on the PocketPC form.

Solution: This is a known problem [bug ID SF-00247] in Satellite Forms 6.1.x. There is no known solution at this time, so the options are to either:

- manually correct all affected controls after the PocketPC target has been created
- prevent the problem by temporarily unchecking the alternate-shape property for all of these controls in the Palm target before creating the PocketPC target, then re-enabling the alternate-shape property after the PocketPC target has been created

Status: RESOLVED This bug is resolved with Satellite Forms 8.

Keywords: alternate shape, checkbox, radio button, target, PocketPC, platform

BugID: SF-00247

KB ID: 10029

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

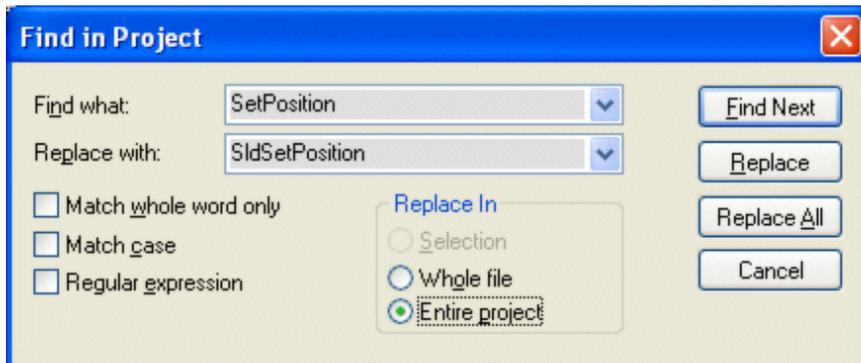
Error C016: Method 'Controls.SetPosition' takes 4 param(s) : 1 specified

Problem: Error C016: Method "Controls.SetPosition" takes 4 param(s) : 1 specified

Solution: This error message may appear when compiling an application using Satellite Forms 7, even if the same application compiled okay under Satellite Forms 6. The problem stems from a conflict between an extension that includes a SetPosition method and the new [SetPosition method for standard form controls](#) introduced with SatForms 7.

The Slider and Color Slider controls included with Satellite Forms include SetPosition and GetPosition control methods. We have modified the extension INF files for Slider and Color Slider to rename those methods with SatForms 7. They have been renamed SldSetPosition and SldGetPosition, and the SetMinMax function was renamed to SldSetMinMax as well. If your application uses the Slider or Color Slider controls, you will need to perform a find and replace operation on your script code, to use the updated extension function names. That will eliminate the compiler error as described above.

1. With your application opened in App Designer, go to the Scripts tab, open the Global section (by clicking on the + character), then select Variables. The script edit window should open. It does not matter if the global variables section is blank or not.
2. Select Edit | Replace from the menu, or press the CTRL-H hotkey, to bring up the Find & Replace dialog. In the Find What: field, enter SetPosition, and in the Replace With: field enter SldSetPosition. Click the Entire Project option in the Replace In selection, and then click on the Replace All button.



All instances of SetPosition in your script code will be replaced with SldSetPosition.

3. Repeat the process to replace GetPosition with SldGetPosition. Repeat also to replace SetMinMax with SldSetMinMax.

You should now be able to compile your application that uses the Slider or Color Slider control, without generating the compiler error complaining about the SetPosition taking a different number of parameters.

Attention LSGlobalList Control Users

The LSGlobalList custom control from PalmDataPro also includes a function named SetPosition, and will therefore generate the same error when you compile that application that uses LSGlobalList under Satellite Forms 7. To solve this problem you must modify both the extension INF file and update your script code. Perform the following steps to make the changes so that the application compiles under SatForms 7 with the LSGlobalList control.

1. Close App Designer. Modifications to extension INF files must be done while App Designer is closed, or it will not see the changes.
2. Locate the LSListBox.inf file in the \Satellite Forms 7\Extensions\{manufacturer} folder, for example \Satellite Forms 7\Extensions\PalmDataPro. Open the INF file in a text editor (such as Notepad).
3. Scroll down to the list of extension methods, and edit the name of the SetPosition function to be LSLSetPosition. Save and close the LSListBox.INF file.
4. Restart App Designer, and reload your application project.
5. Go to the Scripts tab, open the Global section (by clicking on the + character), then select Variables. The script edit window should open. It does not matter if the global variables section is blank or not.
6. Select Edit | Replace from the menu, or press the CTRL-H hotkey, to bring up the Find & Replace dialog. In the Find What: field, enter SetPosition, and in the Replace With: field enter LSLSetPosition. Click the Entire Project option in the Replace In selection, and then click on the Replace All button. All instances of SetPosition in your script code will be replaced with LSLSetPosition.

You should now be able to compile your application that uses the LSListBox control in Satellite Forms 7.

Note that if your application uses both the Slider/Color Slider control and the LSListBox control, you should use the Replace button in the Find & Replace dialog, instead of Replace All. That will enable you to search through your code and replace the Slider calls with SldSetPosition, and skip the LSListBox calls. You can then go back and run the replace operation again replacing the LSListBox SetPosition calls with LSLSetPosition. Note that with LSListBox V2.2 and higher the necessary changes have been made to the INF file.

Keywords: error, C016, SetPosition, GetPosition, Slider, LSListBox

KB ID: 10040

Updated: 2007-07-11

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

SocketScan PalmOS control does not work with laser SDIO scanner

Problem: SocketScan PalmOS control does not work with laser SDIO scanner

Solution: An updated version of the Satellite Forms SocketScan control for PalmOS is available that adds support for the Socket laser SDIO barcode scanner (SDSC 3M). No update is needed for the PocketPC SocketScan control.

Download the updated SocketScan extension for PalmOS from the Products & Support | Support Files | Software Updates & Patches section of the Satellite Forms website:

http://www.satelliteforms.net/cat16_1.htm

Unzip the SFE_SocketScan.prc extension file from the zip file that you downloaded. Place that prc file in the

\Satellite Forms 7\Extensions\Standard\Palm\
folder. Hotsync that updated prc file to your PalmOS devices to overwrite the older extension.

NOTE: If you use the Transmit Code ID feature of the Socket scanners that returns the symbology (barcode type) of the barcode that was scanned, you may need to modify your scanner initialization to ensure that you enable this feature. Do not rely on it being set properly just by resetting the scanner to default settings. Enable the Transmit Code ID feature explicitly by setting parameter #45, to the value 2 which returns the "Symbol ID code". Setting it to 0 will return no code ID char, and setting it to 1 will return an AIM ID string. Most developers want the Symbol ID Char if anything.

eg: error = SocketScan1.SetScanParam(45, 2, true|false)

To retain that setting across power cycles you need to set the Permanent flag to True. If you are setting this parameter every time the scanner is initialized (as recommended) then there is no need to set the Permanent flag to True, just leave it False.

Status: RESOLVED

Keywords: Socket, SocketScan, scanner, barcode, laser, SDIO, SDSC 3M

KB ID: 10046

Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Bug in Binarysearch function with PocketPC PDB

Problem: There is a bug (#SF-00328) in the SatForms 7.0.0.020 Binarysearch function with PocketPC PDB. The binarysearch function returns the row number of the matching record if a match is found, or the row number where a match should appear in the sorted table if a match was not found (this is called the sort position). The bug is that if the binarysearch function does not find a match, it returns the sort position row + 1. For example, if the correct sort position should be row 5, the PocketPC PDB runtime is currently returning an incorrect value of 5 + 1 = 6.

Solution: This bug applies to Satellite Forms 7.0.0.020 on the PocketPC platform using the PDB device database format. It does not affect SatForms 7.0 for PalmOS, nor does it affect the PocketPC platform when using the Microsoft CDB device database format.

We will have this bug corrected in the next maintenance release of Satellite Forms 7.

In the meantime, you may apply a workaround in your script code to handle this bug, in the following manner. What you can do is when you use the Binarysearch function, test if the search found a match or not. If it does find a match, the returned FoundRow will be correct. If it does not find a match, and your app is running on the Satellite Forms 7.0.0.020 runtime engine in your PocketPC PDB build target, subtract one from the FoundRow to account for the erroneous FoundRow value.

You can use a Private Global Function in the PocketPC PDB build target that returns True if the runtime engine version is 7.0.0.020 or less, or False otherwise. If you also have PalmOS and/or pocketPC CDB build targets, create a Private Global Function in each of those targets that always returns False. Let's name this function IsPPCbinSearchBug. Call this global function from your Binarysearch script to know whether you need to adjust the FoundRow value, like this:

```
dim bFound, iFoundRow
bFound = Tables().BinarySearch("Product", True, edFind, iFoundRow)
if (IsPPCbinSearchBug = true) and (bFound = false) then
    iFoundRow = iFoundRow -1    'fix PPCPDB binsearch bug
endif
```

Status: RESOLVED This problem has been resolved with the release of an updated SatForms PocketPC runtime engine with SF Patch 70002. If you have applied Patch 70002 to your system, or are using Satellite Forms 7.1 or higher, this problem is resolved.

Keywords: Binarysearch, PocketPC, PDB, Windows Mobile, WM5, bug,

BugID: SF-00328

KB ID: 10048

Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

PromptCustom may cut off some text on PocketPC

Problem: There is a bug in Satellite Forms 6.x and 7.0 for PocketPC in which the PromptCustom function may cut off some prompt text or button text. This problem does not affect the PalmOS platform.

Solution: A workaround for this problem is to pad some additional blank spaces to the end of your text, in order to cause the PromptCustom function to wrap the text down to the next line.

Status: RESOLVED This problem has been resolved with the release of an updated SatForms PocketPC runtime engine with SF Patch 70003. If you have applied Patch 70003 to your system, or are using Satellite Forms 7.1 or higher, this problem is resolved.

Keywords: prompt, promptcustom, msgbox, messagebox, dialog, PocketPC

BugID: SF-00278

KB ID: 10049

Updated: 2007-06-28

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

PocketPC target requires icon bitmap to compile even though it is not used

Problem: PocketPC target may require an icon bitmap to compile even though it is not used. If you create a PocketPC target from an existing PalmOS target, and you have the Create Launcher Application option enabled (checked) in the Project Properties of the PalmOS target, that Project Property is also checked in the PocketPC target, but is greyed out because it does not apply to that target. You cannot toggle that option off because it is greyed out, and an error message appears when you go to compile the target.

Solution: The workaround is to supply an icon bitmap file even though it is not used on the PocketPC target. Use the same PalmOS icon bitmap file from your original PalmOS build target. This enables the compiler to complete the build process without stopping to complain about a missing icon bitmap file. The compiler will not use this bitmap file, but will instead automatically use a Windows .ico icon file for your app if it is found in the images folder (it must be named the same as the .SFA project file).

Status: OPEN This bug has not been resolved yet.

Keywords: bitmap, icon, compile, project properties, create launcher icon

BugID: SF-00285

KB ID: 10050

Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Problems printing scripts in App Designer

Problem: There are some problems printing scripts in App Designer, from the File | Print menu options. Problems include:

1. App Designer continues to print scripts even when the Cancel button is clicked.
2. If you Print All Scripts to a file, it prints each script to the same filename, overwriting each time, instead of printing them all to one long file.
3. When you print a script, there is no indication on the printout as to which form and which event or control the script came from.

Solution: There are no worarounds or solutions to these printing problems at this time.

Status: OPEN This bug has not been resolved yet.

Keywords: print, print all

BugID: SF-00258

KB ID: 10051

Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

PocketPC app is relaunched after it is closed

Problem: If a SatForms PocketPC app is running, and the user switches away to another app without closing the SF app (for example, the user starts the File Explorer app), and then the user taps on the SF app icon again, the app is correctly brought back to the foreground, but it is automatically relaunched after it is closed.

Solution: There is no solution or workaround for this issue at this time.

Status: RESOLVED This problem has been resolved with the release of an updated SatForms PocketPC runtime engine with SF Patch 70003. If you have applied Patch 70003 to your system, or are using Satellite Forms 7.1 or higher, this problem is resolved.

Keywords: launch, icon, relaunch, restart, close

BugID: SF-00329

KB ID: 10052

Updated: 2007-06-28

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

An invisible 'hotspot' button is still clickable on PocketPC

Problem: An invisible "hotspot" button is still clickable on PocketPC. If you have a button control that is visible, but with no border and with no label text, a special type of "hotspot" button is used on the PocketPC. This hotspot button always returns False to the .visible property, but still responds to pen taps. A control should not respond to pen taps when it is not visible, and the hotspot button should properly support the .visible property. This problem does not affect the PalmOS platform.

Solution: There is no solution to this problem at the current time. A workaround may be to use the form's OnPenDown/OnPenUp scripts to either detect or ignore pen taps on the screen instead of using the hotspot button.

Status: OPEN This bug has not been resolved yet.

Keywords: button, visible, hotspot, click, pen tap, onpendown, onpenup

BugID: SF-00336

KB ID: 10053

Updated: 2007-06-28

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Ink control does not allow bitmap overlay on PocketPC

Problem: When an ink control is placed overtop of a bitmap image on PocketPC, the bitmap image is not visible, and cannot be used as a "template" for the ink control. On the PalmOS platform, the bitmap beneath the ink control will be displayed and can be used as a template. The ink control properly saves only the scribbles drawn by the user, and not the template image.

Solution: There is no solution to this problem at the current time.

Status: OPEN This bug has not been resolved yet.

Keywords: bitmap, ink, template, overlay, image, signature

BugID: SF-00341

KB ID: 10054

Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)

[Satellite Forms Website Home](#)

-0-

PocketPC CDB application cannot handle more than 127 fields per table

Problem: A Satellite Forms PocketPC CDB application cannot handle more than 127 fields per table. If more than 127 fields per table are used in a PocketPC CDB application, numerous record-oriented problems occur. The CDB database implementation cannot support more than 127 fields in a table. This problem does not occur with the PocketPC PDB database format, nor on the PalmOS platform.

Solution: There is no solution to this problem at the current time. You must ensure that you keep your CDB databases to no more than 127 fields per table.

Status: OPEN This bug has not been resolved yet.

UPDATE: CDB databases are not supported in Satellite Forms 8. Version 8 supports PDB handheld databases only.

Keywords: field, table, CDB, record, PocketPC

BugID: SF-00342

KB ID: 10055

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

PocketPC PDB listbox problems with more than 127 table fields

Problem: The PocketPC listbox has problems when used with tables that have more than 127 fields. While the PocketPC PDB database format can utilize up to 255 fields (columns, not records or rows) in a table, the listbox does not display the columns correctly when used on tables with more than 127 fields. This problem does not affect the PalmOS platform. PocketPC applications using the [CDB database format cannot handle more than 127 fields](#) per table, regardless of whether a listbox is used or not.

Solution: There is no solution to this problem at the current time. To avoid the listbox display problem, ensure that your PocketPC PDB database tables have no more than 127 table fields.

Status: RESOLVED This bug has been resolved in Satellite Forms 8.

Keywords: listbox, PDB, PocketPC, 127, field, database

BugID: SF-00342

KB ID: 10056

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Pen tap on Text or Lookup control generates 2 OnPenDown events on PocketPC

Problem: A pen tap on a Text or Lookup control generates 2 OnPenDown events on PocketPC, instead of a single event. This does not affect the PalmOS platform, nor does it affect other control types on PocketPC.

Solution: There is no solution to this problem at the current time. If you need to track the number of pen taps as a count, or otherwise want to trigger a pendown script only once for these unwanted double-tap events, you can use the following workaround:

1. Use the [PenTapInControl global script](#) to determine if the tap was within a text or lookup control, or otherwise.
2. If the tap was in a text or lookup control, toggle a global flag variable (eg. gFlagDoubleTap = not gFlagDoubleTap).
3. In the OnPenDown script, check the gFlagDoubleTap var and only take action when gFlagDoubleTap = false.
4. Set gFlagDoubleTap = false in AfterOpen. When the first pen down event fires, gFlagDoubleTap will be set True, and the script action will not fire because gFlagDoubleTap = True. When the second pen down event fires, gFlagDoubleTap gets set back to false, and the script action will proceed. Pen taps on other controls or on the form will not set gFlagDoubleTap = True, therefore they will execute the script on the first tap.

Status: OPEN This bug has not been resolved yet.

Keywords: OnPenDown, OnPenUp, pen, tap, double-tap, GetPenStatus, PenTapInControl

BugID: SF-00353

KB ID: 10057

Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Cannot set droplist caption in code on PocketPC

Problem: Cannot set a droplist caption via code on PocketPC, unless that caption is one of the items in the droplist control's list table. If the caption does not exist as an item in the list, the control caption is blank. On the PalmOS platform, any text can be used to set the droplist control caption, regardless of whether it is included in the list table or not.

Solution: There is no solution to this problem at the current time. Unfortunately, this problem does appear to be caused by platform differences in the way that droplist controls are implemented on the PocketPC platform compared to the PalmOS platform. The droplist control just does not support "unliked" captions on the PocketPC OS, whereas this is supported in the PalmOS platform. We have not yet found an acceptable way to overcome this PocketPC OS platform limitation in Satellite Forms.

Status: OPEN This bug has not been resolved yet.

Keywords: droplist, caption, platform, PocketPC

BugID: SF-00354

KB ID: 10058

Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Form Scrollbars do not Appear Automatically in PocketPC

Problem: Form scrollbars do not appear automatically in PocketPC applications, when the display orientation changes or when the application is run on a device with a small square screen. When an application designed for the standard portrait mode 240x320 QVGA PocketPC screen is run on a device with a small square 240x240 screen, or on a landscape 320x240 QVGA screen, a form scrollbar should automatically appear/disappear as needed. The vertical form scrollbar allows the user to scroll down to see the rest of the form, since it does not fit on the shorter screen.

Solution: The PocketPC ScreenSize extension can help with this problem by enabling you to query the display dimensions, and by firing an event when the display dimensions (or orientation) change. This allows you to dynamically adjust the controls on the form, using the GetPosition and SetPosition methods, in order to adapt the form layout to the current screen.

However, in many cases developers would rather just allow a scrollbar to appear on the form to handle these different display scenarios, allowing users with devices that have different display sizes to scroll down to see the rest of the form.

Status: RESOLVED This problem has been resolved with the release of an updated SatForms PocketPC runtime engine with SF Patch 70003. If you have applied Patch 70003 to your system, or are using Satellite Forms 7.1 or higher, this problem is resolved. With this updated version of the Satellite Forms runtime engine for PocketPC, form scrollbars automatically appear and disappear as needed depending on the display size.

Keywords: scrollbar, scroll, screen, dimensions, display, size, QVGA, square, landscape, portrait

BugID: SF-00356

KB ID: 10062

Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Form needs to be tapped by pen before accepting keyboard input on PocketPC

Problem: In certain circumstances on PocketPC devices, when a form opens it needs to be tapped by the pen before accepting keyboard input. If it is not tapped first, the keyboard input is ignored with a warning beep. This is a problem if your user interface relies on keyboard input, and not pen tap input. When the form is in this state, the keyboard input does not get to the OnKey event, as it is being ignored by the system before the InKey event gets a chance to handle it.

The circumstances that can lead to this problem include having a hidden edit control as the "topmost" control on the form, meaning the control that has the focus when the form opens.

Solution: A simple workaround to this problem is to make some other visible control, such as a text label, the topmost control on the form. Select the control in App Designer, right click to display the context menu, and select "Bring Control To Front". Recompile the application, and redeploy to the PocketPC device.

Status: RESOLVED This problem has been resolved with the release of an updated SatForms PocketPC runtime engine with SF Patch 70003. If you have applied Patch 70003 to your system, or are using Satellite Forms 7.1 or higher, this problem is resolved.

Keywords: keyboard, input, pen, OnKey, GetLastKey

BugID: SF-00351

KB ID: 10063

Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Deleting a record causes 'Error 30: Table won't open or invalid' on PocketPC PDB

Problem: Deleting a record causes "Error 30: Table won't open or invalid" on PocketPC using the PDB database format. The user can tap on the OK button in the error message prompt, and continue using the application from that point.

This problem can occur under certain circumstances, including script code that writes a value to a table other than the form's linked table just prior to the record being deleted. What happens is the form's current table record pointer can get "out of sync" when deleting a record, resulting in the Error 30 message. The application can continue from that error, and the table and form can be "re-synced".

Solution: This problem can be resolved by updating the Satellite Forms runtime engine for PocketPC to a newer version.

Status: RESOLVED This problem has been resolved with the release of an updated SatForms PocketPC runtime engine with SF Patch 70003. If you have applied Patch 70003 to your system, or are using Satellite Forms 7.1 or higher, this problem is resolved.

Keywords: error, invalid, delete, record, pointer, PocketPC

BugID: SF-00347

KB ID: 10064

Updated: 2007-07-11

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Janam XP20/XP30 scanner powers off but needs to be reset to turn back on

Problem: Janam XP20/XP30 barcode scanner device powers off but needs to be reset to turn back on. This problem can occur after scanning items in a Satellite Forms application on the Janam XP scanner, then powering off the device or allowing it to power off automatically. It will not turn back on, thus you need to remove the battery door and reset the device, either by pressing the reset button or by removing and reinserting the battery.

Solution: This problem is resolved by updating the ROM on the Janam XP device to the latest version. The updated ROM from Janam solves this barcode scanner library problem, and also corrects some other system issues. We recommend using the latest available ROM from Janam, which you can download from the ParterZone section of the Janam website at <http://www.janam.com>

Status: RESOLVED This problem is resolved with the release of updated Janam system ROMs.

Keywords: Janam, XP20, XP30, scanner, ROM, power, freeze, lockup

BugID: SF-00346

KB ID: 10065

Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Symbol MC50 scanner shows Error Enabling Scanner Library after scanning for a while

Problem: Symbol MC50 scanner shows Error Enabling Scanner Library after scanning for a while.

Solution: This problem is caused by a memory leak in the scanner/imager device driver in the Symbol device ROM. It is not a Satellite Forms error. The problem can be resolved by updating the MC50 ROM to a newer release that includes a fix for the device driver memory leak. Download the updated MC50 ROM from the Motorola/Symbol website at <http://www.symbol.com>

Status: RESOLVED This problem is resolved with the release of updated Symbol system ROMs.

Keywords: Symbol, MC50, scanner, imager, barcode, memory, library, error, PocketPC

BugID: SF-00344

KB ID: 10066
Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

PocketPC PDB problems filtering a record using a 10 digit numeric field

Problem: The Satellite Forms runtime engine for PocketPC using the PDB database format has a problem filtering record using a 10 digit or wider numeric field as the filter criteria. When using a large numeric field 10 digits or longer, the filter does not successfully select matching records to display. This results in incorrect application function.

Solution: A possible workaround is to reduce the width of the numeric field to be 9 or fewer digits wide (to avoid the problem), or to change the field type to a Character field instead of numeric (which would allow the longer numeric values to be used as filter criteria).

This problem is fully resolved in the updated Satellite Forms runtime engine for PocketPC included in the Satellite Forms Patch 70002.

Status: RESOLVED This problem has been resolved with the release of an updated SatForms PocketPC runtime engine with SF Patch 70002. If you have applied Patch 70002 to your system, or are using Satellite Forms 7.1 or higher, this problem is resolved.

Keywords: filter, numeric, 10 digit, PDB, PocketPC, Patch 70002

BugID: SF-00339

KB ID: 10067

Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Binarysearch function is not case sensitive on PocketPC PDB

Problem: The Tables().Binarysearch function is not case sensitive on PocketPC using the PDB database format, but it should be in order to be consistent with the PalmOS and PocketPC CDB Binarysearch behaviour.

Solution: This problem is fully resolved in the updated Satellite Forms runtime engine for PocketPC included in the Satellite Forms Patch 70002.

Status: RESOLVED This problem has been resolved with the release of an updated SatForms PocketPC runtime engine with SF Patch 70002. If you have applied Patch 70002 to your system, or are using Satellite Forms 7.1 or higher, this problem is resolved.

Keywords: binarysearch, binsearch, sort, case, sensitive, PocketPC, PDB

BugID: SF-00338

KB ID: 10068
Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Binarysearch function returns incorrect row number when no match on PocketPC PDB

Problem: The Tables().Binarysearch function returns an incorrect sort position row number when the search value is not found on PocketPC using the PDB database format. It returns a sort position row number (the location in the table where a match should be found based on the sort order) that is 1 higher than it should be.

Solution: This problem is fully resolved in the updated Satellite Forms runtime engine for PocketPC included in the Satellite Forms Patch 70002.

Status: RESOLVED This problem has been resolved with the release of an updated SatForms PocketPC runtime engine with SF Patch 70002. If you have applied Patch 70002 to your system, or are using Satellite Forms 7.1 or higher, this problem is resolved.

Keywords: binarysearch, sort, position, location, PocketPC, PDB

BugID: SF-00328

KB ID: 10069

Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

PocketPC RemoveFilter function does not remove filter on droplist table

Problem: PocketPC RemoveFilter function does not remove a filter on a droplist list table, if a filter is changed on a form that is not linked to a table, and a droplist on that form uses the filtered table as its list source. After removing the filter, the droplist does not display updated contents that reflect the active filter.

Solution: This problem is fully resolved in the updated Satellite Forms runtime engine for PocketPC included in the Satellite Forms Patch 70002.

Status: RESOLVED This problem has been resolved with the release of an updated SatForms PocketPC runtime engine with SF Patch 70002. If you have applied Patch 70002 to your system, or are using Satellite Forms 7.1 or higher, this problem is resolved.

Keywords: filter, removefilter, PocketPC, droplist

BugID: SF-00334

KB ID: 10070

Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

PocketPC project settings automatically change to MDB desktop DB format when project is loaded

Problem: PocketPC project settings automatically change to MDB desktop DB format when a project is loaded, regardless of what desktop database format the developer had selected when the project was saved. PocketPC PDB applicatino targets allow either DBF or MDB desktop DB formats in the Project Properties settings, but App Designer always changes the desktop DB format to MDB when the project is loaded.

Solution: The cause of this unwanted behaviour is due to a minor error in the PocketPC platform target definition files. The platform definition files are located in the \Satellite Forms 7\Platforms\ folder, with a .plt filename suffix. This problem can be corrected by manually editing the platform definition files to correct the minor error, as follows. The example below refers to the PocketPCPDB.plt platform definition file, but the same solution applies to all Windos Mobile/PocketPC platform files.

1. Load the \Satellite Forms 7\Platforms\PocketPCPDB.plt file into a plain text editor, such as Notepad.
2. Locate the following line:

```
SFDDDB = DDBMDB.dll
```

and comment out this line by prefixing a semicolon before it like this:

```
;SFDDDB = DDBMDB.dll
```

3. Restart App Designer, and reload your project, and it should now preserve the desired desktop database format instead of always defaulting to MDB.
4. Repeat this edit process for all of the Windows Mobile/PocketPC platform definition files.

Status: RESOLVED This problem is resolved with the release of Satellite Forms 7.1, or by manually editing the .plt files.

Keywords: MDB, DBF, project, properties, settings, desktop, database, PocketPC

BugID: SF-00327

KB ID: 10071

Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

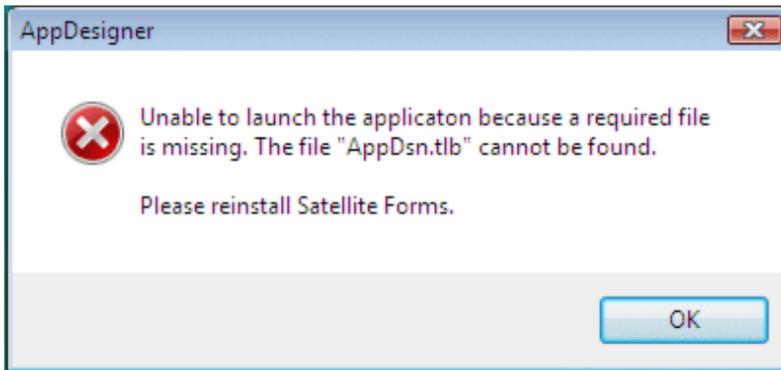
-0-

Windows Vista Compatibility Issues

Problem: Satellite Forms has some compatibility issues with Microsoft Windows Vista and Windows 7, outlined below.

A. Installation Issues

1. Administrator privileges are required to install Satellite Forms on Windows Vista.
2. When you install the Satellite Forms development package to a Windows Vista PC, then try to run the Satellite Forms MobileApp Designer (either from the desktop icon or from the Start menu or by launching an .sfa project file), an error message is displayed. The error message states "AppDesigner: Unable to launch the application because a required file is missing. The file "AppDsn.tlb" cannot be found. Please reinstall Satellite Forms."



To resolve this issue, right click on the Satellite Forms desktop icon, and select "Run as administrator". A "program needs your permission to continue" dialog should appear, MobileApp Designer should now start correctly. You can now close MobileApp Designer, and should now be able to launch it normally from the desktop icon, or from the Start menu, or by launching an .sfa file without using "Run as administrator".

3. If you install Satellite Forms into the Program Files folder, then try to compile one of the sample projects which are located in the \Program Files\Satellite Forms 7\Samples\Projects folder, you may be blocked from saving changes to the .sfa file. Vista may report that "access is denied". You can avoid this issue by installing Satellite Forms to the root of the C: drive instead of into Program Files, eg. C:\Satellite Forms 7. Disabling the Vista User Account Control may also resolve this issue. Starting with Satellite Forms 7.2, the default installation folder is changed from C:\Program Files\Satellite Forms 7 to C:\Satellite Forms 7 in order to avoid this issue.

B. Operation Issues

1. Windows Vista may prompt you for permission when running the RDKInst utility. Prior to Satellite Forms 7.2, it may also state that RDKInst.exe is an "unknown application". Starting with Satellite Forms 7.2, the RDKInst tool and other Satellite Forms EXE files are now signed with an authenticode certificate, so that the publisher is identified as Thacker Network Technologies Inc., avoiding the "unknown application" warning.
2. Using the RDKInst utility to install prc/pdb files, or using the InstallPrcToPalmPilot function of the SatForms Hotsync ActiveX control, or using the Hotsync user management functions of the ActiveX control on Windows Vista (or on Windows XP with palm Desktop 6.2 and Hotsync 7.0.2) would usually fail, because Satellite Forms could not obtain the list of Hotsync users.

The list of users shown by the RDKInst tool would be blank. This is resolved with Satellite Forms 7.2.

Status: RESOLVED Many of these issue are resolved with the release of Satellite Forms 7.2, or by following the instructions above.

Keywords: Windows Vista, Hotsync, RDKInst, ActiveX, authenticode, administrator, AppDsn.tlb

KB ID: 10083

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How-To Guides

How To use SatSync to send data to the Palm device

Problem: How To use SatSync to send data to the Palm device

Solution: SatSync is a sample application written in Visual Basic and provided in the \Samples folder of your SatForms installation. SatSync demonstrates how to use the SatForms HotSync control to send and receive data between the desktop PC and PalmOS handheld.

These instructions assume that you have created and compiled your application in App Designer, and want to send data from an existing database table on the PC (with the same schema as the table in App Designer) to the handheld. It is also possible to bring this data back into the table editor in App Designer with an additional step. If your data already exists in the App Designer table editor, there is no need to follow these steps, since you can just use the Handheld | Download App & Tables function. These instructions are for sending data from an existing PC database to your SatForms application on the handheld, and optionally bringing that data back into the App Designer table editor.

1. Start the SatSync sample application in \Samples\SatSync.
2. Select File | New, then File | Configure Send List.
3. Select the DBF or MDB files to send data to the handheld.
4. Save your SatSync config file, then click on Send Tables.
5. Hotsync, and the data should get sent to the handheld, where it now resides in the handheld tables.

[Optional step to bring data back into App Designer Table Editor]

6. In App Designer, select Handheld | Upload Tables and perform another sync, bringing the data back into the table Editor.

Keywords: SatSync, PalmOS, table, DBF, MDB, database, HotSync

See Also: [How To use SatSyncPPC to send data to the PocketPC device](#)

KB ID: 10007

Updated: 2006-10-12

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To use different platform targets for PocketPC applications

Problem: How To use different platform targets for PocketPC applications.

Q: There are several PocketPC platform targets available in Satellite Forms 6.1, including PocketPC 2002, pocketPC 2003, WinMobile 5, WinMobile5 Square, and WinCE.NET. Which target(s) should I include in my application?

This article is updated December 3, 2007 with new recommendations due to improvements in Satellite Forms 7.1. Older information is archived below for previous versions of Satellite Forms.

Solution: In many cases, you need only include a single PocketPC target in your project to create an application that will run on many different PocketPC target devices. This is the case whether you are upgrading an application created with an earlier version of Satellite Forms, or are creating a new application from scratch. The following guidelines apply:

1. Generally speaking, you can handle most PocketPC devices with a single PocketPC target. This target will use the Palm DB (PDB) device database format by default. DO NOT change it to use the obsolete Microsoft Pocket PC DB (CDB) database format.

You may use the PocketPC target for use on PocketPC 2002, PocketPC 2003, 2003SE, WinMobile 5, and WinMobile 6 Classic and Professional devices. You do not need to create separate targets for all of those platforms. The device application and database files created for the PocketPC target can be used on all of those devices.

2. Creating new applications

For new applications, you should use the PocketPC target. Most PocketPC target platforms are really not different except for the name, and are kept solely to maintain compatibility with projects written in an older version. There is no real difference between these targets, aside from the name. There is no need to add more than one of these targets to your application, since they are all compatible with each other.

3. WinMobile 5 Square target

The WinMobile 5 Square target is designed for the Windows Mobile devices that use a square 240x240 screen instead of the regular PPC 240x320 screen, like the Palm Windows Mobile Treo 700W/WX.

If you have a device that requires a different screen dimension (for example a device with a 320x320 square screen), contact [Satellite Forms Support](#) for help in building a specific target support file for that device.

The older KB article information applicable to Satellite Forms 6.x is archived below.

1. Generally speaking, you can handle most PocketPC devices with a single PocketPC 2003 target.

2. Supporting PocketPC 2002 devices

The only PocketPC devices that require a dedicated target are PocketPC 2002 devices. If you want to support PocketPC 2002 devices with your application, you will need to build a separate PocketPC 2002 target for your application. The device database (CDB) files for this target are not compatible with the CDB files created for other PocketPC targets. If you do not need to support PocketPC 2002 devices with your application, ignore this target altogether.

3. Upgrading an application created with an earlier version of Satellite Forms

You may continue to use the PocketPC2003 target for use on PocketPC 2003, 2003SE, WinMobile 5 and WinCE.NET devices. You do not need to create separate targets for all of those platforms. The device application and database files created for the PocketPC 2003 target can be used on all of those devices (but not on PocketPC 2002 devices).

4. Creating new applications

For new applications, you could consider using the PocketPC 2003 target or the WinMobile 5 target. They are really not different except for the name. WinMobile 5 targets will work on PocketPC 2003 devices as well. The WinCE.NET target is also the same: it is included for developers writing new WinCE.NET applications so they can select that target instead of asking "why do I use the PPC 2003 target for WinCE.NET?". There is no real difference between these targets, aside from the name. There is no need to add more than one of these targets to your application, since they are all compatible with each other.

5. WinMobile 5 Square target

The WinMobile 5 Square target is designed for the forthcoming Palm Windows Mobile Treo, which is expected to use a square 240x240 screen instead of the regular PPC 240x320 screen. If you have a device that requires a different screen dimension (for example a WinCE.NET device with a 320x320 square screen), contact [Satellite Forms Support](#) for help in building a specific target support file for that device.

Keywords: target, platform, PocketPC, 2002, 2003, WinMobile 5, WinCE.NET

KB ID: 10024

Updated: 2007-12-03

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To use Global Functions & Subs to replace extension functions not available on the current target platforms

Problem: How To use Global Functions & Subs to replace extension functions not available on the current target platforms

Solution: You can have an application that uses an extension to perform a function on one platform, but uses a global function or subroutine to perform the same function on another platform where that same extension might not be available. You can use this method to prevent the compiler from complaining that the function does not exist on the current target platform.

This is easily handled with Global Subs and Functions, specifically the Private (not Shared) global funcs and subs. Simply create a function or subroutine matching the name and parameters of the extension function which you do not have access to under that platform. The sub does not have to do anything, though functions need to return a value.

This is very useful when porting apps between platforms. For example, here is a global sub to handle the LockScreen and UnlockScreen functions of the PalmDataPro SFScreenLock extension that is currently only available for PalmOS. Place this code in your PocketPC target global functions and subroutines (Private) script section:

```
'global pseudo subroutines to replace extension calls
```

```
Sub LockScreen  
'do nothing on PocketPC  
End Sub
```

```
Sub UnlockScreen  
'do nothing on PocketPC  
End Sub
```

Now your existing PalmOS target code that uses the LockScreen/UnlockScreen functions can be compiled for PocketPC without compiler errors.

You can also use this technique to work out differences between PalmOS and PocketPC implementations of similar extensions, like the PalmOS Internet extension and the PocketPC Winsock extension. While they share several common functions and design, they each have different platform-specific network setup and teardown functions. As shown in the SatForms 6.1 cross platform SMTP and TCPIP Socket sample applications, we can use global functions to write common network code that runs on either platform.

In the older PalmOS-only SMTP sample, this code was used to initialize the network using the OpenNetLib extension function:

```
' open the network connection  
Rec_data = "connecting..."  
if ( OpenNetLib() <> 0 ) then  
  MsgBox("Error opening NetLib - " & GetLastError())  
endif
```

In the new cross platform SMTP sample included with SatForms 6.1, this form level script is changed to this:

```
' open the network connection  
'calls a private (per target) global script OpenNetwork to open network taking into
```

```
'account the different startup functions for PalmOS and PocketPC targets
Rec_data = "connecting..."
if ( OpenNetWork() <> 0 ) then
  MsgBox("Error opening network - " & GetLastError())
endif
```

The above code is the same for the PalmOS and PocketPC targets.

Then, in the Global Funcs & Subs (Private) script for PalmOS, we have this code:

```
'script function to open the network
'in the PalmOS implementation this calls OpenNetLib() extension function
Function OpenNetwork()
  OpenNetwork = OpenNetLib()
End Function
```

```
'in the PalmOS implementation this calls CloseNetLib() extension function
Function CloseNetwork( closemode )
  CloseNetwork = CloseNetLib(closemode)
End Function
```

In the Global Funcs & Subs (Private) script for the PocketPC target, we have this code:

```
'script function to open the network
'in the PocketPC implementation this calls WSAStartup() extension function
Function OpenNetwork()
  OpenNetwork = WSAStartup()
End Function
```

```
'in the PocketPC implementation this calls WSACleanup() extension procedure
'the close mode is ignored
Function CloseNetwork( closemode )
  WSACleanup()
  CloseNetwork = 0 'always return 0 on PPC
End Function
```

That's it -- all of the rest of the SMTP sample network code is the same for both the PalmOS and PPC targets, using the Internet and Winsock extensions.

Keywords: extension, target, platform, compiler, global, function, sub

KB ID: 10016
Updated: 2006-10-02

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Install the SatForms Runtime for PocketPC Programmatically

Problem: How To Install the SatForms Runtime for PocketPC Programmatically

Updated for Version 8: With Satellite Forms 8 there is a new capability to use an integrated runtime engine, which removes the need to install the runtime engine files separately from the application files.

Updated: Starting with Satellite Forms 7.0, it is possible to bundle the PocketPC runtime engine directly with your application files, instead of needing to install the runtime engine separately. That is the new recommended approach, described in [How To Bundle the SatForms PocketPC runtime engine with your app](#). The information below is retained for Satellite Forms 6.x users.

Solution: The installation of the SatForms Runtime for PocketPC can be automated using the SatForms ActiveSync OCX. An example is given here for Visual Basic, though the same functions could be executed under any development language that employs ActiveX controls. The necessary steps are:

1. Install the SF runtime cab file to the PPC device
2. Remotely launch the cab file to install the files on the PPC

1. Use FileSendToPPC function to send the right CAB file to a folder on the PPC, eg. \MyAppInstall

Use the FileSendToPPC OCX function to send the CAB file to the PPC in the desired folder (may need to use the CreateFolder function first if the folder does not exist). It is safe to send the CAB to the default \My Documents folder if desired.

eg. SyncAx1.FileSendToPPC("SatFormsRuntimeRDK.Arm.CAB",
"\MyAppInstall\SatFormsRuntimeRDK.Arm.CAB")

2. Use the StartApp function in CeSync to invoke the CAB file to install. To do this you need to start the wceload.exe utility on the PPC and pass it the name of the CAB file to install:

```
SyncAx1.StartApp("wceload.exe", "\MyAppInstall\SatFormsRuntimeRDK.Arm.CAB")
```

NOTE: If you install stuff to a path with spaces (eg. \My Documents\MyApp) then you will have to deal with enclosing the commandline arg in quotes, which in VB ends up looking like this:

```
SyncAx1.StartApp("wceload.exe", """"\My  
Documents\MyApp\SatFormsRuntimeRDK.Arm.CAB""")
```

The SatForms Runtime installation should proceed without needing any user input. On PocketPC 2002/2003/2003SE the operating system will automatically delete the CAB file once it has been installed, but on Windows Mobile 5.0 the CAB will not be deleted automatically by the system.

Keywords: PocketPC, runtime, install, CAB, wceload, StartApp, OCX, ActiveSync

KB ID: 10015

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)

[Satellite Forms Website Home](#)

-0-

How To make Satellite Forms 6.1 PocketPC applications launch faster

Problem: How To make Satellite Forms 6.1 PocketPC applications launch faster

NOTE: This article applies to Satellite Forms PocketPC applications that use the CDB handheld database format ONLY. The techniques herein do NOT apply to SatForms PocketPC application using the PDB database format, which is highly recommended over the CDB format, for numerous reasons. If you are using Satellite Forms 7.0 or higher, the best approach is to [use the PDB database format for your PocketPC applications](#).

Update for Version 8: Satellite Forms 8 no longer supports the CDB handheld database format discussed in this article. This article does NOT apply to Satellite Forms 8.

Solution: The Satellite Forms runtime engine for PocketPC needs to have have an extra record index column at the end of each table in the application. Previous versions of the SatForms runtime engine automatically created this special record index column the first time an application was launched on the handheld, which resulted in longer load times than subsequent launches of the program. This additional load time was more noticeable with larger databases (larger numbers of records). Starting with Satellite Forms 6.1, it is now possible to create this record index right in the table editor in App Designer, resulting in a quicker initial launch of the application.

With previous SatForms runtime engine versions, a numeric column named SYS_POSITION was automatically added to the end of each table on the PocketPC when an application was started for the first time. This SYS_POSITION table was needed by the runtime engine to maintain proper sorting of the data on the handheld. The creation of this additional field and filling it with correct record position data resulted in the application taking longer to load the first time it was launched, and also affected the corresponding desktop database if one was used to synchronize the data with the handheld, since it too would need this extra column. This would lead to a problem if a cross platform PalmOS and PocketPC application wanted to sync to the same desktop database tables, because this SYS_POSITION column would not exist in the database created for the Palm target.

The SatForms App Designer would ignore this SYS_POSITION column when bringing data from the handheld back into the table editor using the Upload Tables function.

With SatForms 6.1, the name of this special column has been changed to SYS_POSIDX, and it is now possible to create this special record index column in the App Designer table editor, just like any other column in your database. This change offers two benefits:

- The application will launch faster the first time on the handheld
- The desktop database can now be the same for both PalmOS and PocketPC targets in a cross platform application

Adding the SYS_POSIDX column to your tables in App Designer is optional, but recommended for applications with larger databases or cross platform applications that need to sync both platforms to the same desktop database. If you do not add this column to a table, it will still be automatically created by the runtime engine the first time that table is opened, as in previous versions.

There are several requirements to follow in order to create the SYS_POSIDX column properly:

- The SYS_POSIDX column must be the last column in your table(s).

- You must use the SatForms 6.1.1 or higher runtime engine for PocketPC. This method will NOT work with earlier versions.
- The SYS_POSIDX column must be a Numeric type, 8 digits, no decimals.
- The SYS_POSIDX column must be filled with correct index values for each record in the table. The first record must have a value of 1, and each subsequent record must be incremented by 1.
- The SYS_POSIDX column must not be accessed or modified by your handheld application in any way. The runtime engine will maintain it automatically when records are created/modified/deleted. Attempting to modify it's contents in the handheld application will resulted in unpredictable application behaviour.
- The data in the SYS_POSIDX column in the desktop database, including in the App Designer table editor, must be maintained by you to ensure that it contains the correct sequential values.
- We recommend that if you create the SYS_POSIDX column in any of your tables, you should consider creating the column in all of your tables, as the consistent table design will make ongoing maintenance and future development easier.

If you take care to follow all of these requirements and add the SYS_POSIDX column to your tables in App Designer, you will find a significant speed benefit the first time your application is launched on the handheld. The larger your application (number of tables and number of records), the more noticeable the speed improvement will be. Subsequent launches of the application will be faster regardless of whether or not you include the SYS_POSIDX column in your tables, since the runtime engine would have automatically created it in each table.

TIP: Using your desktop database application to maintain the SYS_POSIDX contents

You can create a macro in your desktop database application to fill the SYS_POSIDX field of each record with the record number (RECNO), prior to sending it to the PocketPC handheld. This ensures the SYS_POSIDX column is always prepared properly before being sent to the handheld.

Keywords: PocketPC, launch, speed, performance, database, index, data, SYS_POSITION, SYS_POSIDX

KB ID: 10009
Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To use color bitmaps in your application

Problem: How To use color bitmaps in your application

Solution: Color bitmaps are possible if you follow the required image naming convention listed below.

Satellite Forms has supported color bitmaps since V4.1. Unfortunately the documentation for this support has been difficult to come by. Here is the relevant information for using color bitmaps in your SatForms application.

2.1 Color Support

=====

You can now embed color images into Satellite Forms applications. App Designer can read standard Windows BMP files and convert them into Palm bitmap resources.

To achieve great looking graphics on devices with different color capabilities, you can provide multiple bit-depth versions of any image. Satellite Forms runtime engine will use the image that is most suitable for the device.

2.1.1 Specifying Multiple Bit-depth Images

- Create or open an existing form with App Designer
- Insert a Bitmap Control onto the form
- Double-click on the new bitmap control
- Browse or type the location of a black and white BMP file
- Close the control's property dialog box

Satellite Forms supports 5 bit depths: 1-bit, 2-bit, 4-bit, 8-bit and 16-bit. However, you must always supply a 1-bit (black and white) version of the image. This will ensure that your application will always look right on black and white Palm devices.

Once you have associated a black and white image with a bitmap control, App Designer will use a predefined file naming convention to locate the other versions of the image file.

App Designer will use the suffix "-2", "-4", "-8" and "-16" to locate the color versions of the black and white image. The number corresponds to the bit-depth of the file.

For example: if the bitmap control is associated with "Photo.bmp", App Designer will look for "Photo-2.bmp", "Photo-4.bmp", etc.

You don't have to specify images for all bit-depths. Sometimes the lower bit images are good enough. Higher bit images occupy a larger memory footprint.

When building the project files, App Designer will use a dithering algorithm to transform 2, 4, 8 and 16-bit images so that they will use the correct Palm's color palettes, so your images may look different

on the device. To avoid this transformation effect, you may want to use a graphic editing tool to make sure that the images are already using the correct Palm color palettes. For your reference, Satellite Forms installs "PalmPalettes.bmp" in the Templates directory.

Bitmap images used by bitmap controls in a Satellite Forms project will end up as resources in a new target file. This new file uses the following naming convention: ExxxxMMnn#AppName.prc, where xxxx denotes the creator ID of your application, MM and nn are major and minor versions of your application.

You can use Microsoft Paint to create your images. You can usually find this program in \Program Files\Accessories\.

Note: using Microsoft Paint, 1-bit BMP file must be saved as Monochrome Bitmap and other BMP files must be saved as 24-bit Bitmap.

See the about box in any of Satellite Forms sample projects for examples.

Note: There is a limit of 64K for each bitmap family (all bit depths combined).

2.1.2 Using Bitmaps in Tables

- Create, import or open an existing table
- Add a new "BINARY" column
- Create or open an existing form
- Double-click the form
- Link the table to the form
- Close the form's property dialog
- Insert a Bitmap Control onto the form
- Double-click on the new bitmap control
- Change the "Image Source" from File to Table
- Choose the "BINARY" column
- Close the control's property dialog box

Satellite Forms' runtime engine will use the content of the current record to render the image. Only binaries of type "Tbmp" (Palm bitmap) are currently supported by Satellite Forms runtime engine. Future third party extensions may support other binary types.

See the "Deliveries" sample project to see how this is done.

Note: A Palm record can only be 64K in size. This limitation directly impacts the size of your binary fields.

2.1.3 Populating a BINARY column in a Table with a Palm bitmap family

A bitmap control can be linked to a binary column in a table. Satellite Forms engine will then load the "Tbmp" (Palm bitmap) object contained in the record and render it on the screen.

For this process to work, you will need to fill in the linked-column with the proper "Tbmp" objects in advance.

Satellite Forms 4.1 has a new feature that can assist in putting Palm bitmaps into binary columns.

When Satellite Forms translates tables from desktop databases (DBF/MDB) to Palm databases (PDB), it will automatically convert embedded pathnames in the binary columns into binary objects. Translation happens when App Designer builds your project or when Satellite Forms conduit runs.

Pathnames must be embedded in binary columns using the following formats:

HSBM<full pathname>
- or -
HSRW<full pathname>

HSBM and HSRW denote the type of file that follows.
<full pathname> denotes the location of the file on your desktop PC.

If HSBM is specified, the pathname must point to a black and white BMP file. Satellite Forms uses the file naming convention discussed in section 2.1.1 to discover the color BMP files.

If HSRW is specified, the pathname can point to any file. During translation, Satellite Forms will simply embed the content of the file into the column without any modifications. Note: the current built-in bitmap control can only recognize Palm bitmap. However, third party extensions may use this same mechanism to move other data types onto the device. E.g. you can embed MP3 files to be used by a future MP3 extension.

Note: On Palm devices, a record is limited to 64K, so embedding large files will fail.

Examples:

HSBMC:\My Documents\Vacation\Photo.bmp
HSRWC:\Music\Tune.mp3

See the "Deliveries" sample project to see how this is done.

The "Deliveries" sample also demonstrates how to embed pathnames using scripts.

Note: When assembling a record, Satellite Forms makes sure that a record will not exceed 64K in size. Satellite Forms will omit binary contents that will cause the record to grow beyond 64K.

2.1.4 Populating a BINARY column in a Table with a Resource Locator

Satellite Forms can also render Palm bitmaps that already exist on existing Palm databases.

In this case, you need to embed a resource locator. A resource locator uses the following format:

SFRLRSRC: <PRC name>:Tbmp: <id>

where,

<PRC name> is the name of the resource database (PRC)

<id> is the resource ID of the Palm bitmap family

Example:

SFRLRSRC: MyLogoResDb: Tbmp: 1001

2.1.5 Using the App Designer Table Editor to Populate BINARY Columns

You can enter strings with the format specified in section 2.1.3 and 2.1.4 using the built-in table editor in App Designer. When the project is built, App Designer will convert any valid entry to the proper Palm bitmap family.

Note: App Designer will not embed bitmaps that would cause a record to exceed 64K.

Warning: Using "Upload Tables..." feature in App Designer will clear the contents of these binary columns. You will need to reenter the strings again. You could consider adding a CHAR column to your table to save the pathname strings, in addition to entering them into the BINARY column, thus making it easier to repopulate the contents of the BINARY column if you use the Upload Tables function, by copying and pasting the pathname strings from the CHAR column to the BINARY column in the table editor.

Keywords: bitmap, color, image, graphic, logo

KB ID: 10010

Updated: 2005-11-17

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To use SatSyncPPC to send data to the PocketPC device

Problem: How To use SatSyncPPC to send data to the PocketPC device

Solution: SatSyncPPC is a sample application written in Visual Basic and provided in the \Samples folder of your SatForms installation. SatSync demonstrates how to use the SatForms ActiveSync control to send and receive data between the desktop PC and PocketPC handheld.

These instructions assume that you have created and compiled your application in App Designer, and want to send data from an existing database table on the PC (with the same schema as the table in App Designer) to the handheld. It is also possible to bring this data back into the table editor in App Designer with an additional step. If your data already exists in the App Designer table editor, there is no need to follow these steps, since you can just use the Handheld | Download App & Tables function. These instructions are for sending data from an existing PC database to your SatForms application on the handheld, and optionally bringing that data back into the App Designer table editor.

1. Start the SatSyncPPC sample application in \Samples\SatSyncPPC.
2. Select File | New, then File | Configure Send List.
3. Select the DBF or MDB files to send data to the handheld. These are your existing data tables, not the tables generated by App Designer when it builds your project. Enter the folder on the handheld where you wish to store the tables (eg. \My Documents\Work Order).
4. Save your SatSyncPPC config file, then click the Download Tables checkbox.
5. Attach your PocketPC device, and the data should get sent to the handheld, where it now resides in the handheld tables.

[Optional step to bring data back into App Designer Table Editor]

6. In App Designer, select Handheld | Upload Tables, bringing the data back into the table Editor.

Keywords: SatSync, SatSyncPPC, PocketPC, table, DBF, MDB, database, ActiveSync

See Also: [How To use SatSync to send data to the Palm device](#)

KB ID: 10008

Updated: 2006-10-12

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To use the Ink View OCX to display uploaded signatures on the PC

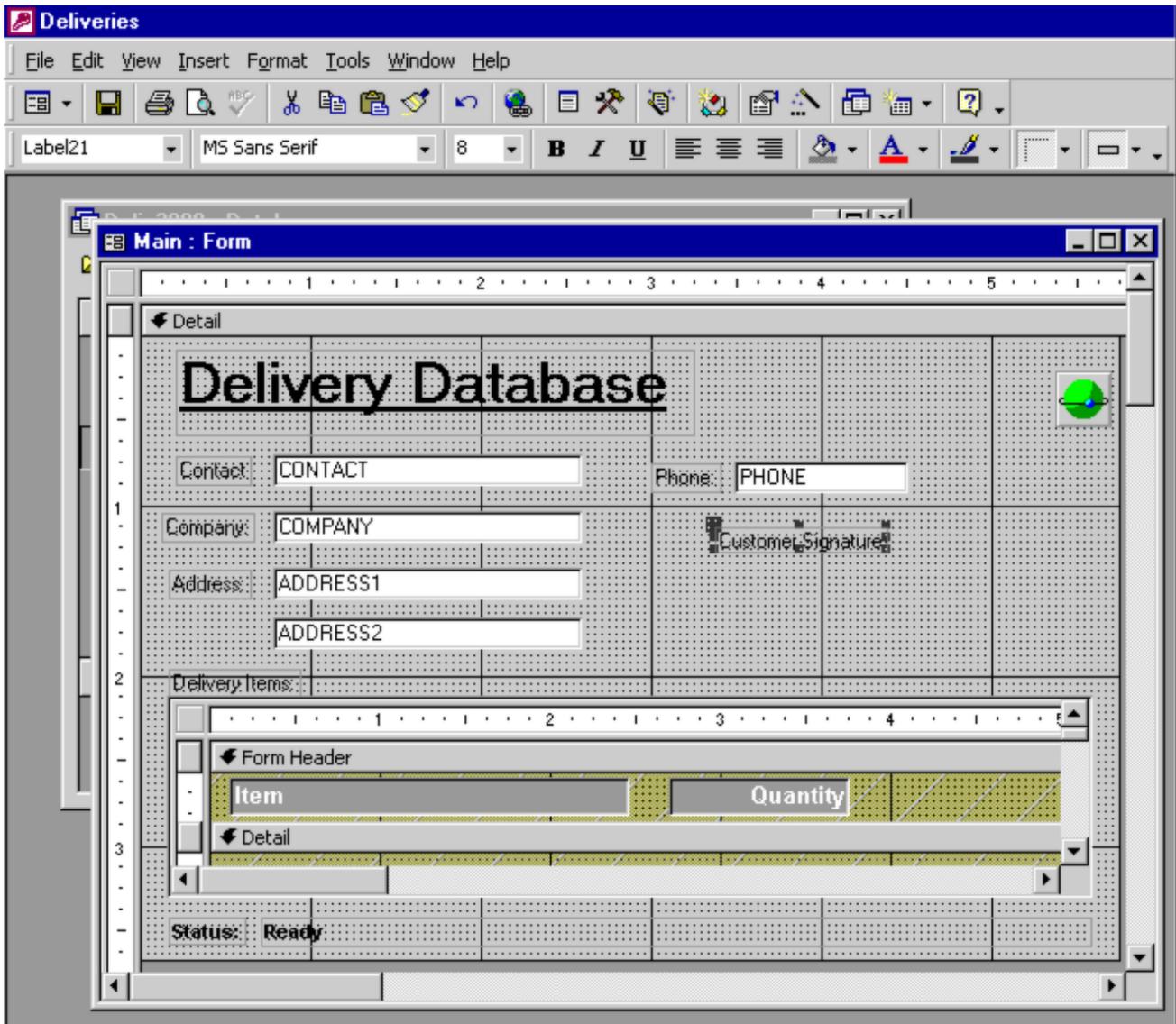
Problem: How To use the SatForms Ink View OCX to display uploaded signatures on the desktop PC

Solution: There is an ActiveX control included with Satellite Forms that is used to display the ink images captured in your handheld application on your desktop PC. This control is called the Ink View ActiveX Control (OCX), and is designed to display ink contents (such as signatures) captured on either PalmOS or PocketPC handhelds, and uploaded back to the desktop.

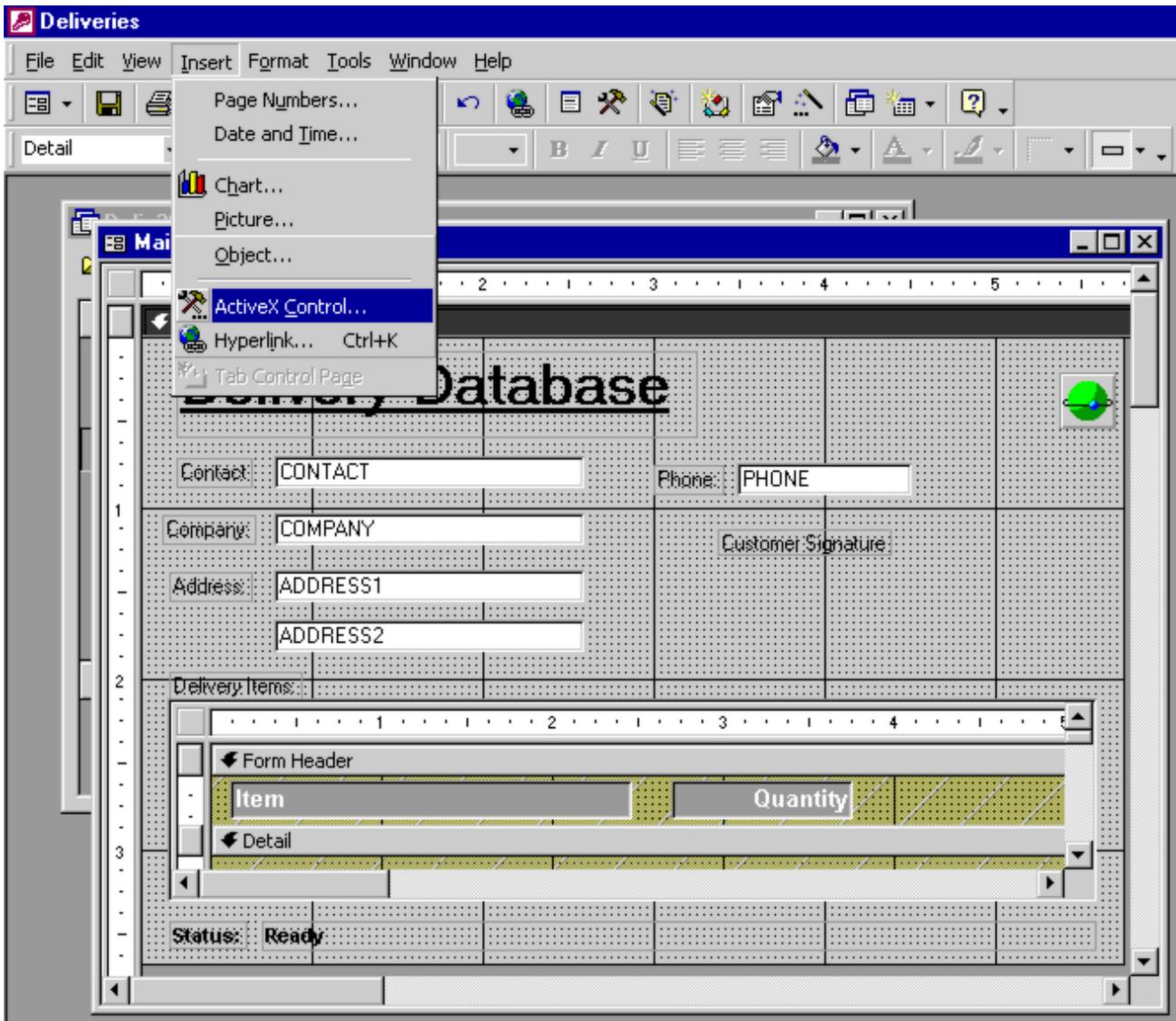
The instructions for using this control were omitted from earlier Satellite Forms documentation.

Below is a step-by-step instruction on how to insert and set up the Satellite Forms Ink View ActiveX control. The example used here is using the Access 2000 "Deliveries" sample found in the Satellite Forms directory under "Projects\Sample Projects\Deliveries\Access 2000\Access". Although this example uses Access 2000(R) the steps taken here are applicable to almost any ActiveX form.

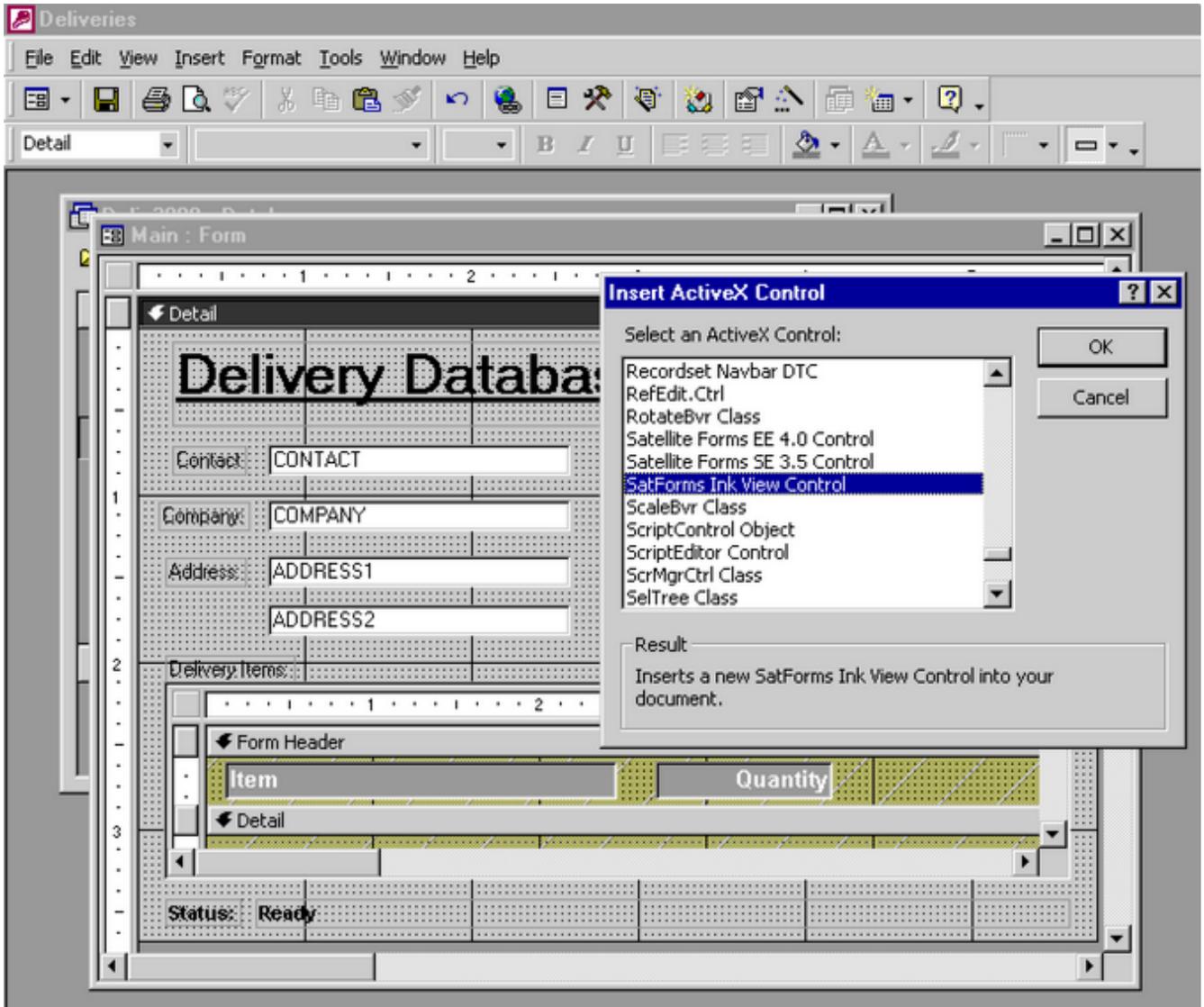
Step 1 - Open the desired form in the "Design" mode and decide where the Ink View control should be located.



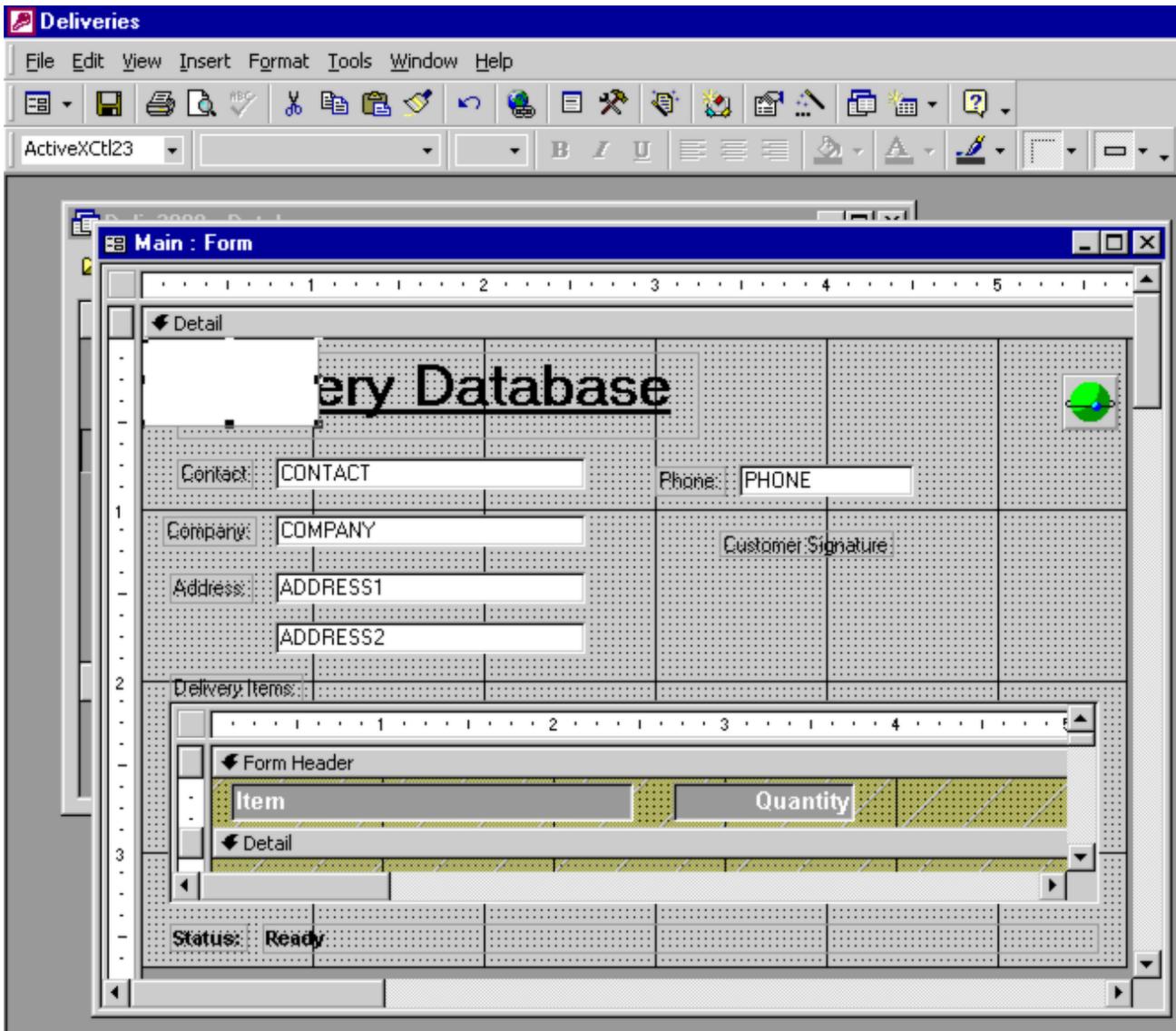
Step 2 - Go to "Insert" and select "ActiveX Control".



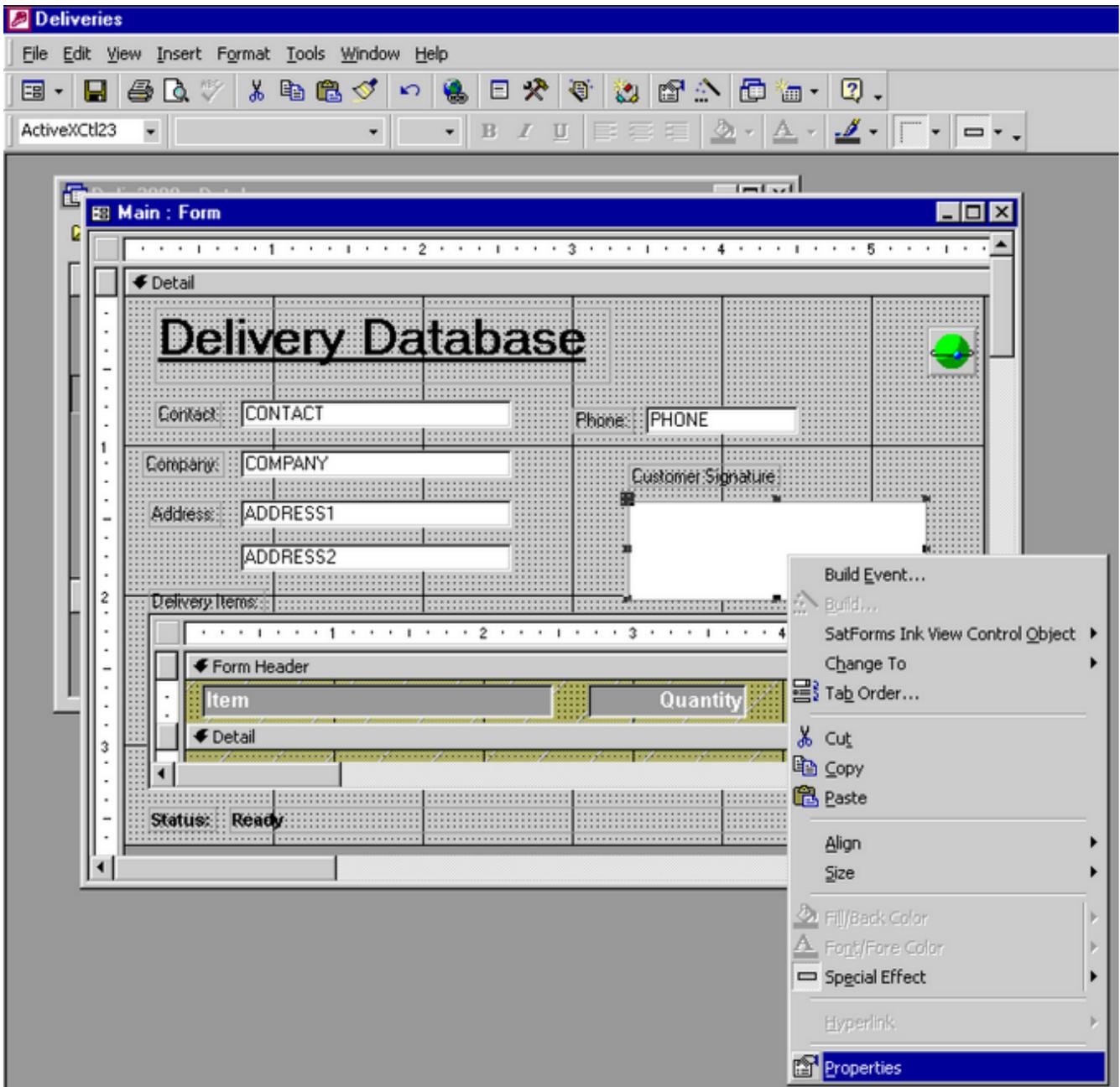
Step 3 - Select "SatForms Ink View Control" in the "Insert ActiveX Control" dialog box.



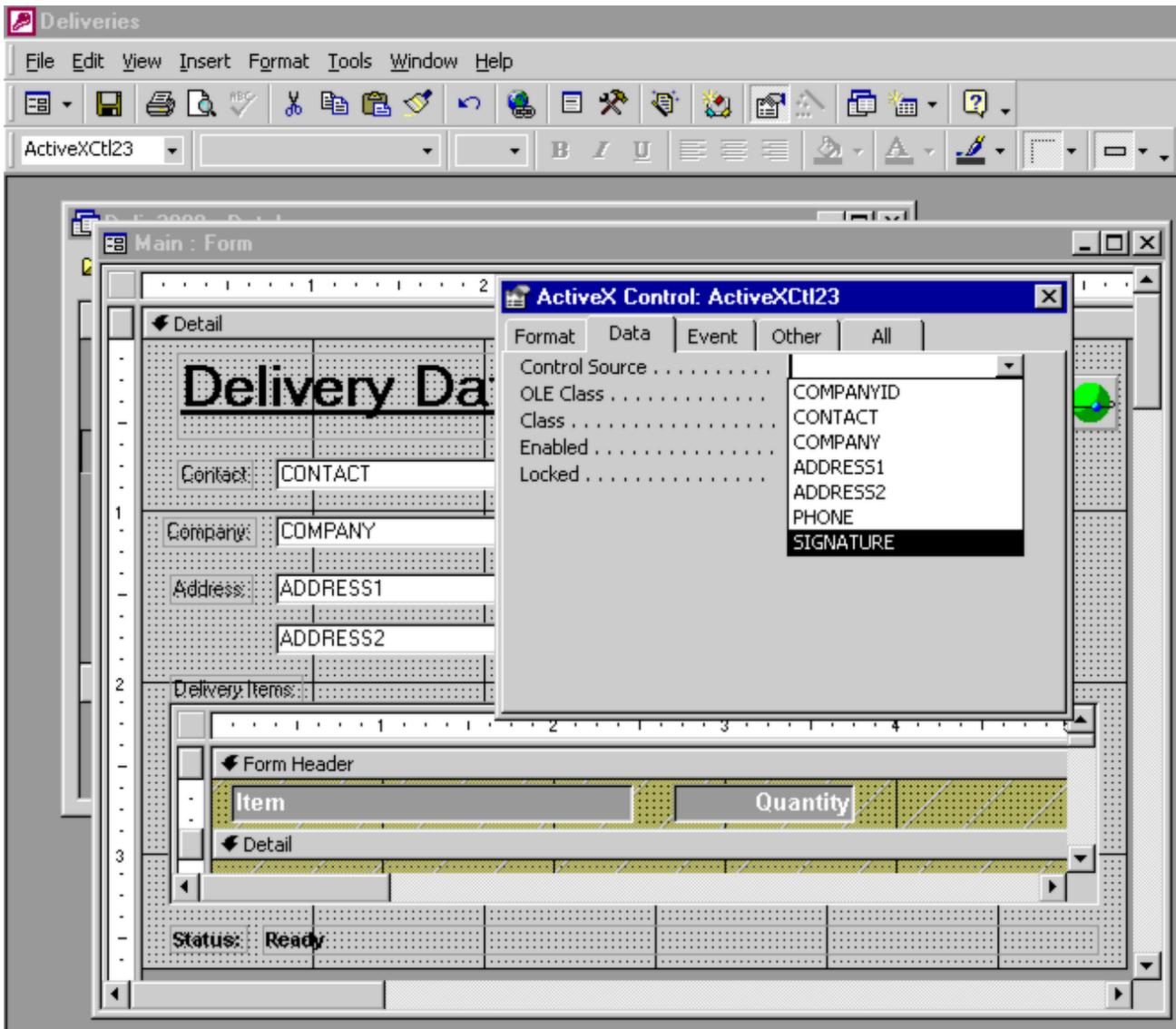
Step 4 - The Ink View Control will now appear as a white box on the form. The entered ink control data will be displayed in this box exactly as it appears while in the Satellite Forms Ink Control. Move this box to the appropriate space designated by the title bar.



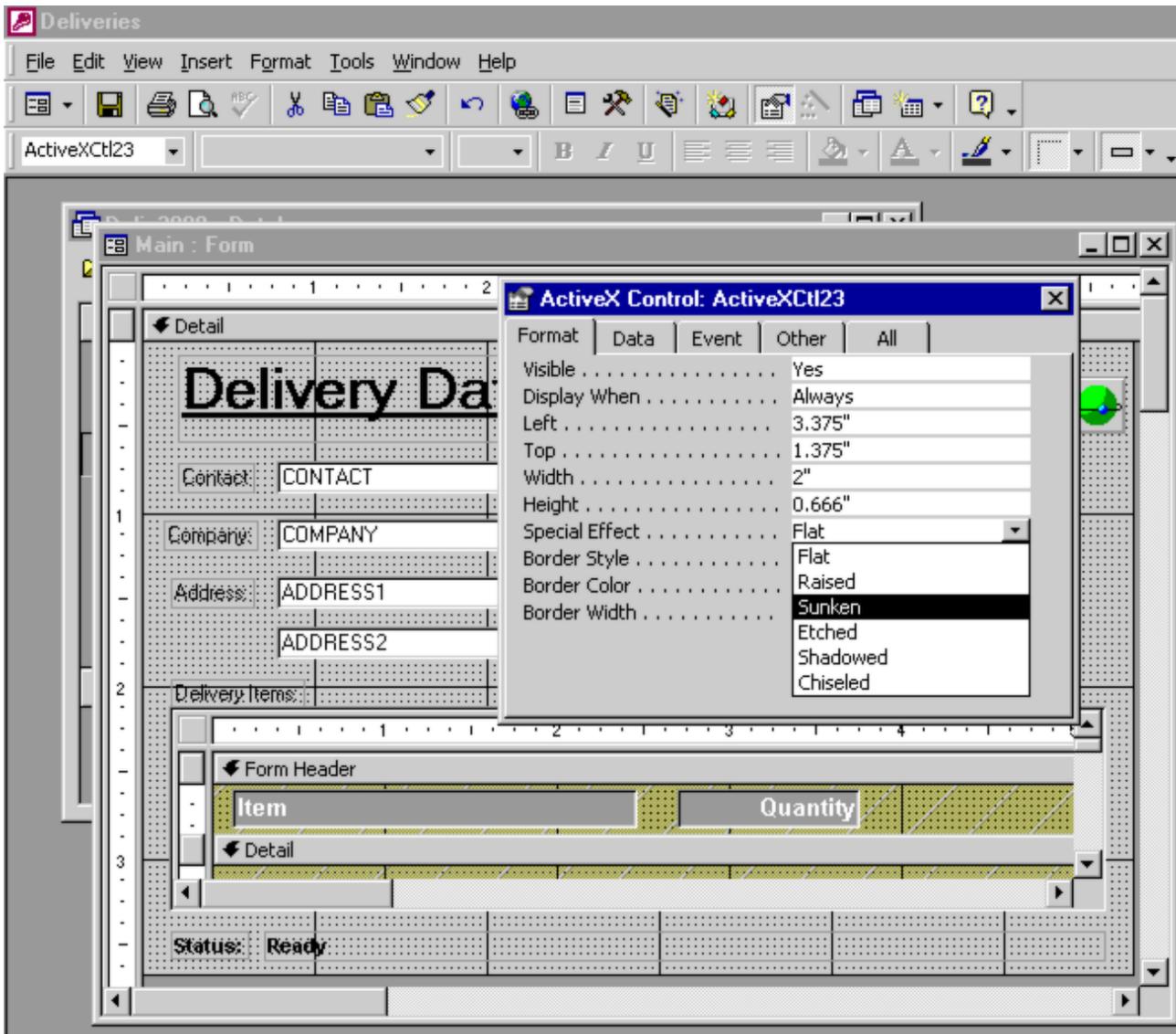
Step 5 - Resize the control to the desired width and height. Then right-click on the control and select "Properties".



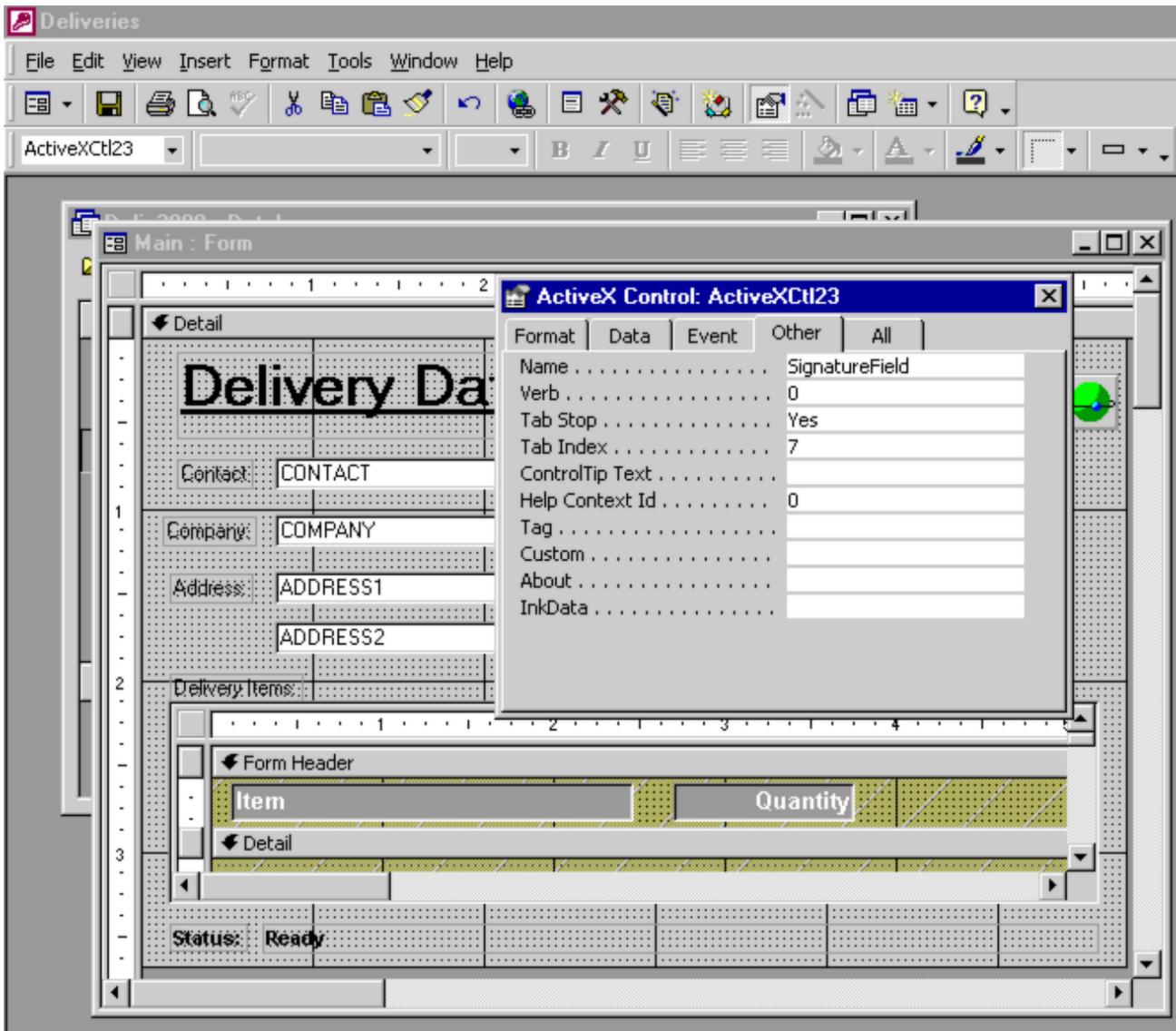
Step 6 - Once the "Properties" window is up select the "Data" tab. Clicking on "Control Source....." will bring up a list of columns in the table linked to this form. Select the column that contains the Ink Control data that needs to be displayed.



Step 7 - Select any "Special Effects" or border styles under the "Format" tab.



Step 8 - Select the "Other" tab and name this control. The name does not matter so name it anything you want. We'll rename it to SignatureField in this example.



Step 9 - That's it. You're done. View the finished form. Your form is now integrated with the Satellite Forms Ink View control.

Deliveries

File Edit View Insert Format Records Tools Window Help

Jack Nishimura Phone: (617) 999-1212

Company: Karaoke Inc. Customer Signature

Address: 150 Walnut Ave.

Delivery Items:

	Item	Quantity
▶	3" Spotlights (black)	5
	Microphone	1
	Mic. Stand	1
*		

Status: Ready

Record: 1 of 3

Keywords: ink, signature, bitmap, image, display, inkview, activex, ocx

KB ID: 10014
Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Install SatForms PocketPC Runtime to multiple handhelds

Problem: How To Install SatForms PocketPC Runtime to multiple handhelds

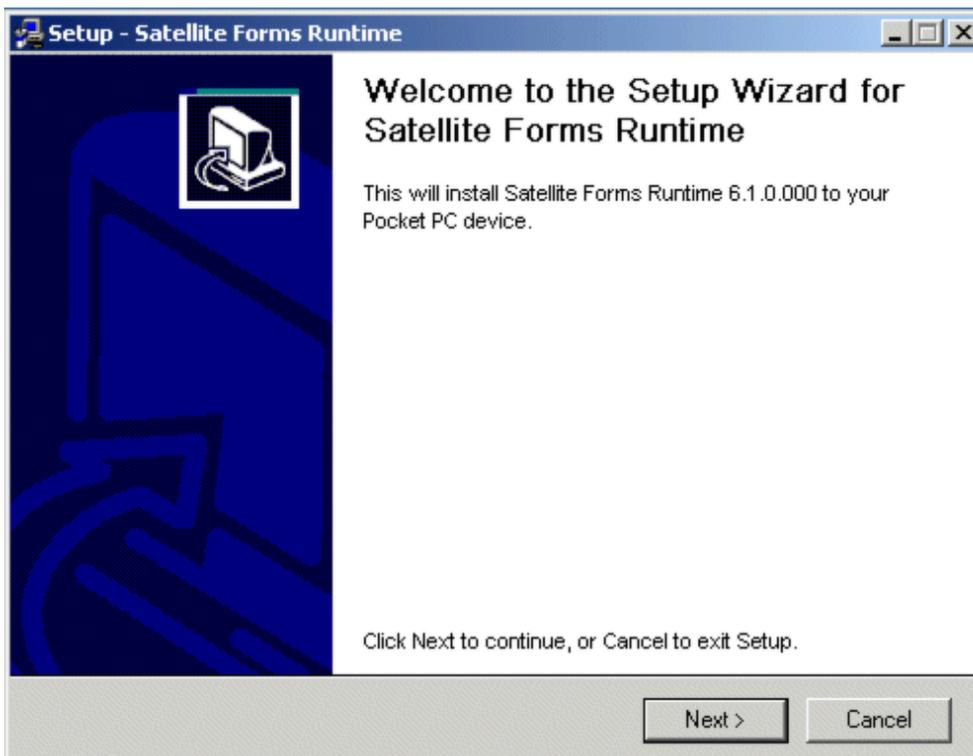
Updated for Version 8: With Satellite Forms 8 there is a new capability to use an integrated runtime engine, which removes the need to install the runtime engine files separately from the application files.

Updated: Starting with Satellite Forms 7.0, it is possible to bundle the PocketPC runtime engine directly with your application files, instead of needing to install the runtime engine separately. That is the new recommended approach, described in [How To Bundle the SatForms PocketPC runtime engine with your app](#). The information below is retained for Satellite Forms 6.x users.

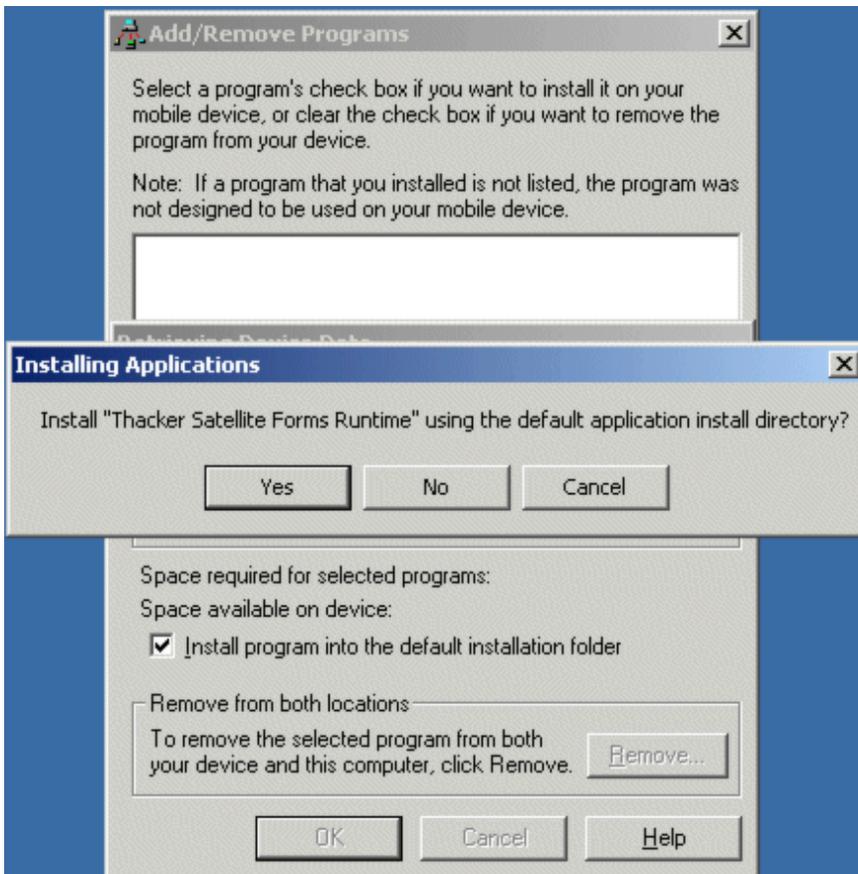
Solution: You can install the SatForms PocketPC runtime to multiple handhelds on a single PC using different methods.

1. Method 1 - SatForms Runtime Installer

To install the runtime on multiple handhelds connected to the same PC, you can simply run the RDK or SDK runtime installer when each handheld is attached to the PC and connected via ActiveSync. Use the installer shortcuts from the Satellite Forms menu, for example Start | Programs | Satellite Forms 6.1 | Runtime | PocketPC | RDK Runtime Installer.



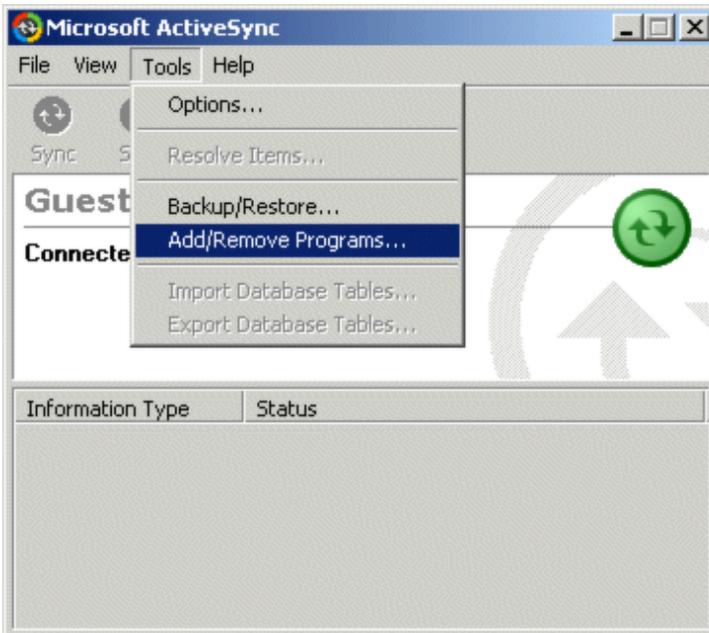
Follow the onscreen prompts to complete the installation.



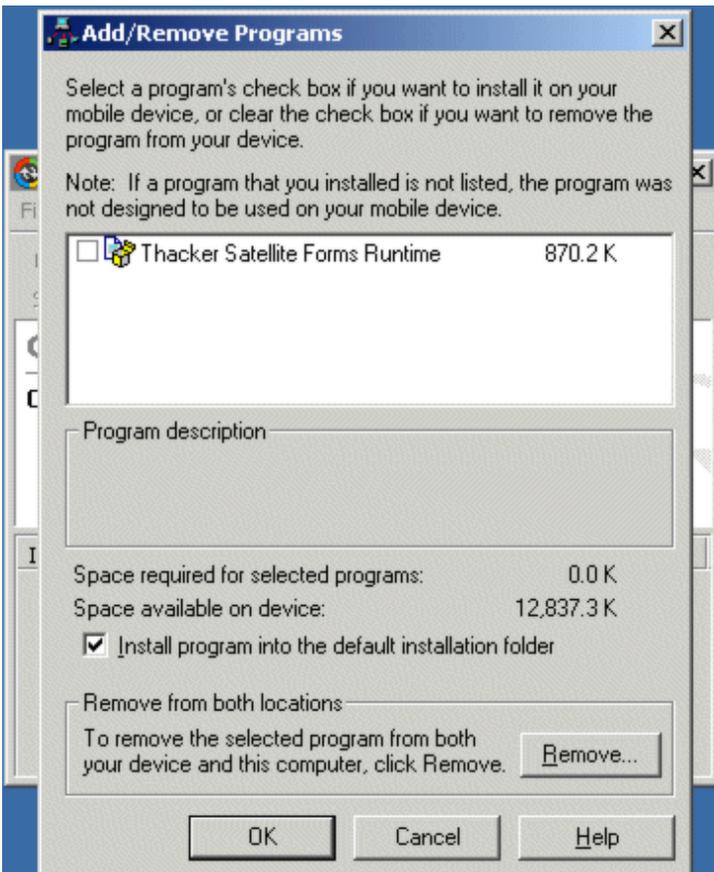
2. Method 2 - ActiveSync Add/Remove Programs

Another method is to use the Add/Remove programs option in Microsoft ActiveSync. Attached the PocketPC handheld to the PC, and wait for ActiveSync to make the connection. It does not matter if you have a sync relationship with this handheld, or use a guest connection.

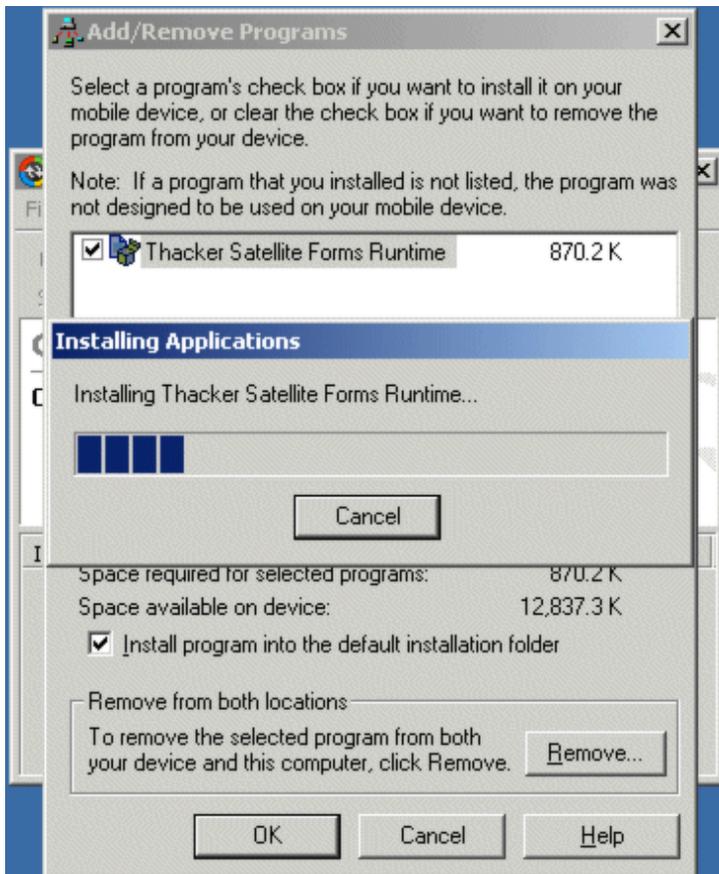
Select Add/Remove Programs from the Tools menu in the ActiveSync window.



ActiveSync will query the handheld and see what is installed. If the SatForms runtime is already installed on that device, it will be listed with a checkmark in the checkbox to the left of the name.



If it is not installed, the checkbox will be blank. Check the checkbox, and then tap on OK.



Follow any additional prompts to complete the installation.

3. Method 3 - Use the functions of the SatForms ActiveSync OCX to install via the runtime CAB file

See this KB article: [How To Install the SatForms Runtime for PocketPC Programmatically](#)

Keywords: Runtime, install, multiple, ActiveSync, PocketPC

KB ID: 10020

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Install the SatForms Runtime for Palm silently

Problem: How To Install the SatForms Runtime for Palm silently

Updated: Starting with Satellite Forms 7.1, the "Satellite Forms Runtime for Palm" has been renamed to "Satellite Forms Runtime for PCs", as that more accurately reflects what the runtime components are for, and applies to both Palm OS and PocketPC applications. See the Satellite Forms 7.1 manual for the updated runtime installation instructions. The information below is retained for Satellite Forms 6.x/7.0 users.

Q: Is there a way to hide or bypass the Satellite Forms logo from the Palm installer - or - is there a way to install and register these files without any user intervention?

Solution: Yes, there are several ways to install the SatForms Runtime for Palm to the desktop PC with reduced or no user interface:

1. Performing a silent installation using the Runtime Installer package:

(a) Launch the Setup.exe file with switches telling it to complete the installation with no user intervention at all:

```
setup.exe /s /v/qn
```

or

(b) Simply bypass the Satellite Forms logo but present the rest of the setup UI:

```
setup.exe /s
```

2. If you can assure that your target PC already has the required Microsoft Installer executable files installed, you can install directly from the .msi file instead of the setup package. You only need to work with the .msi file, and do not need any of the other files in the "Disk" folder.

```
msiexec.exe /q /i "Satellite Forms Runtime for Palm.msi"
```

or

```
"Satellite Forms Runtime for Palm.msi" /q
```

3. Utilize the redistribution kit merge module in your own installer package. Simply add the SatFormsRedist.msm merge module to your installer project. It will install and register all of the files without any UI.

4. Install and register the necessary files yourself using the instructions provided in the SatForms 6.1 help file in the "Deploying your Application | Creating a custom installer | File placement and registration" topic.

Keywords: Install, silent, runtime, palm, msi, setup

KB ID: 10021

Updated: 2007-12-03

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

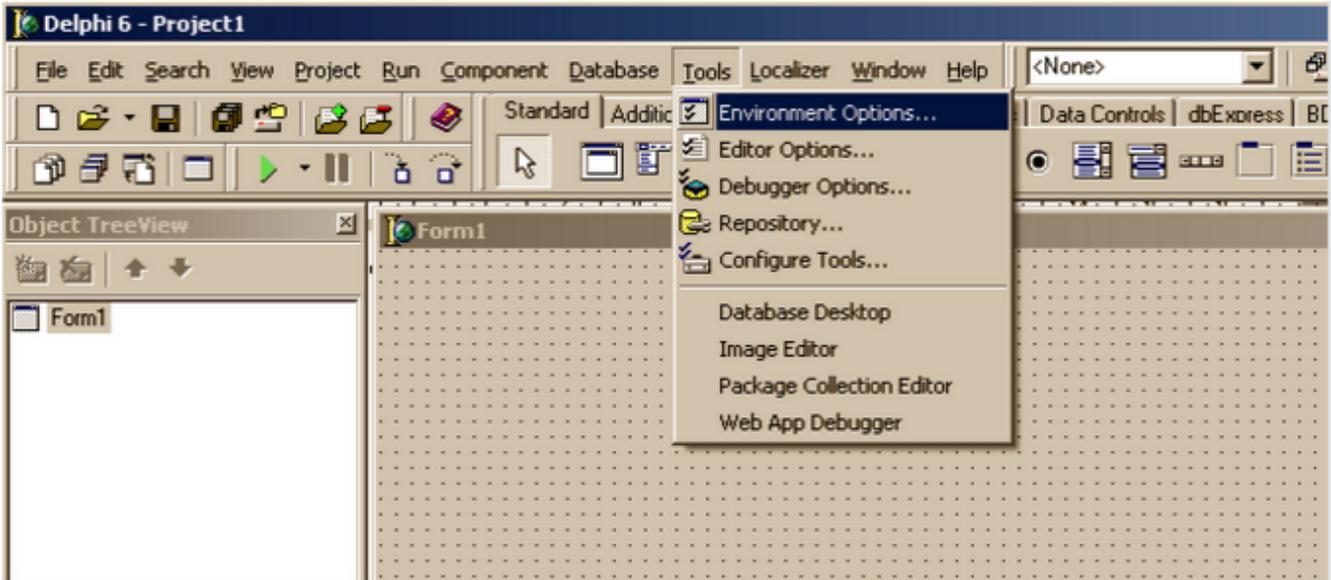
-0-

How To use the SatForms ActiveSync control with Delphi

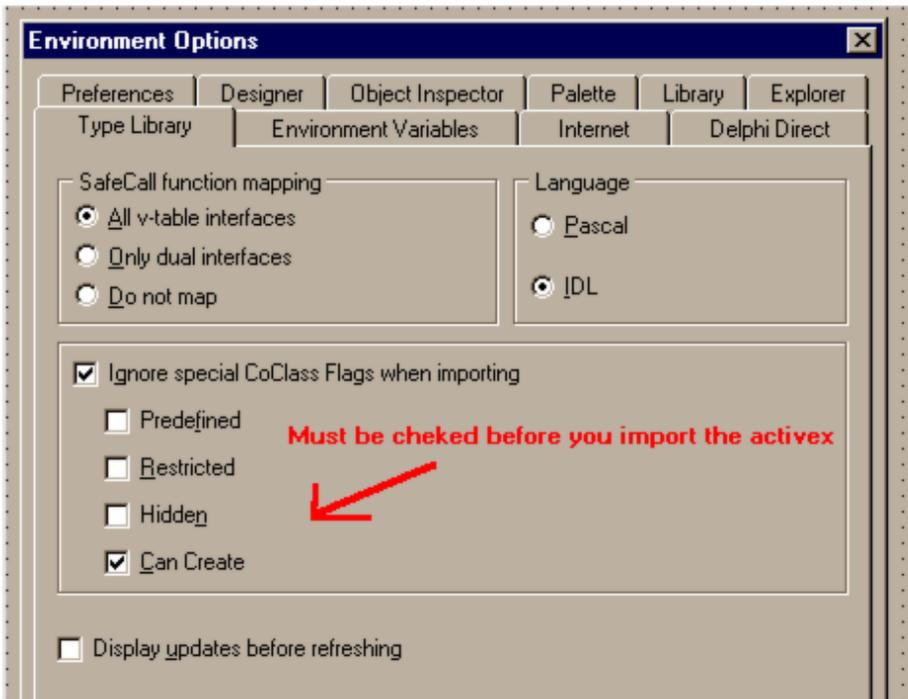
Problem: How To use the SatForms ActiveSync control with Delphi

Solution: Delphi 6-7 Step by Step

1. Select Menu tools/env. options

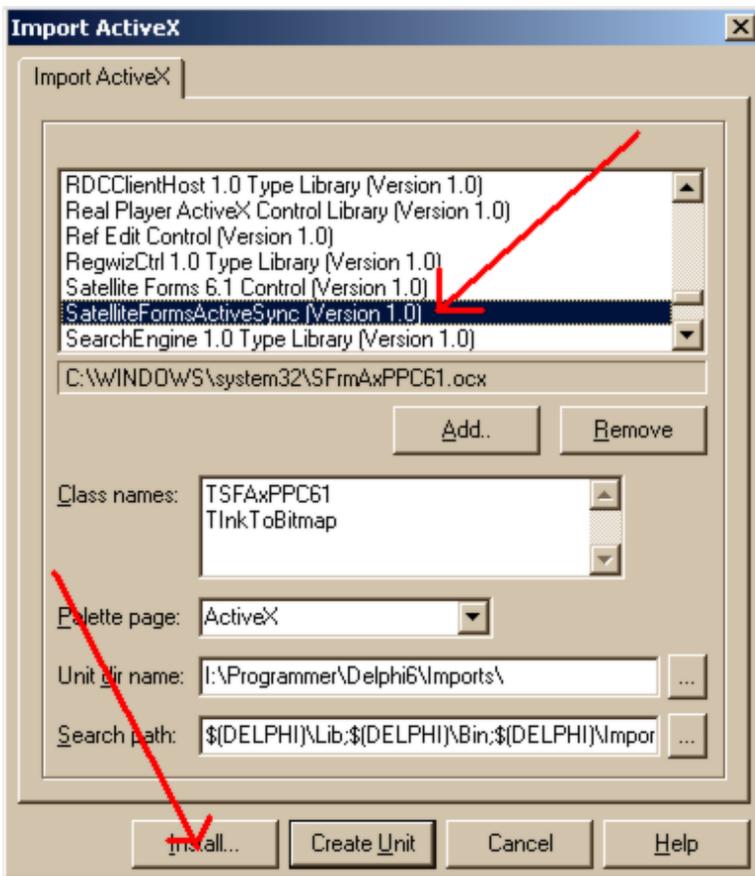
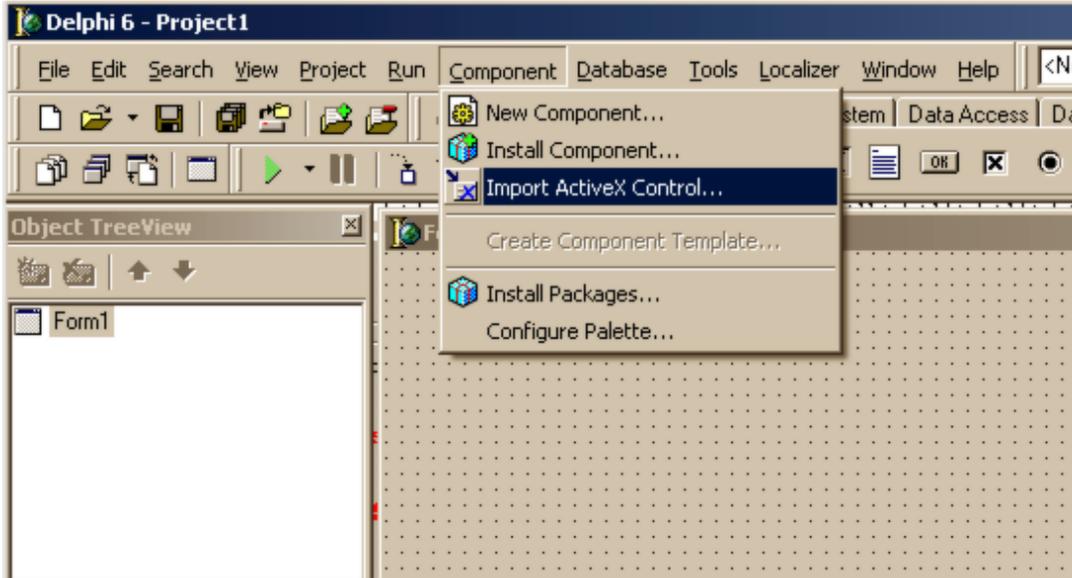


2. Select Type Library Tab and check "Can Create" Option



Click OK

3. Select Menu Component - Import Activex Control



Click Install

The activex will be Installed and will be available at the Selected Component palette location (here ActiveX)

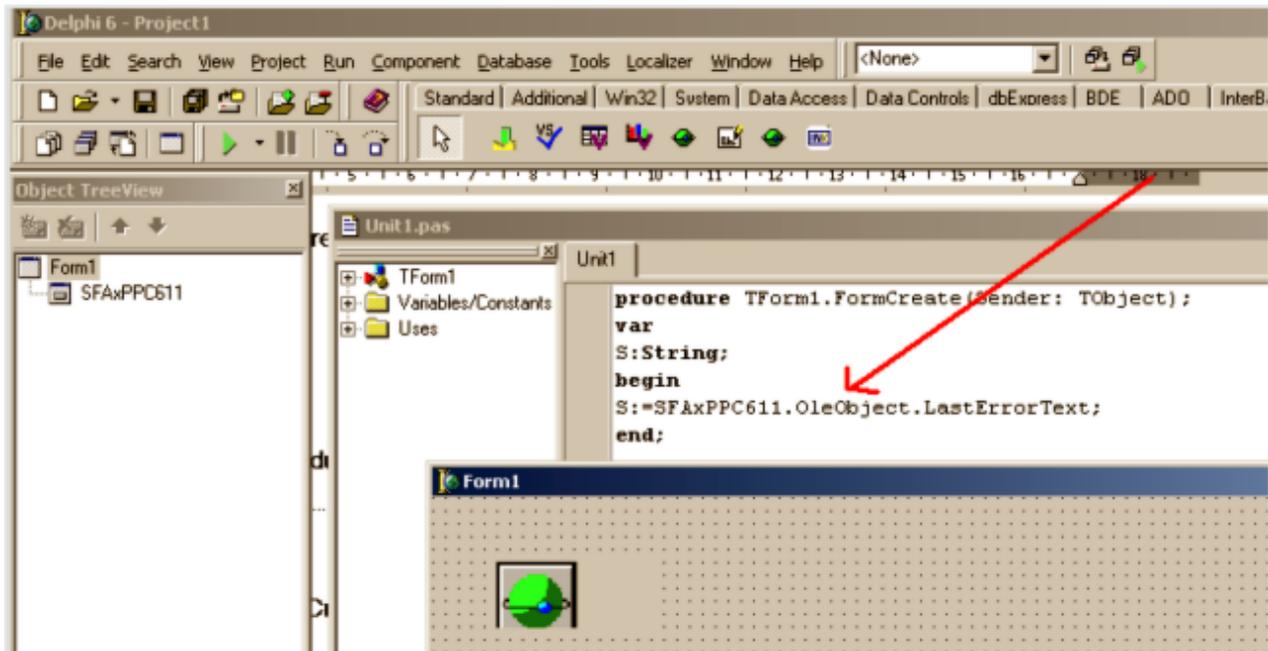
Now be sure that Microsoft Active Sync is active
Open your form
Add the activex (TSFAXPPC61)

For The activex to be created you must Use it at least once (or the events will not fire)

Click the events
OnCreate
Create an Event Procedure (here OnCreate)

Add the Code:

```
procedure TForm1.OnCreate(Sender: TObject);  
var  
S: string;  
begin  
S:=SFAXPPC611.OleObject.LastErrorText;;  
end;
```



Watch the Oleobject part of the string. This must be used when referring the functions and procedures in the SFAXPPC611 object. There is a slight difference in using this object in oppsite to normal Delphi objects because you must reference the Oleobject as part of the object calls.

This is all. You can use it and the events will fire.

In previous Delphi versions:

Use File|Open, set the filter to Type Library files, open the SFrmAxPPC61.ocx in the windows\system32 directory , and have a look at the CanCreate flag in the type library editor. Set it to true, save the typelibrary as SatelliteFormsActiveSync_TLB.TLB , and try importing that.

This article was contributed by Jorn Johanneson, based on a similar HowTo from PDA-TECH on their TopSync OCX.

<http://www.apimo.dk/>

Keywords: Delphi, ActiveSync, PocketPC

KB ID: 10025

Updated: 2008-05-16

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Create an Installer for your SatForms 6.x PocketPC Application

Problem: How To Create an Installer for your SatForms 6.x PocketPC Application

Solution: There are several different approaches to creating an installer for your SatForms PocketPC application. This article presents an approach that generates a single exe file to install the Satellite Forms runtime engine and your compiled application to the PocketPC, providing a smooth user experience.

[NOTE: This article is applicable to Satellite Forms 6.x PocketPC applications, and to SatForms 7 applications that use the CDB database format. A new and improved installer creation method is described for Satellite Forms 7.x and 8.x PocketPC applications in the article [How To Create an Installer for your SatForms 7 PocketPC Application](#)]

This example describes all steps needed to create an installer for the PocketPC application "Work Order Sample.exe", based on the SF sample application "Work Order". Directories and program names can be modified to suit your own application by changing the references in the *.bat, *.inf, *.ini, and *.txt files described below. The process described herein will enable you to create a single-EXE installer file for your SatForms PocketPC application. NOTE: You will need to repeat this process with your PPC2002 and PPC2003 target builds, resulting in separate EXE installers for each. See the article [How To use different platform targets for PocketPC applications](#) for more information about PocketPC targets.

The overall approach is to create a PocketPC CAB file for your application, then combine the SatForms runtime engine CAB and your app CAB into a single EXE installer.

NOTE: Do not let the length of this article dissuade you from using this installer creation process. This article is quite long so as to ensure no steps are left out, but you will find it is a straightforward process to create you own app installers using this method, and the professional looking results are worth the investment of your time.

This installer creation method utilizes a free open source installer creation tool called Nullsoft Scriptable Install System (NSIS). NSIS is similar in function to other installer creation tools such as InstallShield, Wise Installer, and InnoSetup, but is completely free of charge for personal and commercial use.

Please download the WorkOrderInstallerSample.zip file here:

<http://www.satelliteforms.net/support/WorkOrderInstallerSample.zip>

A. What does the installer do?

It implements a complete install program for the your PocketPC application. The install program "WorkOrderInstall.exe" in this example does the following (note that the PPC must be connected to the desktop and ActiveSync must be running):

1. Displays a license agreement (EULA) which the user must accept to proceed (you supply the text).
2. Installs the Satellite Forms runtime engine.
3. Installs the application "Work Order Sample.exe" and all *.cdb files into the "\Program Files\Work Order" folder on the PPC.
4. Creates a shortcut for the program in "\Windows\Start Menu\Programs" on the PPC.
5. Registers the program and creates an "Unload" file on the PPC for future deletion of the

application.

B. Files required to create the installer

1. Microsoft CABWiz files:

Cabwiz.exe, cabwiz.ddf, Makecab.exe

All of these files are available free of charge from Microsoft, in the Microsoft PocketPC 2002 SDK zip file. We have included them in the \Utils folder in the sample zip file.

2. Application files: Work Order Sample.exe (icon/executable for PPC), Work Order Sample.pda, all *.cdb files in the AppPkg folder (wrkLookup.cdb, wrkSites.cdb, wrkWorkItems.cdb). These are included in the sample zip file in the \WorkOrder folder, but you could also create them by compiling the PPC2003 target of the Work order sample project included with Satellite Forms.

3. Batch and configuration files for Cabwiz:

MakeCAB-WorkOrder.bat, WorkOrder.inf

4. Readme, EULA, installer icon files:

readme.txt, eula.txt, WorkOrderInstall.ico

Use these sample files as a template, and modify as needed for your application.

5. Satellite Forms Runtime Engine CAB file:

The Satellite Forms V6.1 RDK runtime engine installer CAB is provided in the \Satellite Forms

6.1\Redist\PocketPC\Runtime CABs\ folder:

SatFormsRuntimeRDK.Arm.CAB

6. Nullsoft Scriptable Install System:

Download the open source NSIS installer creation tool from <http://nsis.sourceforge.net> and install it to your development PC. Accept all of the default installation options.

C. Step By Step Process

C.1. Unzip the WorkOrderInstallerSample.zip to a folder on your desktop PC, for example C:\WorkOrderInstaller, (and make sure you allow the unzip process to create the subfolders within the archive).

C.2. Create the CAB files containing your app

Run the MakeCAB-WorkOrder.bat batch command. This will create a WorkOrderSample.Arm.cab file and other intermediate files in the C:\WorkOrderInstaller folder. [Note that this CAB file is also included in the sample zip file, so it will already exist in your folder.]

Note: you will likely get the error log file makecab.err created when you perform this step. It might look something like this:

Warning: Section [DefaultInstall] key "AddReg" - there are no section entries to process
Warning: Section [DestinationDirs] key "Shortcuts" is not using the string "%InstallDir%"
Warning: Section [DefaultInstall] key "AddReg" - there are no section entries to process

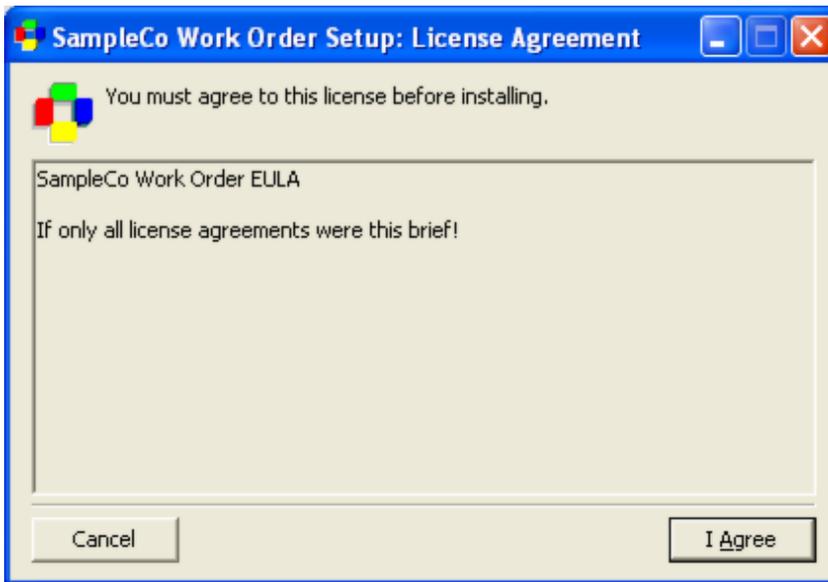
Those warnings are harmless. If you see other warning or error messages, then you will need to follow them up to resolve the problem.

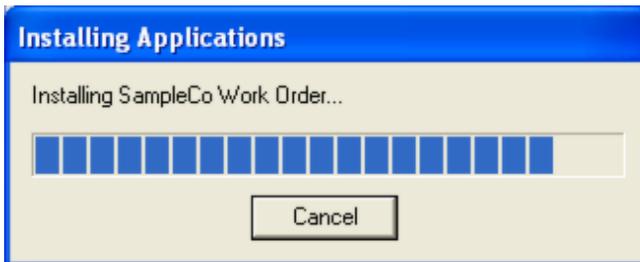
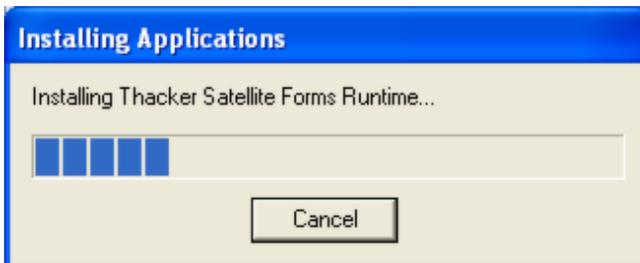
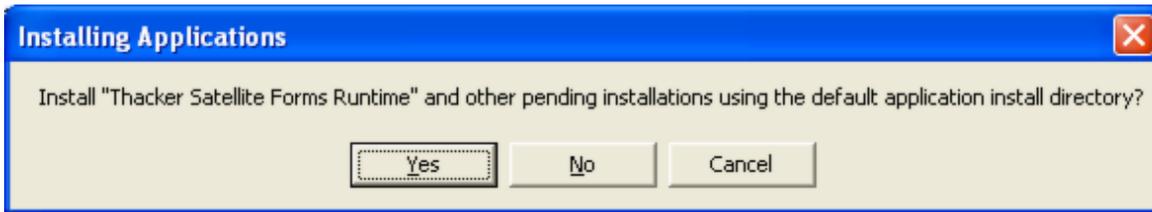
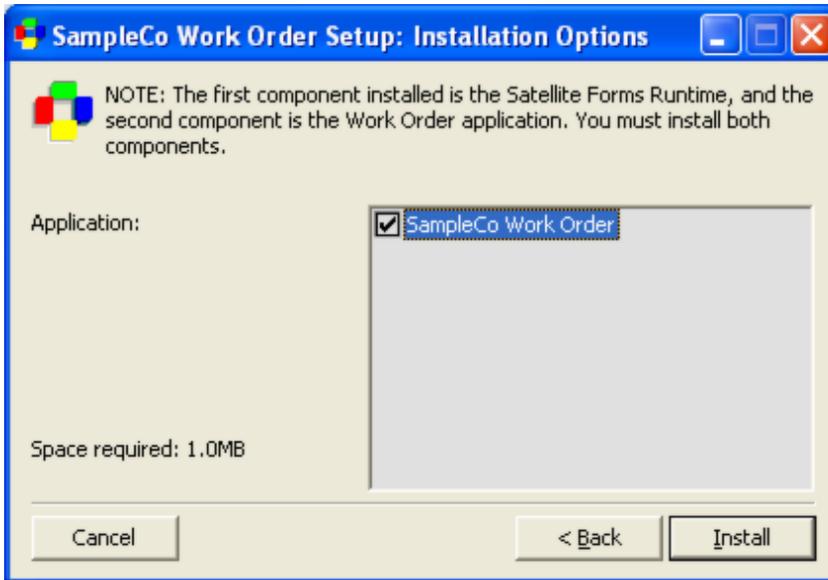
C.3. Copy the SatForms RDK runtime engine CAB file from \Satellite Forms 6.1\Redist\PocketPC\Runtime CABs\ to the C:\WorkOrderInstaller folder.

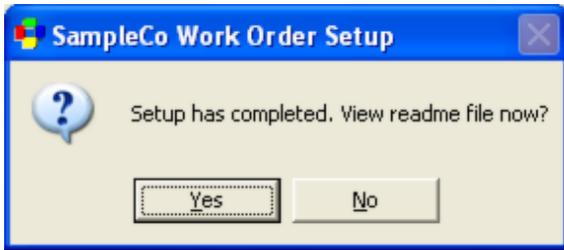
C.4. Use NSIS to convert the CAB files to a single EXE installer

Right-click on the WorkOrderInstall.nsi file and select Compile NSIS Script. This will launch the graphical interface for NSIS and compile the CAB files into a single WorkOrderInstall.exe file. NSIS uses the configuration parameters in the WorkorderInstall.nsi file to define how it creates the installer.

That's it! The installer is created, and NSIS displays the creation details. You can test the newly created installer by clicking on the Test Installer button. [You need to have your PocketPC device connected to the PC at this point.] Follow the onscreen prompts on the PC and the handheld as needed.







That's it: the SatForms runtime engine and your application are installed!

To start the Work Order app on your PPC, tap on Start | Programs | Work Order



D. Modifying the sample to suit your application

In order to modify the sample installer files to suit your application, several files must be edited.

D.1. CAB creation INF file for your app

Make a copy of the WorkOrder.inf file and rename it to YourApp.inf. The name you choose for

the INF file will determine the resulting CAB filename (eg. YourApp.Arm.CAB).

Edit the YourApp.inf file in a text editor to change references to Work Order and SampleCo to your app and company, and to change the files included in the application to your app files. The WorkOrder.inf file is marked up with the changes below. The sample entry is highlighted in grey, with the modified value highlighted in yellow. Notes are highlighted in blue:

--

[Version]

Signature="\$Windows NT\$"
CESignature="\$Windows CE\$"
Provider=%CompanyName%

[CEDevice]

BuildMax=0xE0000000
;this ensures that you do not get the "this app may not run on this version of windows Mobile" warning

[SourceDisksNames]

1=,"Application Files",, .\WorkOrder
1=,"Application Files",, .\YourApp

NOTE: This is the subfolder you must create. You need to copy your app EXE, PDA, and CDB files there from the AppPkg folder.

[DefaultInstall]

CopyFiles=Files.App
CEShortcuts=Shortcuts

[Shortcuts]

%AppName%,0,"Work Order Sample.exe"
%AppName%,0,"Your App.exe"

[CEStrings]

InstallDir=%CE1%\%AppName%
AppName="Work Order"
AppName="Your App"

[Strings]

CompanyName="SampleCo"
CompanyName="Your Company"
LinkFileName="Work Order"
LinkFileName="Your App"

[DestinationDirs]

Shortcuts = ,%CE11% ; create shortcut in \Windows\Start Menu\Programs
Files.App = 0,%InstallDir% ; install app to %CE1%\%AppName% = \Program Files\Work Order

[SourceDisksFiles]

Work Order Sample.PDA = 1
Work Order Sample.exe = 1
wrkLookup.cdb = 1
wrkSites.cdb = 1
wrkWorkItems.cdb = 1

```
Your App.PDA = 1  
Your App.exe = 1  
YTable1.cdb = 1  
YTable2.cdb = 1  
YTable3.cdb = 1
```

;note add your extension SFX files here too

```
[Files.App]  
Work Order Sample.PDA  
Work Order Sample.exe  
wrkLookup.cdb  
wrkSites.cdb  
wrkWorkItems.cdb
```

```
Your App.PDA  
Your App.exe  
YTable1.cdb  
YTable2.cdb  
YTable3.cdb
```

;note add your extension SFX files here too

--

D.2. MakeCAB batch command

Copy the MakeCAB-WorkOrder.bat file and rename it to MakeCAB-YourApp.bat. Edit the .bat file in a text editor and change the commandline from:

```
".\Utils\Cabwiz.exe" ".\WorkOrder.inf" /err makecab.err /cpu "Arm"  
to  
".\Utils\Cabwiz.exe" ".\YourApp.inf" /err makecab.err /cpu "Arm"
```

D.3. License and readme files

Edit the eula.txt and readme.txt files to your liking.

D.4. INI file

Copy the WorkOrder.ini file to YourApp.ini. Edit the YourApp.ini file in a text editor to change references to Work Order and SampleCo to your app and company. The WorkOrder.ini file is marked up with the changes below. The sample entry is highlighted in grey, with the modified value highlighted in yellow. Notes are highlighted in blue:

--

```
[CEAppManager]  
Version = 1.0  
Component = WorkOrder  
Component = YourApp
```

```
[WorkOrder]  
Description = Work Order application  
Uninstall = WorkOrder  
CabFiles = WorkOrder.ARM.cab  
[YourApp]
```

Description = Your Application
Uninstall = YourApp
CabFiles = YourApp.ARM.cab

--

D.4. NSIS install script

Copy the WorkOrderInstall.nsi file and rename it to YourApp.nsi. Edit the YourApp.nsi file in a text editor to change references to Work Order and SampleCo to your app and company, and to change the files included in the application to your app files. The WorkOrder.inf file is marked up with the changes below. The sample entry is highlighted in grey, with the modified value highlighted in yellow. Notes are highlighted in blue:

```
; NOTE: this .NSI script is designed for NSIS v1.8+  
;  
; This is a setup script that installs both the SatForms runtime engine and the compiled  
application  
; together as a single listed application using two CAB files (RDK runtime CAB plus your app  
CAB)  
; Both the SatForms runtime and your app will be listed in the Remove Programs tool and both  
; must be accepted during the install process  
; Note that all files are also installed to the PC hard drive in the INSTDIR folder
```

```
Name "SampleCo Work Order"  
Icon "WorkOrderInstall.ico" ;note this is not the same icon as the Work.ico used by SatForms  
OutFile "WorkOrderInstall.exe"  
Name "YourCompany Your App"  
Icon "YourAppInstall.ico" ;note this is not the same icon as the YourApp.ico used by SatForms  
OutFile "YourAppInstall.exe"
```

```
; Some default compiler settings (uncomment and change at will):  
; SetCompress auto ; (can be off or force)  
; SetDatablockOptimize on ; (can be off)  
; CRCCheck on ; (can be off)  
; AutoCloseWindow false ; (can be true for the window go away automatically at end)  
; ShowInstDetails hide ; (can be show to have them shown, or nevershow to disable)  
; SetDateSave off ; (can be on to have files restored to their original date)
```

```
; BrandingText " " ;hides the "NullSoft Install System" text if desired
```

```
LicenseText "You must agree to this license before installing."  
LicenseData "eula.txt"  
; Note: if you do not want to display a license dialog first just comment out the above two lines
```

```
; Note: this is the path on the PC (not the PDA) where the files will be installed  
; The PDA install location is defined in the CAB file itself
```

```
InstallDir "$PROGRAMFILES\Work Order"  
InstallDirRegKey HKEY_LOCAL_MACHINE "SOFTWARE\SampleCo\Work Order" ""  
InstallDir "$PROGRAMFILES\Your App"  
InstallDirRegKey HKEY_LOCAL_MACHINE "SOFTWARE\YourCompany\Your App" ""
```

```
ComponentText "NOTE: The first component installed is the Satellite Forms Runtime, and the  
second component is the Work Order application. You must install both components." ""  
"Application: "
```

```
ComponentText "NOTE: The first component installed is the Satellite Forms Runtime, and the
second component is the YourApp application. You must install both components." ""
"Application:"
```

```
Section "SampleCo Work Order" ; (default, required section)
Section "YourCompany Your App" ; (default, required section)
SetOutPath "$INSTDIR"
```

```
File eula.txt
File readme.txt
```

```
File WorkOrder.ini
File WorkOrder.Arm.CAB
File YourApp.ini
File YourApp.Arm.CAB
```

```
File SatFormsRuntimeRDK.ini
File SatFormsRuntimeRDK.Arm.CAB
```

```
; one-time initialization needed for InstallCAB subroutine
ReadRegStr $0 HKEY_LOCAL_MACHINE "software\Microsoft\Windows\CurrentVersion\App
Paths\CEAppMgr.exe" ""
IfErrors Error
Goto End
Error:
MessageBox MB_OK|MB_ICONEXCLAMATION \
"Unable to find Application Manager for PocketPC applications. \
Please install ActiveSync and reinstall this application."
End:
```

```
StrCpy $1 "$INSTDIR\WorkOrder.ini"
StrCpy $1 "$INSTDIR\YourApp.ini"
StrCpy $2 "$INSTDIR\SatFormsRuntimeRDK.ini"
Call InstallCAB
```

```
SectionEnd ; end of default section
```

```
Section "-post" ; (post install section, happens last after any optional sections) ; add any
commands that need to happen after any optional sections here
```

```
WriteRegStr HKEY_LOCAL_MACHINE "SOFTWARE\SampleCo\Work Order" "" "$INSTDIR"
WriteRegStr HKEY_LOCAL_MACHINE
"Software\Microsoft\Windows\CurrentVersion\Uninstall\SampleCo Work Order" "DisplayName"
"SampleCo Work Order (remove only)"
WriteRegStr HKEY_LOCAL_MACHINE
"Software\Microsoft\Windows\CurrentVersion\Uninstall\SampleCo Work Order" "UninstallString"
WriteRegStr HKEY_LOCAL_MACHINE "SOFTWARE\YourCompany\Your App" "" "$INSTDIR"
WriteRegStr HKEY_LOCAL_MACHINE
"Software\Microsoft\Windows\CurrentVersion\Uninstall\YourCompany Your App" "DisplayName"
"YourCompany Your App (remove only)"
WriteRegStr HKEY_LOCAL_MACHINE
"Software\Microsoft\Windows\CurrentVersion\Uninstall\YourCompany Your App"
"UninstallString" "" "$INSTDIR\uninst.exe"
```

```
; write out uninstaller
WriteUninstaller "$INSTDIR\uninst.exe"
```

```
MessageBox MB_YESNO|MB_ICONQUESTION \  
"Setup has completed. View readme file now?" \  
IDNO NoReadme
```

```
ExecShell open '$INSTDIR\readme.txt'
```

```
NoReadme:
```

```
Quit
```

```
SectionEnd ; end of -post section
```

```
ShowInstDetails nevershow ;never show installation details
```

```
; begin uninstall settings/section
```

```
UninstallText "This will uninstall SampleCo Work Order from your system"
```

```
UninstallText "This will uninstall YourCompany Your App from your system"
```

```
Section Uninstall
```

```
; add delete commands to delete whatever files/registry keys/etc you installed here.
```

```
Delete "$INSTDIR\uninst.exe"
```

```
Delete "$INSTDIR\eula.txt"
```

```
Delete "$INSTDIR\readme.txt"
```

```
Delete "$INSTDIR\SatFormsRuntimeRDK.ini"
```

```
Delete "$INSTDIR\SatFormsRuntimeRDK.Arm.CAB"
```

```
Delete "$INSTDIR\WorkOrder.ini"
```

```
Delete "$INSTDIR\WorkOrder.Arm.CAB"
```

```
Delete "$INSTDIR\YourApp.ini"
```

```
Delete "$INSTDIR\YourApp.Arm.CAB"
```

```
DeleteRegKey HKEY_LOCAL_MACHINE "SOFTWARE\SampleCo\Work Order"
```

```
DeleteRegKey HKEY_LOCAL_MACHINE
```

```
"SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\SampleCo Work Order"
```

```
DeleteRegKey HKEY_LOCAL_MACHINE "SOFTWARE\YourCompany\Your App"
```

```
DeleteRegKey HKEY_LOCAL_MACHINE
```

```
"SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\YourCompany Your App"
```

```
RMDir "$INSTDIR"
```

```
SectionEnd ; end of uninstall section
```

```
; Installs a PocketPC cab-application
```

```
; The var $0 contains the path to the CeAppMgr tool
```

```
; It expects $1 and $2 to contain the absolute location of the ini files
```

```
; to be installed.
```

```
Function InstallCAB
```

```
ExecWait ""$0" "$1" "$2""
```

```
FunctionEnd
```

```
; eof
```

```
--
```

Note that there are other options that can be adjusted in the NSIS script if desired -- see the NSIS documentation for details.

That is it, you do not need to modify any other files in order to customize the installer sample to suit your application. Once you have made the changes, repeat the step by step process (C.1. through C.4.) to create your own single-exe application installer.

Keywords: PocketPC, install, installer, NSIS, CAB, CABWiz, setup

KB ID: 10027

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)

[Satellite Forms Website Home](#)

-0-

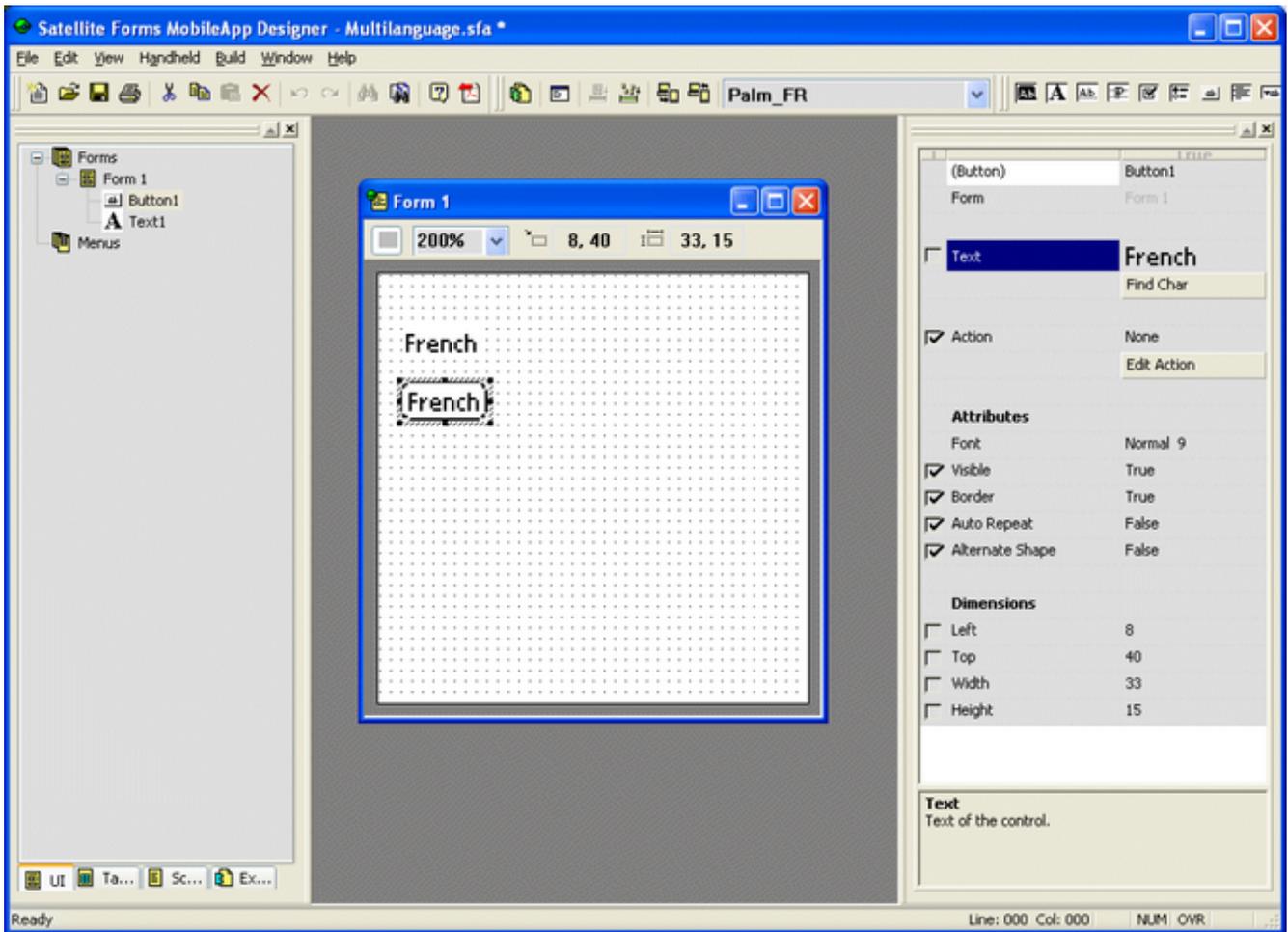
How To support multiple languages using build targets

Problem: How To support multiple languages using build targets

Solution: Using multiple Build Targets in your application project enables easier support of multiple language versions of your application

At the present time Satellite Forms does not offer special features to support localization of an application into different languages. However, you can add additional targets to your application in the same platform (eg. multiple Palm targets), so you could have an English target, French target, etc. You would need to modify the UI of your application to the different languages in each target, but you could keep that in a single project file with common code, rather than multiple projects. It would probably be easier to maintain that multi-target project than it would be to maintain multiple projects each with a single target.

You can set different control properties on a per-target basis, including the text of the control. Go to the French target, and display the form, click on the button to select it. In the control property space, uncheck that checkbox to the left of the Text property.



Then, change the text to suit that target. Switch to the other target, and change the text as needed. That checkbox column indicates whether that control property is shared (checked) by

all targets, or private (unchecked) for that target only.

Note also that you can have both shared and private global functions and subs, with the private ones existing in the current target only. So, you can have different scripts with the same function names between targets, which could for example do some target-specific things relating to the different languages.

The SF runtime engine itself does use string resources for error messages that you could modify with an external localization tool (RsrcEdit for example), and you could do the same with the standard menu contents as well. This would mean you would have language-specific versions of the runtime engine.

Keywords: localization, internationalization, language, targets

KB ID: 10028

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Change a control color using SFControlMagic

Problem: How To Change a control color using SFControlMagic

Solution: It is possible to change the color of a single control on a form using the PalmDataPro SFControlMagic extension (<http://www.palmdatapro.com/itm00090.htm>).

Updated for Version 8: With Satellite Forms 8 there is a new Colorizer extension that enables you to add color to forms and controls.

The PalmOS color UI system is based on working with groups of common form elements (such as controls) together at the same time, rather than on individual color support for each control. Since the PalmOS color system is designed this way, that is the way the SFControlMagic extension works. If you set the UI colors and then repaint the screen, all of the form elements are changed.

However, it is possible to change the color of a single control in this manner:

```
'temporarily change the color of one control  
'the color will automatically be reset back to the "normal" color  
'when the control is tapped (though you may be able to handle  
'that with the OnClick script of a button or OnPenDown/OnPenUp  
'events for other control types by applying the color change again)
```

```
'let's change the btnPDPTHEME button background color to the index value  
'in the edCI control (use PickcolorIndex to select a color value)  
'in the SFControlMagic sample application  
dim oldColor  
oldColor = GetColorIndex(1)  
SetColorIndex(1, edCI)  
btnPDPTHEME.visible = true 'redraw it  
SetColorIndex(1, oldColor)
```

Keywords: color, control, SFControlMagic, PalmDataPro

KB ID: 10031

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To use the PocketPC Emulator with Satellite Forms

Problem: How To use the PocketPC Emulator with Satellite Forms

Solution: Microsoft provides a standalone PocketPC 2003SE Emulator that allows you to develop and test PocketPC applications on your PC without having an actual PocketPC device attached. It is similar in function to the PalmOS Emulator, but emulates a PocketPC 2003SE device. Syncing between the emulator and your PC with ActiveSync is supported.

NOTE: This article used to be based on the Microsoft PocketPC 2003SE Emulator, but that emulator is no longer available from Microsoft. This article has been updated to reflect the current Windows Mobile 5 PocketPC emulator available from Microsoft, here:

<http://www.microsoft.com/downloads/details.aspx?FamilyID=c62d54a5-183a-4a1e-a7e2-cc500ed1f19a&DisplayLang=en>

Using the PocketPC emulator gives you the ability to:

- compile PocketPC CDB applications in Satellite Forms without having a PPC device attached
- test your PocketPC application without having a real device
- improve your development testing productivity even when you have a real device
- demonstrate your application to many people at once via a PC or laptop

These instructions explain how to get the PocketPC emulator set up for use with Satellite Forms.

Download the PocketPC emulator (Standalone Device Emulator 1.0 with Windows Mobile OS Images) from the above URL. There are two files to download, V1Emulator.zip (867 KB) and efp.msi (57 MB !!). Next, download the "Virtual Machine Network Driver for Microsoft Device Emulator" from the URL:

<http://www.microsoft.com/downloads/details.aspx?familyid=DC8332D6-565F-4A57-BE8C-1D4718D3AF65&displaylang=en>

1. Unzip the V1Emulator.zip file and then run the standalone_emulator_V1.exe installer. Accept all default prompts to complete the installation. This part is simple, and there are no new Start menu entries or desktop icons when the install is done.
2. Launch the netvswrap.msi installer (the virtual machine network driver), and accept all install defaults. Again, there are no new Start menu entries or desktop icons when the install is done.
3. Double click the efp.msi installer (Windows Mobile 5 OS images), and accept the default install prompts. When this install is completed, there will be a new Start menu program folder "Microsoft Windows Mobile 5.0 MSFP Emulator Images" with several PocketPC emulator image choices.
4. Start | Programs | Microsoft Windows Mobile 5.0 MSFP Emulator Images | PocketPC Coldboot.

That fires up the emulator, which takes a little while from a cold boot.

You may see an error message "Failed to open the VPC Network Driver...". Just dismiss that by clicking on OK. The emulated PocketPC will continue to boot. Once it is fully booted up, go through the setup screens.

5. Start ... | Device Emulator Manager

This is the key to making it sync with ActiveSync. You should see a cryptic string of letters and numbers shown in the list. Right click on it and choose Cradle.

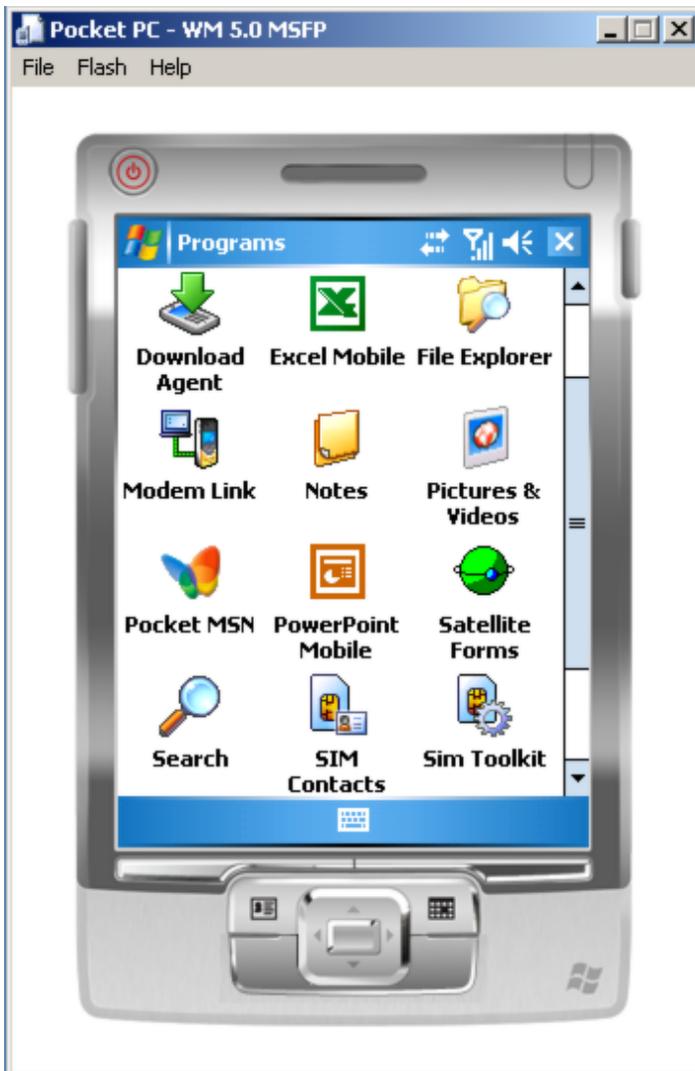
6. Open ActiveSync, then choose File | Connection Settings. In the Allow serial... droplist, select DMA, then OK. ActiveSync should then find the emulator, and you can then setup a sync partnership.

At this point, you can sync with the emulator just like a real device!

You can go ahead and install the SF runtime engine (use the link from the Start menu). Then, I would suggest closing the emulator and saving its state, so that you do not need to go through the cold boot and setup process next time.

Right click on the emulator in the Device Emulator Manager screen, and Uncradle. Then just click on the close X button on the emulator window, and answer Yes to the prompt to save state.

Now, to start it again, go Start ... | Pocket PC - Savestate to load that saved session. Refresh the Device Emulator Manager list, and re-cradle the emulator. (Sometimes ActiveSync does not see the emulator when you restart it, so you might have to go through the Get Connected steps again.)



Keywords: emulator, test, virtual, PocketPC, POSE, simulator

KB ID: 10032

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To use PalmDB (PDB) tables in a PocketPC application

Problem: How To use PalmDB (PDB) tables in a PocketPC application

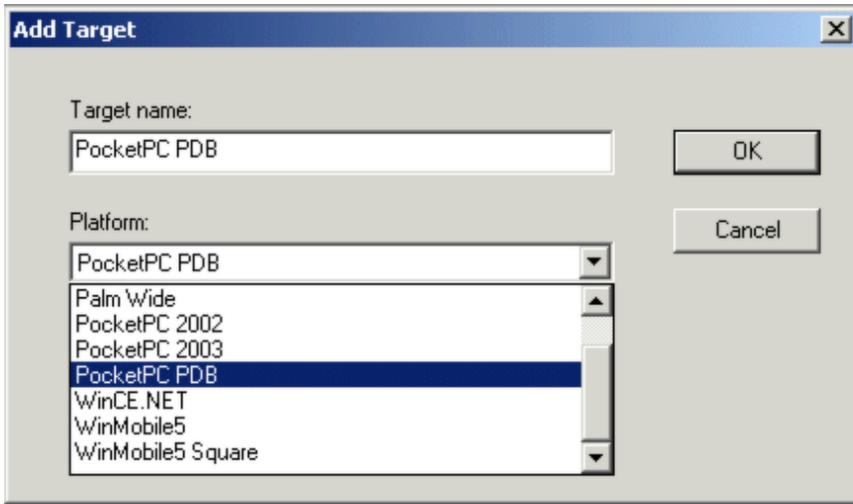
Updated: Starting with Satellite Forms 7.1, the recommended build target is named "PocketPC". Wherever you see "PocketPC PDB" in the article below, use "PocketPC" instead with Satellite Forms 7.1 and higher.

Solution: Starting with Satellite Forms 7.0, PocketPC applications can utilize the more efficient Palm Database (PDB) format for handheld tables, instead of the PocketPC Compact Database (CDB) format.

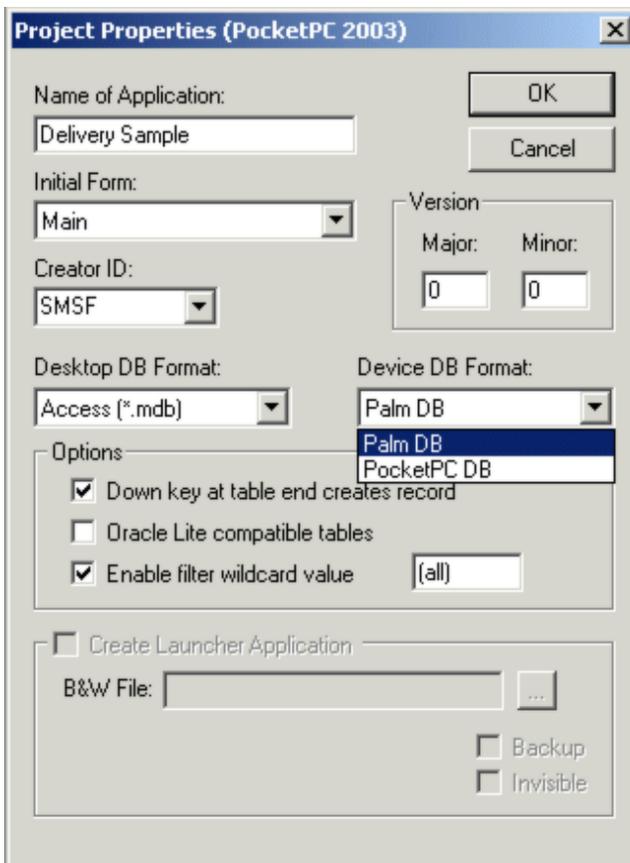
Using PDB format tables on the PocketPC offers numerous advantages over CDB format tables, including:

- PDB tables are significantly smaller than CDB tables because of their more efficient structure. While CDB tables have a minimum size of 48KB even when they are empty (128KB on PocketPC 2002 devices), the minimum size of a PDB table is a mere 80 bytes.
- Using PDB tables on the PocketPC removes the reliance on the Microsoft ActiveSync functions that convert desktop databases to/from the handheld CDB databases, thus improving the speed and reliability of data synchronization. With the changes Microsoft made to ActiveSync 4.x, handheld database synchronization using CDB tables has become less reliable than it was with ActiveSync 3.x, especially on Windows Mobile 5 powered devices. Using PDB tables on the PocketPC avoids these reliability and performance issues.
- With the use of PDB tables on the PocketPC and PalmOS devices, the data tables can be transferred between platforms with complete compatibility. You can use the same PDB tables created on the desktop PC with both PocketPC and PalmOS handhelds. You can transfer PDB tables directly between devices on different platforms, using infrared beaming, Bluetooth, or SD memory cards.
- Compiling Satellite Forms applications for the PocketPC platform is quicker and easier using PDB tables than it is with CDB tables, because there is no need to generate the handheld tables on the connected device at compile time (this is how CDB tables are created, which makes the compile process take longer). With the PDB format, the data tables can be generated directly on the desktop PC without needing a connected PocketPC device.
- The same application target can be used for PocketPC 2002 devices in addition to PocketPC 2003, 2003SE, and Windows Mobile 5 for PocketPC devices. There is no need to build a separate application target for PocketPC 2002 devices when using PDB tables, as there was when using CDB tables. PocketPC 2002 CDB tables are not compatible with PocketPC 2003 and later devices.

To create a new PocketPC application that uses PDB tables, we recommend using the new PocketPC application platform target. This build target is preset to use PDB tables on the PocketPC platform. The figure below shows the PocketPC PDB platform option in the Add Target selection screen that is displayed when you create a new project:



To modify an existing PocketPC application to switch from PocketPC CDB databases to Palm PDB databases, open the Project properties for your existing project. In the Device DB Format droplist, select Palm DB instead of PocketPC DB. That build target will now generate and use PDB tables instead of CDB tables.



Another possible option for existing applications is to add another build target using the PocketPC platform to your project. This would result in your application having multiple build targets, to select either the PocketPC CDB or Palm PDB format handheld tables as per your

needs.

TIP: Unless you specifically require the PocketPC CDB format for the tables in your handheld application (perhaps due to some server synchronization requirements, for example), we recommend using Palm PDB format tables for your Satellite Forms PocketPC applications. The advantages of the PDB format make it the preferred choice, especially on Windows Mobile 5 devices.

Keywords: Palm, PDB, PocketPC, CDB, table, platform, target, build

KB ID: 10033

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

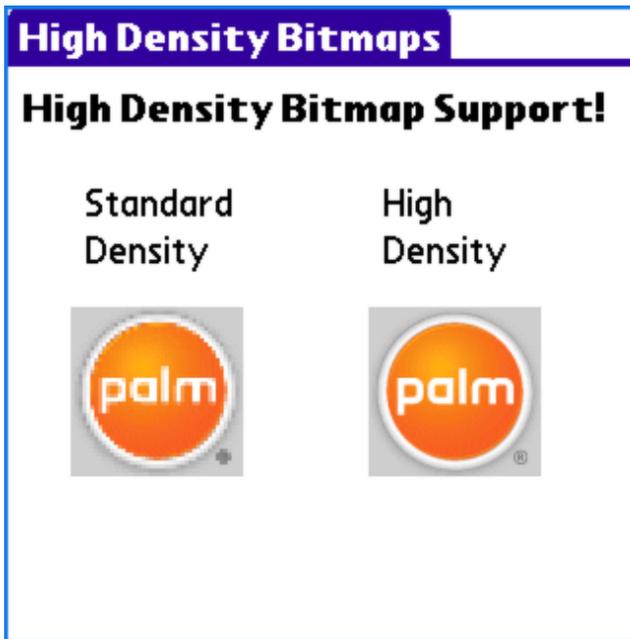
How To use High Density Bitmaps in PalmOS applications

Problem: How To use High Density Bitmaps in PalmOS applications

Solution: Starting with Satellite Forms 7.0, high density (HD) bitmaps can be used in your PalmOS applications. Most Palm Powered devices running the PalmOS 5.x operating system have higher resolution screens (320x320 pixels) than previous generation PalmOS devices that have 160x160 pixel screens. Rather than adopting a design like most PCs where the physical size of objects on the screen change based on the screen resolution, PalmOS designers cleverly adopted a different approach where the density of objects on the screen increased for higher display quality, but the physical size remains the same as with previous generation standard density displays.

On Palm Powered devices with high density screens, the number of pixels in a screen object is quadrupled in the same physical space, as the pixels have doubled both in width and in height. This design automatically results in sharper fonts, and if the application includes them, sharper images as well. On a device with the standard density screen, the standard density fonts and images are shown.

Previous versions of Satellite Forms included support for standard density images only. Now, with Satellite Forms 7.0, high density images for PalmOS applications are also supported. An example application showing both a standard density and high density version of the same image is shown below (displayed on a high density screen):



As you can see, the high density image occupies the same amount of space on the screen, but is much sharper.

Assuming you are already familiar with [how to use color bitmaps in your application](#), adding support for high density images is easy. High density bitmaps are supported at the 8 and 16 bit color depths. To add an 8 bit high density image to your application, save the high density version of your bitmap with the -8-HD.bmp file suffix. To add a 16 bit high density image to your application, save the high density version of your bitmap with the -16-HD.bmp file suffix.

The HD versions of your bitmaps need to be exactly twice as wide and twice as tall as the standard density versions. In the example above (included with SatForms 7.0 as the HighDensityBitmaps sample project), the standard density bitmap is 44x43 pixels. The high density version of the image is thus 88x86 pixels in size.

App Designer will always display the standard density version of the image in the form designer view: it will not display the high density version of the image even if it is available.

Note: PalmOS record size limitations require that image families (a collection of the same image at different color bit depths, eg. monochrome, 8-bit and 16-bit colors) must not exceed a total of 64KB in size. Because high density images require 4 times as many pixels as standard density images, this limitation becomes much more of a concern than it does with standard density images.

TIP #1: As demonstrated in the HighDensityBitmaps sample project, one strategy to reduce the total size of a bitmap family in order to make room for high density images, is to reduce the number of bit depths included in your image family. The Palm logo image in the sample project is included at the required monochrome depth plus an 8-bit standard density, 16-bit standard density, and 16-bit high density version. The app does not include an 8-bit high density version, because high density PalmOS devices can automatically use the 16-bit version. In fact, if your app is guaranteed to run only on PalmOS 5.x devices (whether they have standard density or high density screens), you can opt to leave out the 8-bit images altogether in favour of just the 16 bit versions.

TIP #2: If you find that a particular image you want to display is simply too large, divide it in half into two images. Place them side by side on your form, and the result should appear to be the single full size image on the screen.

Keywords: bitmap, high density, HD, pixel, color, image, logo, picture

KB ID: 10034

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)

[Satellite Forms Website Home](#)

-0-

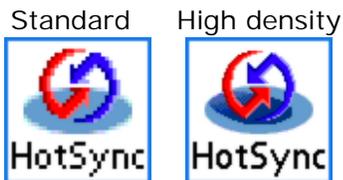
How To use High Density Icons for your PalmOS applications

Problem: How To use High Density Icons for your PalmOS applications

Solution: Starting with Satellite Forms 7.0, your application can now include high density icons. You can add high density icons for both the standard and small (list view) sizes of your application icon. High density icons look much sharper on the Palm application launcher screen, because they use four times as many pixels as the standard density icons.

As described in the KB article [How To use High Density Bitmaps in PalmOS applications](#), many newer generation Palm Powered devices have higher density 320x320 pixel screens compared to the older generation PalmOS devices that had 160x160 pixel screens. In addition to supporting higher density bitmaps and sharper fonts, high density capable PalmOS devices can also support high density application icons in the launcher screen. Satellite Forms 7.0 now enables you to use both standard and high density application icons.

The sample images below demonstrate standard density and high density versions of the HotSync icon:



To add high density icons for your application, simply name your large color icon with the -Color-HD.bmp file suffix, and place it in the same folder as your -Color.bmp standard density icon bitmap file. To add an HD version of your small (list view) icon, simply name your icon bitmap with the -Small-Color-HD.bmp file suffix.

The HD versions of your icon bitmaps need to be exactly twice as wide and twice as tall as the standard density versions. Therefore, the high density version of your large size icon needs to be 62x42 pixels, compared to the standard density color icon bitmap size of 31x21 pixels. The high density small color icon needs to be 30x18 pixels, compared to the 15x9 size of the standard density version.

TIP: To make icon creation for your PalmOS application easier, we recommend using the icon templates provided in the \Templates subfolder of your Satellite Forms program folder. Templates are provided for the standard and high density large icons, and standard and high density small icons. The use of the template file also ensures that the correct PalmOS color palette is used for your icons, resulting in the best possible image.

NOTE: If you do not include high density versions of your application icons, Satellite Forms will automatically insert default high density icons. on devices with high density screens, these default icons will be displayed instead of your standard density icons. The default high density icon appear as a red circle with a pen, like these:



Keywords: icon, high density, HD, color, red circle

KB ID: 10035

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To support Expandable Screens in PalmOS applications

Problem: How To support Expandable Screens in PalmOS applications

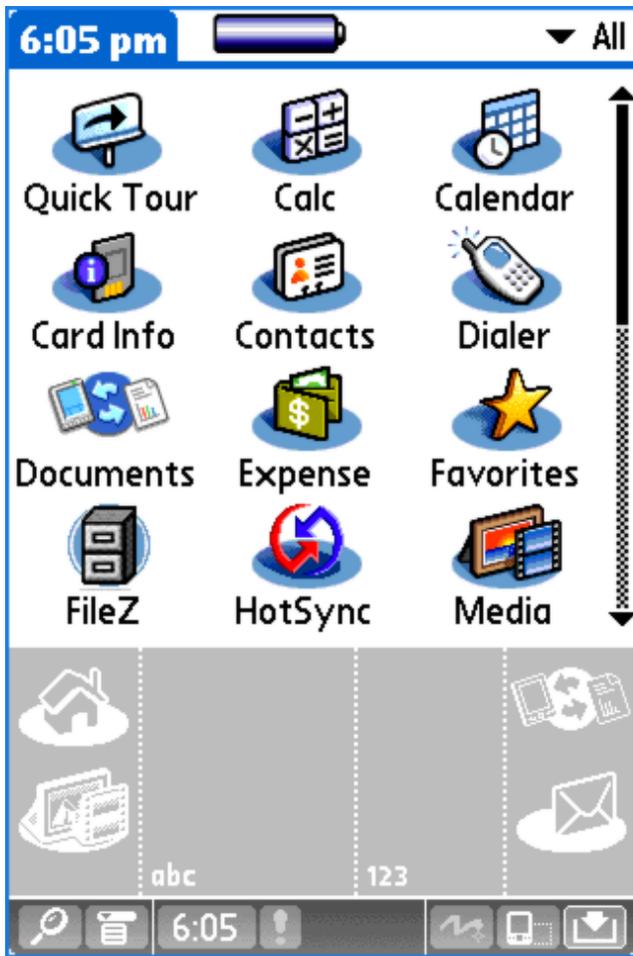
Solution: Starting with Satellite Forms 7.0, your PalmOS applications can now support the expandable screens and dynamic input area available on some Palm Powered devices.

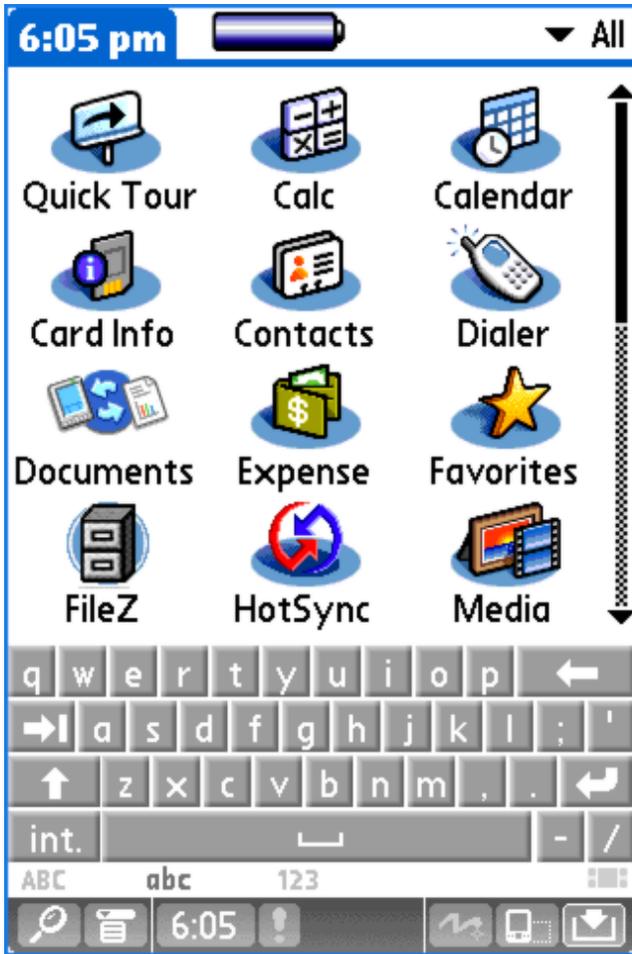
Background Information

Some newer PalmOS devices include support for using the area of the screen that was formerly dedicated to the silk-screened Graffiti input area, and for rotating the display between tall (portrait) and wide (landscape) display orientations. This area of the screen can display the standard Graffiti input area, or other input options such as an onscreen keyboard, or it can be utilized by applications to display larger forms. This area of the screen is known as the Dynamic Input Area (DIA), and is sometimes called the "soft" or "virtual" Graffiti area.

Some Palm Powered devices that have dynamic input areas include the Palm Tungsten T3, T5, TX, and LifeDrive, the Garmin iQue 3600, the Tapwave Zodiac 1 and 2, and some Sony handhelds.

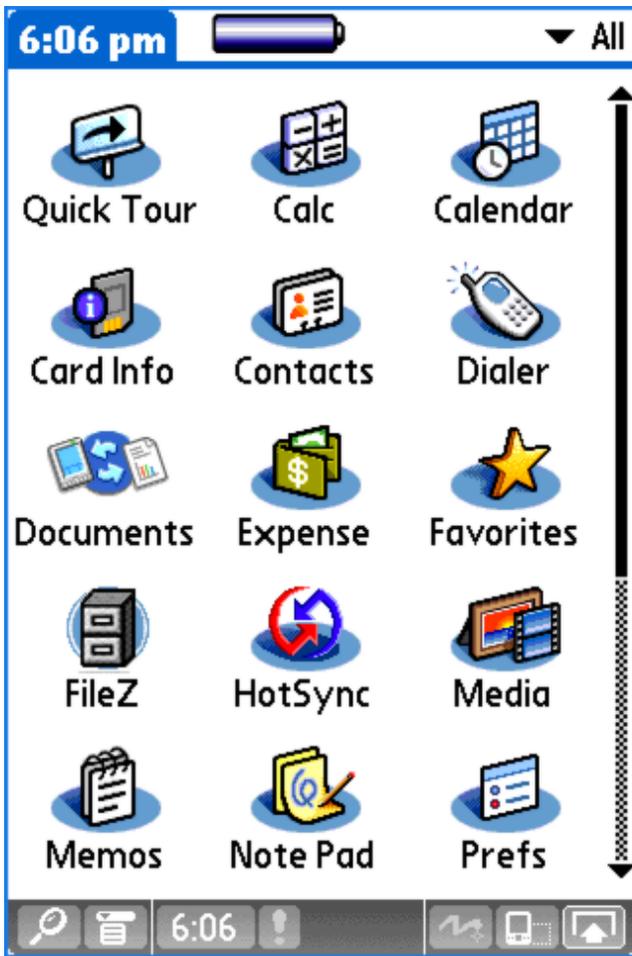
The screenshot below shows two different input configurations available on some PalmOS devices with a Dynamic Input Area (DIA). The first shows the virtual Graffiti mode, which resembles and acts like the standard input area on devices that do not have a DIA. The second shows a keyboard input mode that is quite handy (these screenshots are from a Palm TX):





Note the input trigger icon on the status bar, at the far right. The input trigger enables the user to decide whether to display the DIA, or to minimize it for more screen space. The status bar icon to the left of the input trigger in these screenshots allows the user to switch the screen orientation between portrait mode and landscape mode. The Palm LifeDrive has a hardware button on the side of the device to perform this function, instead of an icon on the status bar.

The next screen demonstrates the extra screen space that is available to applications when the DIA is minimized. The form expands in height to become taller (portrait orientation) and a small status bar is displayed along the bottom of the screen.



When the DIA is minimized, 40% more screen area becomes available to the application, as the effective screen height (or width, if in landscape mode) increases from 160 pixels to 224. How you decide to use that additional screen space is up to you -- do what makes the most sense for each individual form in your application.

Satellite Forms DIA Control Overview

By default, Satellite Forms applications will still use the square 160x160 screen area on DIA-capable devices, and the input trigger is disabled (greyed out). With the input trigger disabled, the user cannot minimize the DIA, but they can still select which DIA input configuration they want to use (eg. keyboard or virtual Graffiti, etc.). This is the way that older Satellite Forms (and other) applications that did not specifically add DIA support behaved on DIA-capable devices.

In order to support the Dynamic Input Area in SatForms PalmOS applications, a new DynamicInputArea custom control extension has been created. When this DIA control is added to your application, you gain these capabilities:

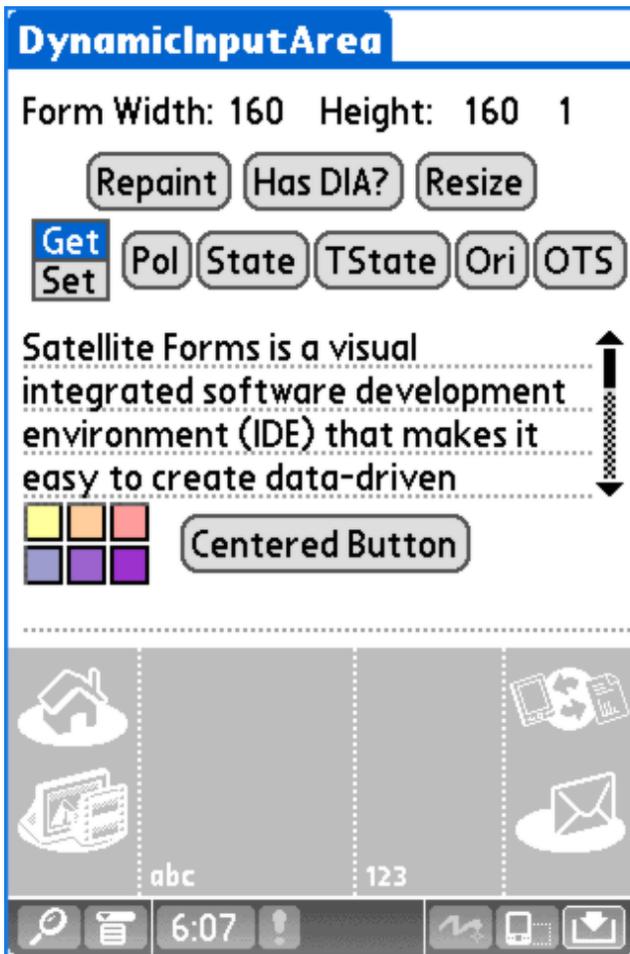
- You will receive an event (the OnClick event of the DIA control) when the size or orientation of the screen changes. This enables you to respond to the hiding or showing of the DIA, and changes in the screen orientation.
- You have the ability to query and change the DIA state (minimized or maximized) and orientation (portrait or landscape) from script code under your control, in addition to responding to the use of the input trigger and/or orientation trigger.

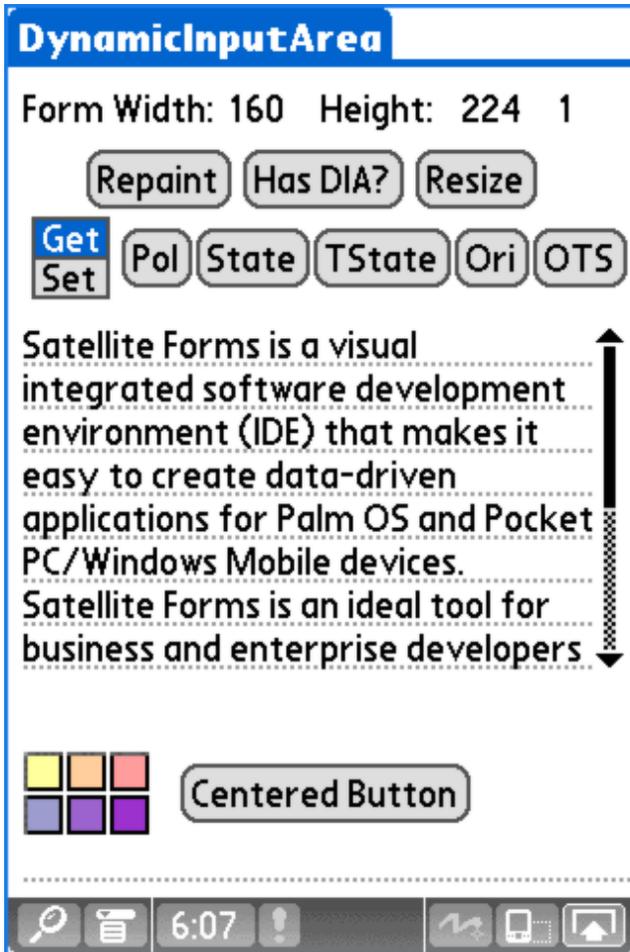
- You have the ability to disable the input trigger or orientation trigger, thus locking out the user from changing those settings.

When you receive the event informing you that the DIA state or orientation has changed, you can respond by moving or expanding controls on the form. How can you move or change the size of controls? Satellite Forms 7.0 now includes the ability to get and set the position and size of form controls at runtime through the new <control>.GetPosition and <control>.SetPosition properties, as explained in the KnowledgeBase article "[How To Move and Resize Controls at Runtime](#)".

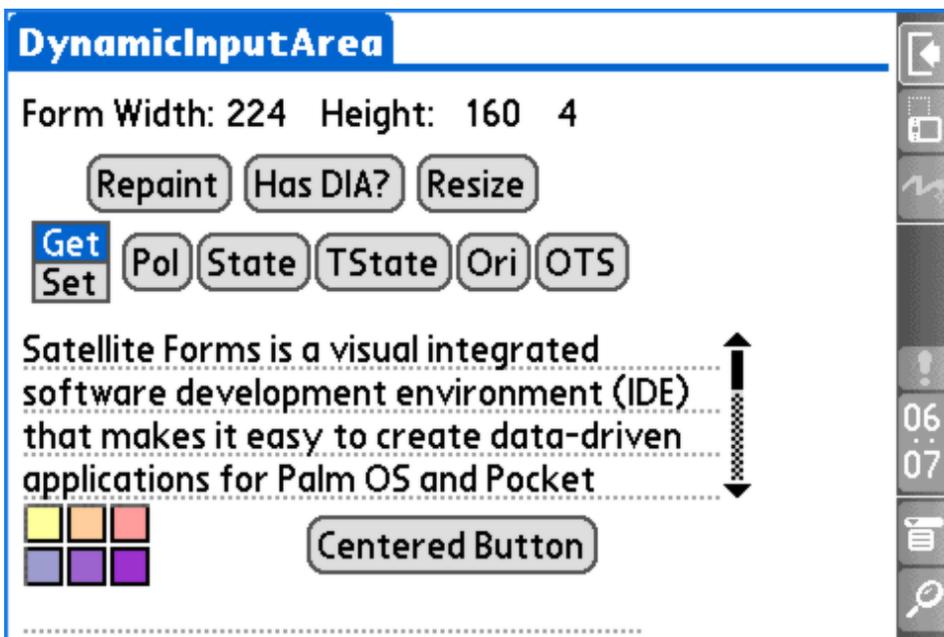
The DIA control and SetPosition/GetPosition control properties are demonstrated in the sample project "DynamicInputArea" included with Satellite Forms 7.0. Screenshots from that sample application are displayed below.

The first screenshot below shows a typical SatForms application displaying some buttons, a paragraph control, color bitmap, and edit control, with the DIA displayed (maximized). The second screenshot shows the same form after responding to the event when the user minimized the DIA by tapping on the input trigger. The form adapted to the increased screen size by moving some controls down lower on the form, and expanding the paragraph control to display more text.



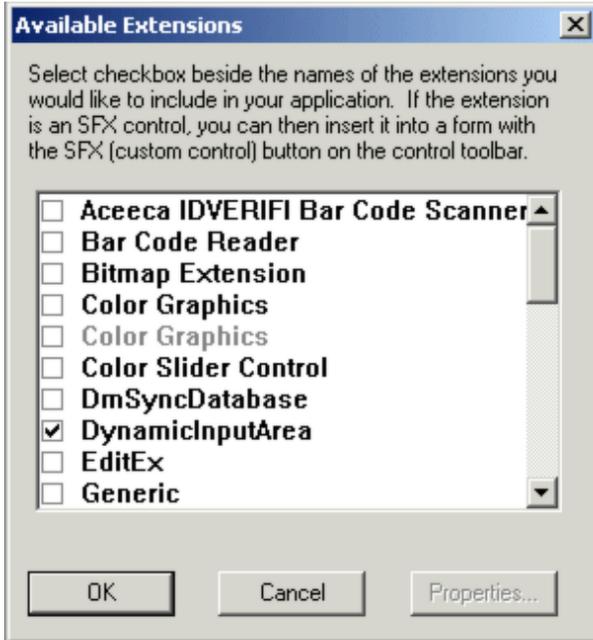


The next screenshot illustrates how the application responded to a change from portrait orientation to landscape orientation, by increasing the width of the paragraph control and shifting the centered button over to stay centered on the now wider screen.

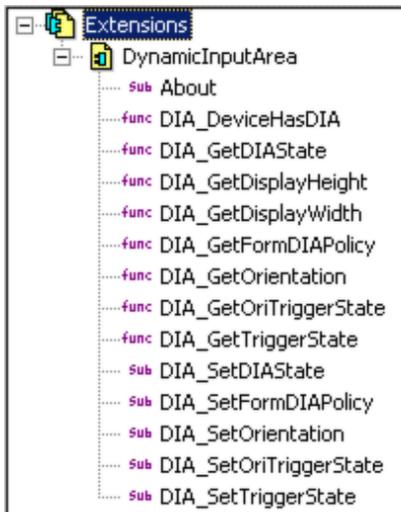


Implementing DIA Support (Technical Details)

1. In order to support the DIA in your application, you must first include the DynamicInputArea control in your project. Click on the Manage Extensions toolbar icon, and select DynamicInputArea from the list of available extensions:



2. With the DIA control added to your application, several script functions to get and set the various DIA options and states become available:



Some of these functions are intuitively named, while others might seem somewhat mysterious, so we'll try to explain them here.

The current settings of the DIA (maximized or minimized) is known as the DIA State.

PalmSource has defined these possible DIA states:

DIA State Value	Description
0	The dynamic input area is being displayed. The form is the standard 160x160 square dimensions.
1	The dynamic input area is not being displayed. The form is either tall or wide, but is not square.
2	The input area is not dynamic, or there is no input area on this device.
5	Pass this value to activate the last user-selected input area state.

The current status of the input trigger is known as the Input Trigger State, with these possible values:

Input Trigger State	Description
0	The input trigger is enabled, meaning that the user is allowed to open and close the dynamic input area.
1	The input trigger is disabled, meaning that the user is not allowed to close the dynamic input area.
2	There is no dynamic input area on this device.

The current status of the orientation trigger is known as the Orientation Trigger State, with these possible values:

Orientation Trigger State	Description
0	The orientation trigger is disabled, meaning that the user is not allowed to change the display orientation.
1	The orientation trigger is enabled, meaning that the user is allowed to change the display orientation.

The current screen orientation can have one of these possible values:

Orientation State	Description
0	Set this value to tell the system to activate the last user-selected orientation.
1	The display is in portrait (tall) orientation.
2	The display is in landscape (wide) orientation.
3	The display is in reverse portrait orientation (upside-down from the normal portrait orientation).

4	The display is in reverse landscape orientation (upside down from the normal landscape orientation).
---	--

Finally, a dynamic input area Policy specifies how the dynamic input area should be handled while a form is active. These values are possible:

Form DIA Policy	Description
0	Forces the dynamic input area to stay displayed while the form is active. The input trigger is disabled.
1	The user and the application control whether the input area is active.

Now that you understand the DIA jargon and have memorized all the possible values, we'll explain how to put that knowledge to work in your application.

3. The DIA control is a custom control extension which must be added to your forms like other custom controls such as the Symbol Integrated Scanner control. It is not visible on the device screen, but is visible on the form design window in App Designer as a control that resembles the input trigger on the status bar.

The control supports 5 control configuration settings:

DIAPOLICY = n

where n is one of the form DIA Policy values listed in the table above. In order to allow the user to control the DIA state via the input trigger on the status bar, set the DIAPOLICY=1. To effectively disable DIA support and act just like older applications, set the DIAPOLICY=0. The default value for this config option if omitted is 1. Get/Set values at runtime via script functions DIA_GetFormDIAPolicy/DIA_SetFormDIAPolicy.

DIASTATE = n

where n is one of the allowable DIA State values listed above. Setting this value to DIASTATE=5 is a good option, because that sets the form to the last user-selected DIA state. It stands to reason that if the user prefers to have the DIA closed (minimized), then they would probably prefer to keep it that way while running your application too. The default value for this config option if omitted is 5. Get/Set values at runtime via script functions DIA_GetDIAState/DIA_SetDIAState.

DIATRIGGERSTATE = n

where n is one of the allowable Input Trigger State values listed above. Set this value to DIATRIGGERSTATE=0 in order to enable the user to change the DIA state via the input trigger, or to DIATRIGGERSTATE=1 to prevent the user from changing the DIA state via the input trigger on the status bar. The default value for this config option if omitted is 1. Get/Set values at runtime via script functions DIA_GetTriggerState/DIA_SetTriggerState.

ORIENTATION = n

where n is one of the Orientation State values listed above. The default value for this config option if omitted is 0. Get/Set values at runtime via script functions DIA_GetOrientation/DIA_SetOrientation.

ORIENTATIONTRIGGERSTATE = n

where n can be 0 to disable the orientation trigger or 1 to enable the orientation trigger. The default value if omitted is 1. Get/Set values at runtime via script functions DIA_GetOriTriggerState/DIA_SetOriTriggerState.

In addition, script methods are provided to determine if the device has DIA capability, and to get the current screen width and height.

4. The DIAcontrol's OnClick event is fired whenever the display size or orientation is changed (let's call this the resize event). Use this event to handle the moving & resizing of your form controls in reaction to the display change. The sample application handles this by keeping track of the form height, width, and orientation, and comparing those values to the current values when the resize event is fired. The form's current height is obtained via the DIA_GetDisplayHeight function; the current form width is obtained with the DIA_GetDisplayWidth function. The current orientation is determined with a call to DIA_GetOrientation. If the screen has in fact been resized, then the resize script moves some controls to new locations and changes the size of some controls, as needed.

Here's an example of the script code to change the height of the paragraph control to make it taller or shorter depending on the display height:

```
'calc paragraph size
dim pgHeight
if frmHeight > 160 then
    pgHeight = 88
else
    pgHeight = 44
endif
```

```
'expand paragraph height
pgText.SetPosition(4, 80, 152 + (frmWidth - 160)/2, pgHeight )
```

Here is an example of script code to center a button control based on the form width, and to anchor it a specified number of pixels above the bottom of the form:

```
'center the button horizontally and place it 32 pixels above the bottom
btnCenter.SetPosition(44 + (frmWidth - 160)/2, frmHeight - 32, 74, 15)
```

Review all of the scripts in the DynamicInputArea sample project, and test it on a device or simulator with DIA capability, to get a good feel for the possible advantages in supporting the DIA in your application.

Conclusion

If your PalmOS application can benefit from the additional screen area available on devices with the Dynamic Input Area, it is now possible to take advantage of this capability in Satellite Forms 7 using the DynamicInputArea control and the new [.SetPosition](#) and [.GetPosition](#) control properties. Your users will appreciate the ability to use the extra screen space allowed by the expandable screen.

Keywords: DIA, dynamic, input area, virtual Graffiti, soft Graffiti, expandable, PalmOS, T3, T5, TX, LifeDrive

KB ID: 10036
Updated: 2010-06-22

[Satellite Forms Website Home](#)

-0-

How To Move and Resize Controls at Runtime

Problem: How To Move and Resize Controls at Runtime

Solution: Starting with Satellite Forms 7.0, new control methods are provided to move and resize form controls at runtime. This capability is implemented for the PalmOS and PocketPC platforms, and complements the [PalmOS Dynamic Input Area](#) support that is also new with SatForms 7.0.

The .GetPosition method retrieves the current top left X and Y coordinates, width, and height of the specified control.

GetPosition

<code>Controls(ControlName).GetPosition(cX, cY, cW, cH)</code>		
Returns the current position (cX, cY) and size (cW, cH) of a control.		
Parameters	<i>ControlName</i>	Name of a control.
	<i>cX</i>	On return, contains the top left X coordinate of the control.
	<i>cY</i>	On return, contains the top left Y coordinate of the control.
	<i>cW</i>	On return, contains the width of the control.
	<i>cH</i>	On return, contains the height of the control.
Return Value	None, the location and size values are returned in the cW, cY, cW, and cH parameter variables.	
Comments	GetPosition is a method of the Control object. This method is useful when combined with the new Dynamic Input Area support that enables you to move and resize controls on a form in response to changes in the form size or orientation.	
Example	<pre>'example of control GetPosition and SetPosition methods Dim cX, cY, cW, cH 'obtain the current location and size of Button1 Button1.GetPosition(cX, cY, cW, cH) 'move Button1 control down 10 pixels, right 10 pixels, widen by 5 pixels, increase height by 5 pixels Button1.SetPosition(cX+10, cY+10, cW+5, cH+5)</pre>	
See Also	SetPosition	

The .SetPosition method enables you to modify the current top left X and Y coordinates, width, and height of the specified control. This method allows you to both move and resize a control on the form.

SetPosition

<code>Controls(ControlName).SetPosition(cX, cY, cW, cH)</code>		
Modifies the current position (cX, cY) and size (cW, cH) of a control.		

Parameters	<i>ControlName</i>	Name of a control.
	<i>cX</i>	The new top left X coordinate of the control.
	<i>cY</i>	The new top left Y coordinate of the control.
	<i>cW</i>	The new width of the control.
	<i>cH</i>	The new height of the control.
Return Value	None.	
Comments	SetPosition is a method of the Control object. This method is useful when combined with the new Dynamic Input Area support that enables you to move and resize controls on a form in response to changes in the form size or orientation.	
Example	<pre>'example of control GetPosition and SetPosition methods Dim cX, cY, cW, cH 'obtain the current location and size of Button1 Button1.GetPosition(cX, cY, cW, cH) 'move Button1 control down 10 pixels, right 10 pixels, widen by 5 pixels, increase height by 5 pixels Button1.SetPosition(cX+10, cY+10, cW+5, cH+5)</pre>	
See Also	GetPosition	

Keywords: move, resize, control, width, height, size, location, coordinate

KB ID: 10037

Updated: 2006-10-02

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To use the SFConvertPDB utility

Problem: How To use the SFConvertPDB utility

Solution: SFConvertPDB is a commandline utility included with Satellite Forms 7.0 that enables you to convert database files from PDB format to/from DBF or MDB format, on the desktop PC. It does not require a Palm HotSync or Microsoft ActiveSync session to be active, as it runs entirely on the PC. This provides a handheld-platform-independent PC based mechanism to convert data to & from PDB files, for use with Satellite Forms applications on the PalmOS platform and on the [PocketPC platform when using PalmDB database tables](#).

Updated July 11, 2007 to include new -CreateFlag parameter and table flag details.

A. Overview

SFConvertPDB enables you convert Satellite Forms table data from:

- PDB table to dBase DBF table
- PDB table to MS Access MDB table
- dBase DBF table to PDB table
- MS Access MDB table to PDB table

The SFConvertPDB utility does not make any assumptions or have any requirements about how you get the PDB database file to the PC, whether it be via PalmOS HotSync, Microsoft ActiveSync, TCPIP, Bluetooth file transfer, memory cards, server synchronization tools, etc. It does not deal with the movement of the PDB file to/from the PDA; rather it deals with converting the data between PDB <--> MDB/DBF on the desktop PC only. This makes it very flexible, and enables it to be integrated into a PDA synchronization system regardless of how that system moves the PDB file back and forth from the PDA to the PC.

SFConvertPDB is an EXE executable program, rather than an ActiveX control or DLL. To integrate SFConvertPDB into your synchronization system, your sync application must be capable of launching an EXE application.

SFConvertPDB operates via commandline switches, in which you supply the required information about which database file to convert, and how to convert it. If you are familiar with the use of the SatForms HotSync OCX control to move data back and forth at HotSync, you will find most of these parameters to be familiar. If you have used the RDKInst or CeRDKInst utilities for commandline installation of files and/or conduit registration, you will find the use of the SFConvertPDB utility familiar as well.

B. Usage

Usage: SFConvertPDB [*commandline switches*]

-[PDBtoPC|PCtoPDB]: Specifies whether you want to convert from a PDB database to a DBF/MDB database [use -PDBtoPC], or from a PC DBF/MDB database to a PDB [use -PCtoPDB]. You may specify only one of these conversion options, not both.

-filename \path\to\PC_database_file.[DBF|MDB]: You must supply the path and filename of the PC DBF or MDB database file which you wish to convert to or from PDB. You do not specify the PDB file even when converting from PDB to DBF/MDB: rather, you always supply the PC database filename ending in .DBF or .MDB.

-creatorid CRID: Where CRID is the four-character unique creatorID used by your application (and which should be registered with PalmSource via their creatorID registry on their website), as defined in the Project Properties settings screen in App Designer. You must supply the correct case sensitive characters, which cannot include spaces. When testing with the SatForms SDK runtime engine used by App Designer, the creatorid is SMSF. This parameter is needed in order to generate the correct PDB table name for conversion, as the creatorid is incorporated into the PDB table name. The default creatorid if none is supplied is SMSF.

-SDDI_DLL DLLfilename: You must specify the correct SatForms SDDI DLL to perform the conversion. At the present time, the only conversion supported is for SatForms PalmDB (PDB) databases, so you should always specify the SDDI_PalmDB.DLL. This parameter is optional, and if you omit it, the default of -SDDI_DLL SDDI_PalmDB.DLL will be used.

-VersionMajor VV: Where VV is the major version number of your application, as defined in the Project Properties settings screen in App Designer. The allowable values are from 0 - 99. This parameter is needed in order to generate the correct PDB table name for conversion, as the major and minor version numbers are incorporated into the PDB table name. This parameter is optional, and if you omit it, the default value of 0 will be used.

-VersionMinor vv: Where vv is the minor version number of your application, as defined in the Project Properties settings screen in App Designer. The allowable values are from 0 - 99. This parameter is needed in order to generate the correct PDB table name for conversion, as the major and minor version numbers are incorporated into the PDB table name. This parameter is optional, and if you omit it, the default value of 0 will be used.

-CreateFlag n: Where n specifies the desired table flag[s] to set on the PDB table when it is created by SFConvertPDB. The allowable values are any positive integer, corresponding to the combination of desired table flags. Certain PDB table behaviours can be set via this numeric flag value, including Backup, Read-Only, and NoAutoCommit flags. The table flags cause the SatForms runtime engine to treat the table differently on the PDA. For a more detailed reference about table flags, see section G: Table Flags below. This new parameter was added with SFConvertPDB version 7.0.1.040 and higher and is not supported in previous versions. This parameter is optional, and if you omit it, the default value of 0 will be used.

-quiet: This parameter instructs SFConvertPDB to not display any popup error messages when performing the conversion. The exit code of SFConvertPDB can be queried to determine if the conversion was successful or not, making it suitable for calling from sync applications or batch files. When run interactively, you would not likely use the -quiet switch so that you could see error messages pop up if there are conversion problems.

Example usage for common conversion scenarios:

1. Converting the database file C:\MyApp\Data\EMyAp0102_MYTABLE1.PDB to C:\MyApp\Data\MyTable.DBF:

```
SFConvertPDB -PDBtoPC -filename C:\MyApp\Data\MyTable.DBF - creatorID MyAp -VersionMajor 1 -VersionMinor 2
```

Result: the C:\MyApp\Data\MyTable.DBF is created containing the records from EMyAp0102_MYTABLE1.PDB.

2. Converting the database file C:\MyApp\Data\EMyAp0102_MYTABLE1.PDB to C:\MyApp\Data\MyTable.MDB:

```
SFConvertPDB -PDBtoPC -filename C:\MyApp\Data\MyTable.MDB - creatorID MyAp -VersionMajor 1
```

-VersionMinor 2

Result: the C:\MyApp\Data\MyTable.MDB is created containing the records from EMyAp0102_MYTABLE1.PDB.

3. Converting the database file D:\Server\Data\tNames.DBF to

D:\Server\Data\EMyAp0200_TNAMES.PDB:

```
SFConvertPDB -PCtoPDB -filename D:\Server\Data\tNames.DBF - creatorID MyAp -VersionMajor 2  
-VersionMinor 0
```

Result: the D:\Server\Data\EMyAp0200_TNAMES.PDB is created containing the records from tNames.DBF.

4. Converting the database file D:\Server\Data\tNames.MDB to

D:\Server\Data\EMyAp0200_TNAMES.PDB:

```
SFConvertPDB -PCtoPDB -filename D:\Server\Data\tNames.MDB - creatorID MyAp -VersionMajor 2  
-VersionMinor 0
```

Result: the D:\Server\Data\EMyAp0200_TNAMES.PDB is created containing the records from tNames.MDB.

5. Converting the database file D:\Server\Data\tNames.MDB to

D:\Server\Data\EMyAp0200_TNAMES.PDB with a -CreateFlag value of 64 to indicate NoAutoCommit table flag:

```
SFConvertPDB -PCtoPDB -filename D:\Server\Data\tNames.MDB - creatorID MyAp -VersionMajor 2  
-VersionMinor 0 -CreateFlag 64
```

Result: the D:\Server\Data\EMyAp0200_TNAMES.PDB is created containing the records from tNames.MDB and has the NoAutoCommit table flag set.

6. Converting the database file D:\Server\Data\tNames.MDB to

D:\Server\Data\EMyAp0200_TNAMES.PDB with a -CreateFlag value of 7 to indicate Backup + Read-Only + Autoname table flags:

```
SFConvertPDB -PCtoPDB -filename D:\Server\Data\tNames.MDB - creatorID MyAp -VersionMajor 2  
-VersionMinor 0 -CreateFlag 7
```

Result: the D:\Server\Data\EMyAp0200_TNAMES.PDB is created containing the records from tNames.MDB and has the Backup + Read-Only + Autoname table flags set.

C. Integration into a PocketPC synchronization system using PocketPC PDB tables

Starting with Satellite Forms 7.0, it is possible to [use PalmDB \(PDB\) tables with PocketPC applications](#), and there are several advantages in doing so instead of using Microsoft Compact Database CDB format tables. In order to build a successful PocketPC data synchronization system when using PDB tables, integrated with your desktop PC database system, you need to be able to handle these four actions:

- conversion from your PC database table to a PDB database table
- transferring a PDB table from the PC to the PDA (download data to the PDA)
- transferring a PDB table from the PDA back to the PC (upload data to the PC)
- conversion from a PDB table to your desktop PC database table

The SFConvertPDB utility provides the means to convert PDB <--> DBF/MDB on your PC. The action of transferring PDB files between the PDA <--> PC can be achieved in multiple ways. Two of those file transfer methods include using the Satellite Forms ActiveSync OCX, or using the Satellite Forms CeRemote.DLL.

When using the SatForms ActiveSync OCX, the functions FileGetFromPPC and FileSendToPPC can be utilized to transfer PDB tables to/from the PocketPC. This is in contrast to the DatabaseFromPPC and DatabaseToPPC functions that are used with CDB database tables.

A typical data synchronization flow using the SF ActiveSync OCX to retrieve a PDB database table from the PocketPC, then send an updated PDB table back to the PPC, would go something like this:

- retrieve the PDB table from the PPC using FileGetFromPPC
- convert the PDB table to MDB/DBF using SFConvertPDB
- manipulate that data in your PC database, creating an updated DBF/MDB
- convert the updated DBF/MDB table to PDB using SFConvertPDB
- download that updated PDB to the PPC using FileSendToPPC

When using the SatForms CeRemote.DLL to transfer files, the functions GetFile and SendFile can be utilized to transfer PDB tables to/from the PocketPC. This is in contrast to the GetTable and SendTable functions that are used with CDB database tables.

A typical data synchronization flow using the CeRemote.DLL to retrieve a PDB database table from the PocketPC, then send an updated PDB table back to the PPC, would go something like this:

- retrieve the PDB table from the PPC using GetFile
- convert the PDB table to MDB/DBF using SFConvertPDB
- manipulate that data in your PC database, creating an updated DBF/MDB
- convert the updated DBF/MDB table to PDB using SFConvertPDB
- download that updated PDB to the PPC using SendFile

A sample PocketPC synchronization tool written in Visual Basic 6 called SatSyncPPCPDB is provided in the Samples folder of the Satellite Forms installation. This SatSyncPPCPDB sample application demonstrates the use of the CeRemote.DLL and SFConvertPDB utility to sync PDB data back and forth between the PC and PocketPC.

D. Integration into a PalmOS synchronization system

Satellite Forms has used PalmDB (PDB) format database tables for many generations, and synchronization between the PalmOS handheld and PC DBF/MDB databases has been enabled through the Satellite Forms HotSync conduit and the Satellite Forms HotSync OCX. The SatForms HotSync OCX provides functions for sending PC database data to PDB tables on the PDA, and retrieving PDB data from the PDA to the PC database tables at HotSync, via the

CopyTableToPalmPilot and GetTableFromPalmPilot functions. For most SatForms developers, this synchronization system is ideal, and no other options are needed.

However, some SatForms PalmOS application developers need to synchronize data without relying on Palm HotSync as the data transfer mechanism. Palm HotSync does have its limitations, not the least of which is that it only supports the synchronization of a single Palm device at a time: simultaneous HotSync synchronization of multiple devices is not possible on a single PC. It is for that reason that the SFConvertPDB utility may be of interest to SatForms PalmOS application developers in addition to SatForms PocketPC PDB app developers.

Let's consider an example using the File Transfer Protocol (FTP) to transfer PDB database files back and forth between a TCPIP-connected PalmOS device (for example a Palm Treo smartphone or TX PDA) and a PC database server. A typical data synchronization flow using FTP to transfer the PDB files would go something like this:

- upload the PDB table from the PDA to PC server using FTP (for example with the PalmDataPro SF-FTP extension)
- convert that PDB table on the server to DBF/MDB using the SFConvertPDB utility
- manipulate that data in your PC database, creating an updated DBF/MDB
- convert the updated DBF/MDB table to PDB using SFConvertPDB
- download that updated PDB table to the Palm PDA using FTP (via SF-FTP extension)

E. Installation of components needed by SFConvertPDB

SFConvertPDB relies on several Satellite Forms components included in the Satellite Forms Runtime for PalmOS (not to be confused with the SatForms PalmOS runtime engine). Installing the Runtime package is the easiest way to ensure that all of the necessary components needed by SFConvertPDB are installed and registered correctly. See the article [How To Install the SatForms Runtime for Palm silently](#) for tips on how to install the runtime without any installation UI on end users' computers. Note that this runtime needs to be installed on any computer that will use the SFConvertPDB utility, regardless of whether that computer syncs with PalmOS devices, and/or PocketPC devices, or does not perform any device synchronization at all.

F. SFConvertPDB Error Codes

Error Code	Error Name	Description
0	appErrNone	no error
-1	appErrOleInitFailed	OLE is needed to use MDB. Somehow MFC failed to init OLE.
-2	appErrInvalidSwitch	unrecognized switch
-3	appErrCreatorIDBadLength	creator ID must be exactly 4 chars
-4	appErrUnknownParserState	bad logic, should not have ended up in a bad parser state
-5	appErrFailToLoadSDDI	failed to load SDDI plugin
-6	appErrSDDIPluginNoAPI	this sddi plugin is missing APIs
-7	appErrSDDIPluginBadVersion	plugin is too new or too old
-8	appErrSDDICantInstantiate	calling new fails

-9	appErrGenericError	unknown exception when calling SDDI DLL
-10	appErrConvertException	sfddb exception when calling SDDI DLL
-11	appErrConvertError	SDDI API fails
-12	appErrConversionTypeNotSpecified	user did not specify -pctopdb or -pdbtopc
-13	appErrFilenameNotSpecified	user did not specify filename
-14	appErrConversionTypeAmbiguous	user specified both -pctopdb and -pdbtopc
-15	appErrFilenameNotExist	MDB or DBF file to convert does not exist
-16	appErrPDBFilenameNotExist	PDB file to convert does not exist

G. Table Flags

One of the properties of a Satellite Forms table PDB is the Table Flag value. The Table Flag is a numeric value that determines special behaviours of that table when it is in use on the PDA by the Satellite Forms runtime engine. For example, one of the possible table flag values indicates that the table is Read-Only, and the runtime engine therefore prevents any modifications/additions/deletions to the data in the table.

The Table Flags for each table PDB can be specified using the -CreateFlag parameter of the SFConvertPDB utility. Each table can have different flags as they are assigned on a per-table basis. The current supported table flags include:

Flag value	Flag name	Flag description
0	No table flags	No special table behaviours - regular table
1	Backup	The table will be backed up at Hotsync (PalmOS behaviour only)
2	Read-Only	The runtime engine will prohibit table modifications/additions/deletions and will only permit read access to the table data (PocketPC runtime engine will close read-only tables more quickly than read/write tables)
4	Autoname	The desktop table name will automatically match the logical table name (Link table name to filename option in App Designer table editor)
64	NoAutoCommit	The table data that is cached in memory while the application is open will not be automatically committed to storage when the app closes. To save the data before closing, you must call the CommitDatabase function explicitly for each NoAutoCommit table. This flag is implemented on the PocketPC platform only, and is ignored on the PalmOS platform. The NoAutoCommit flag enables you to allow tables to close more quickly by not automatically committing them to storage when the app closes, like a read-only table.

Some table flag values can be combined, while others must be exclusive. For example, the Backup, Read-Only, and Autoname flags can all be present on a given table, and you would specify all of those flags by adding their flag values together ([Backup] 1 + [Read-Only] 2 + [Autoname] 4 = 7). The Read-Only and NoAutoCommit flags must be exclusive to each other:

do not combine them together.

H. Conclusion

The SFConvertPDB utility is a flexible tool that can be used as an important component in a PalmOS or PocketPC PDB data synchronization system. Because SFConvertPDB operates on the PC and does not rely on any specific PDA <--> PC data transport layer or protocol, it can be integrated into a wide variety of synchronization scenarios including PalmOS HotSync, Microsoft ActiveSync, TCPIP, Bluetooth file transfer, memory cards, server synchronization protocols, and more.

Keywords: SFConvertPDB, convert, conversion, PDB, CDB, HotSync, ActiveSync, sync, synchronization, transfer, TCPIP, FTP, DBF, MDB, database, CreateFlag, flag, NoAutoCommit, Read-Only

KB ID: 10038

Updated: 2007-07-11

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To use SatSyncPPC to sync PocketPC data

Problem: How To use SatSyncPPC to sync PocketPC data

Updated: Starting with Satellite Forms 7.1, the SatSyncPPC and SatSyncPPCPDB sample desktop sync projects for PocketPC applications have been merged together into a single SatSyncPPC target, which supports syncing desktop data with PDB databases on the PocketPC, using either the Satellite Forms ActiveSync OCX or the CeRemote.dll non-ActiveX approach. The information below is applicable to the SatSyncPPC sample included with Satellite Forms 7.1.

Solution: SatSyncPPCPDB is a sample application written in Visual Basic and provided in the \Samples folder of your SatForms installation. SatSyncPPCPDB demonstrates how to send and receive data between the desktop PC and PocketPC handheld, using the Satellite Forms CeRemote.dll and [SFConvertPDB](#) utility. This sample is designed for [PocketPC applications using PalmDB \(PDB\) format databases](#) on the handheld. To see a related example based on the SatForms ActiveSync OCX and the use of CDB handheld database tables, see [How To use SatSyncPPC to send data to the PocketPC device](#).

SatSyncPPCPDB is not only a valuable tool to help you learn how to create your own PocketPC synchronization system, it is also a very useful utility in its own right for developers and perhaps even end users of your application.

SatSyncPPCPDB demonstrates these following PocketPC synchronization tasks:

- retrieve a PDB table from a PocketPC
- convert a PDB table to MDB/DBF
- convert a DBF/MDB table to PDB
- download a PDB to the PocketPC

These instructions assume that you have created and compiled your application in App Designer, and want to send data from an existing database table on the PC to the handheld. It is also possible to bring this data back into the table editor in App Designer with an additional step. If your data already exists in the App Designer table editor, there is no need to follow these steps, since you can just use the Handheld | Download App & Tables function. These instructions are for sending data from an existing PC database to your SatForms application on the handheld, and optionally bringing that data back into the App Designer table editor.

1. Start the SatSync sample application in \Samples\SatSyncPPCPDB.
2. Select File | New, then File | Configure Send List.
3. Select the DBF or MDB files to send data to the handheld.
4. Specify the correct creatorID and version numbers for your application, matching the settings in the Project Properties settings in App Designer.
5. Specify the folder on the PocketPC to send the database files to. Generally this will be something like \My Documents\MyApp. SatSyncPPCPDB will create this folder if it does not exist when the files are transferred.
6. Save your SatSync config file, then click on Send Tables to PocketPC. If the PocketPC device is not currently connected, an error message is displayed. Otherwise, the data should get sent to the handheld, where it now resides in the handheld tables in the specified folder. SatSyncPPCPDB will display the file transfer results in the status field on the main form.

To bring the handheld data back to a desktop database file, you would Configure Get List, and then initiate the transfer by clicking on the Get Files from PocketPC button.

Another option is to bring data back into the App Designer Table Editor. Simply start App Designer, select Handheld | Upload Tables and the data is brought back into the Table Editor.

SatSyncPPCPDB performs the tasks of transferring PDB database files between the PC <--> PDA using the Satellite Forms CeRemote.dll. The functions of this DLL are declared in Module1 (SF_PPC_API.bas). The reference documentation for this DLL is the CeRemoteAPI.h header file located in the \Satellite Forms 7\Include folder.

The conversion between PDB <--> DBF/MDB is carried out on the desktop PC by calling the functions in the SFConvertPDB utility. SatSyncPPCPDB calls SFConvertPDB to perform the conversion, and checks the return value to determine if the conversion succeeded or failed.

The combination of the database file transfer and database file conversion functions provides the ability to create a complete data synchronization system for Satellite Forms PocketPC PDB applications.

Keywords: SatSync, SatSyncPPCPDB, PocketPC, PDB, VB, sample, sync, synchronization, SFConvertPDB, DBF, MDB, database

KB ID: 10039

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Change Control Fonts at Runtime

Problem: How To Change Control Fonts at Runtime

Solution: Starting with Satellite Forms 7.0, a new control property is provided to change control fonts at runtime. This capability is implemented for the PalmOS and PocketPC platforms.

Font

Controls(ControlName).Font																													
Returns or sets the current FontID of a control.																													
Parameter	<i>ControlName</i>	Name of a control.																											
Return Value	The current FontID of the control, an integer value from 0..7 (PalmOS) or 0..3 (PocketPC) corresponding to this table of FontIDs:																												
	<table border="1"> <thead> <tr> <th>Font ID</th> <th>Font Description PalmOS</th> <th>Font Description PocketPC</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Normal 9</td> <td>Tahoma 8</td> </tr> <tr> <td>1</td> <td>Bold 9</td> <td>Tahoma 8 Bold</td> </tr> <tr> <td>2</td> <td>Normal 12</td> <td>Tahoma 10</td> </tr> <tr> <td>3</td> <td>Symbol 9</td> <td>Tahoma 10 Bold</td> </tr> <tr> <td>4</td> <td>Symbol 11</td> <td>n/a</td> </tr> <tr> <td>5</td> <td>Symbol 7</td> <td>n/a</td> </tr> <tr> <td>6</td> <td>LED</td> <td>n/a</td> </tr> <tr> <td>7</td> <td>Bold 12</td> <td>n/a</td> </tr> </tbody> </table>		Font ID	Font Description PalmOS	Font Description PocketPC	0	Normal 9	Tahoma 8	1	Bold 9	Tahoma 8 Bold	2	Normal 12	Tahoma 10	3	Symbol 9	Tahoma 10 Bold	4	Symbol 11	n/a	5	Symbol 7	n/a	6	LED	n/a	7	Bold 12	n/a
Font ID	Font Description PalmOS	Font Description PocketPC																											
0	Normal 9	Tahoma 8																											
1	Bold 9	Tahoma 8 Bold																											
2	Normal 12	Tahoma 10																											
3	Symbol 9	Tahoma 10 Bold																											
4	Symbol 11	n/a																											
5	Symbol 7	n/a																											
6	LED	n/a																											
7	Bold 12	n/a																											
Comments	Font is a property of the Control object. You can change a control's font property by setting this property to one of the specified FontID values. In that case, the function does not return a value.																												
Example	<pre>'Example of Font property 'InputA is an edit control. 'change InputA font to Bold If InputA.Font = 0 Then InputA.Font = 1 EndIf</pre>																												
See Also	SetPosition																												

Keywords: Font, control, normal, bold, Tahoma, FontID

KB ID: 10041

Updated: 2006-10-02

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

How To Enable a User to Interrupt a Closed Loop

Problem: How To Enable a User to Interrupt a Closed Loop

Solution: Under normal circumstances, a For..Next loop or While..Wend loop cannot be interrupted until it has completed. This is known as a closed loop. Sometimes, however, it may be advantageous to enable a user to interrupt a closed loop in your application, allowing them to stop a potentially lengthy process.

It is in fact possible to interrupt a closed For..Next loop, using a heretofore undocumented method. Using this method, a pen tap can be used to interrupt the loop. No timer function is required.

The key is to check the pen status in your loop just before the Next statement. For example, if you add this to your loop just before the Next j:

```
For j ...  
...  
'check for pen tap to interrupt loop  
if GetPenStatus(x,y) = true then Exit For  
Next j
```

That enables the user to tap anywhere on the screen to interrupt the loop. The script then jumps to the statement immediately following the Next statement.

If you want to restrict the valid screen area that the user can tap on to interrupt the loop (for example on a Stop button), then further qualify your GetPenStatus check with the x/y pen coordinates. For example, let's say we have a STOP button on the screen with the rectangle bounds 0,120,160,160. You could just allow taps in that rectangle (actually just the y range 120..160 because the x ranges from 0..160) to stop the loop:

```
For j ...  
...  
'check for pen tap on Stop button to interrupt loop  
if (GetPenStatus(x,y) = true) and ((y >= 120) and (y<=160)) then Exit For  
Next j
```

This same technique can apply to While..Wend loops. This technique is applicable to both the PalmOS and PocketPC platforms.

Power Tip: Script Debugging Aid

The GetPenStatus function can also be used for script debugging as a pseudo breakpoint, for times when you want to pause a script without displaying a msgbox or using a delay statement. You can pause the script until the screen is tapped, and it then resumes from there.

```
An example would be:  
'doing stuff before this, want to pause now  
dim pX,pY  
while GetPenStatus(pX,pY) = false  
wend  
'script resumes when the screen is tapped
```

The device will sit there in the endless While..Wend loop until you tap the screen.

Keywords: loop, interrupt, pause, GetPenStatus, debug, for, next, while, wend

KB ID: 10042
Updated: 2006-10-02

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Bundle the SatForms PocketPC runtime engine with your app

Problem: How To Bundle the SatForms PocketPC runtime engine with your app

Solution: Satellite Forms 7 introduces a new way to bundle the PocketPC runtime engine with your application, enabling you to create app installations that are more simple, robust, and professional. This new improved installation approach is supported only when using the [PDB device database format on PocketPC](#), and is not available when using CDB device database files.

Updated for Version 8: With Satellite Forms 8 there is a new capability to use an integrated runtime engine, which removes the need to install the runtime engine SatForms80.exe file separately from the application EXE file.

The previous approach, still supported in SatForms 7 and required when using CDB database files, was to install the SF PocketPC runtime engine centrally in the \Windows folder on the PDA, and use that runtime engine for all of the SatForms apps installed on the device. A runtime engine installer is included in SatForms for this purpose. This approach does work fine if all of the SatForms apps on the device are created using the same version of Satellite Forms, but can be problematic when there are apps written with different versions of Satellite Forms.

The new recommended approach in SatForms 7 is to bundle the required SatForms runtime engine files directly with your application files, together in your application folder. Using this new approach, your installation becomes more robust, as your app will not be affected by other older SatForms applications being installed on the same PDA. Your installation can be more simple, because you can now use practically any method available to install your complete app onto the device, including simply dragging and dropping a folder from the desktop to the PDA, using the SatForms CeRdkInst utility, installing from a CAB file, or using a full installer like that created by EzSetup, PocketSetupInstaller, NSIS, InstallShield, etc. Your installation becomes more professional, because the Satellite Forms runtime engine is not listed separately in the Remove Programs list on the device, and so it presents a more complete, unified appearance to the end user.

How It Works

To make the bundled runtime approach work, you must do two things. First, you must create an icon for your application as per the instructions in the Satellite Forms help, so that when you compile your PocketPC application, both a .PDA and .EXE file are created. If you do not create an icon for your project, the .EXE file will not be created, and the bundled runtime approach will not work. Second, you must copy the SatForms70.exe and DvSddi_PPDPDB.dll files from the \Satellite Forms 7\Runtime\PocketPC folder to your application folder on the PocketPC.

When the user taps on the app.EXE file to launch your app, it will search for the SatForms70.exe file in the app folder. If the SatForms file is found, then it is used to directly run your application.

If the SatForms file is not found, it will then try to launch your app via the PDA file, using the app that is registered to handle the PDA files (the SF runtime installer registers the runtime engine to handle PDA files when it is installed -- this is the approach used in previous versions).

We've added another article entitled [How To Create an Installer for your SatForms 7 PocketPC Application](#) that describes the complete step by step approach for creating a single-EXE installer for your SatForms 7 PocketPC PDB application using the bundled runtime method. Following this approach, you can create a simple, clean, professional looking installer for your PocketPC

application.

Keywords: bundle, runtime, PocketPC, Windows Mobile, install, setup, deployment

KB ID: 10043

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Create an Installer for your SatForms 7 PocketPC Application

Problem: How To Create an Installer for your SatForms 7 PocketPC Application

Solution: There are several different approaches to creating an installer for your SatForms PocketPC application. Satellite Forms 7 introduces a new and improved capability to [bundle the SatForms runtime engine files directly with your application](#) files for an optimal end user experience. This article presents an approach that generates a single exe file to install your compiled application and bundled Satellite Forms runtime engine to the PocketPC, providing a simple, robust, and professional installation.

Updated for Version 8: With Satellite Forms 8 there is a new capability to use an integrated runtime engine, which removes the need to install the runtime engine SatForms80.exe file separately from the application EXE file.

This example describes all steps needed to create an installer for the PocketPC application "Work Order Sample.exe", based on the SF sample project "Work Order". Directories and program names can be modified to suit your own application by changing the references in the files described below. The process described herein will enable you to create a single-EXE installer file for your SatForms 7 PocketPC application. This approach requires that the PocketPC application uses the PDB device database format, and not CDB database files. The single installer created by this process is compatible with PPC 2002, 2003, 2003SE, and Windows Mobile 5 devices.

The overall approach is to create a PocketPC CAB file for your application with the bundled SatForms runtime engine, and then convert that CAB file into a single EXE installer.

NOTE: Do not let the length of this article dissuade you from using this installer creation process. This article is quite long and liberally illustrated so as to ensure no steps are left out, but you will find it is a straightforward process to create you own app installers using this method, and the professional looking results are worth the investment of your time.

This installer creation method utilizes a freeware PocketPC installer creation tool called Pocket PC Installation Creator, from Aperitto Software. There are many other installer creation tools that could be used including free tools such as NSIS, Inno Setup, PocketSetupCreator, and EzSetup, as well as numerous commercial tools such as InstallShield, Wise Installer, e-PocketSetup, and PocketPC Installer. Pocket PC Installation Creator handles the difficult task of correctly creating a CAB file, with no manual editing of INI files, batch files, or anything like that. It presents an easy wizard interface to enter all of the required information, and then generates the CAB and EXE files for you!

A. What does the installer do?

It implements a complete install program for the your PocketPC application. The install program "WorkOrder_Setup.exe" in this example does the following (note that the PPC must be connected to the desktop and ActiveSync must be running):

1. Displays a readme notice and then license agreement (EULA) which the user must accept to proceed (you supply the text).
2. Installs the application "Work Order Sample.exe", the bundled SF7 runtime engine files and all *.PDB files into the "\\Program Files\\Work Order" folder on the PPC.
3. Creates a shortcut for the program in "\\Windows\\Start Menu\\Programs" on the PPC.
4. Registers the program and creates an "Unload" file on the PPC for future deletion of the application.

B. Files required to create the installer

1. Pocket PC Installation Creator, from Aperitto Software

This freeware tool from Aperitto Software is available in our Support Files section at http://www.satelliteforms.net/support/ppc_installation_creator_setup.exe (the original author's URL is no longer valid).

Install the Pocket PC Installation Creator tool and accept all of the default install options.

2. Application files: Work Order Sample.exe (icon/executable for PPC), Work Order Sample.pda, all *.PDB files in the AppPkg folder (ESMS20000_WRKLOOKUP.PDB, ESMS20000_WRKSITES.PDB, ESMS20000_WRKWORKITEMS.PDB), and the SatForms70.exe and DvSddi_PPDPDB.dll runtime engine files. Create the PDA, EXE, and PDB files by compiling the PocketPC PDB target of the Work Order sample project included with Satellite Forms 7. Copy the SatForms70.exe and DvSddi_PPDPDB.dll files from the \Satellite Form 7\Runtime\PocketPC folder to the location where the other application files are stored (in this example, they are all copied to a C:\My Documents\WorkOrderInstall\PocketPC PDB folder.

3. Readme and EULA text files:
setup_readme.txt, eula.txt

Create these two text files as needed for your application. In our example, the setup_readme.txt file contains this text:

The SampleCo Work order application will be installed to your Windows Mobile/PocketPC PDA. After the installation has completed, you can run the Work Order application by tapping on Start | Programs | Work Order.

The sample EULA.txt file contains this text:

End user License Agreement

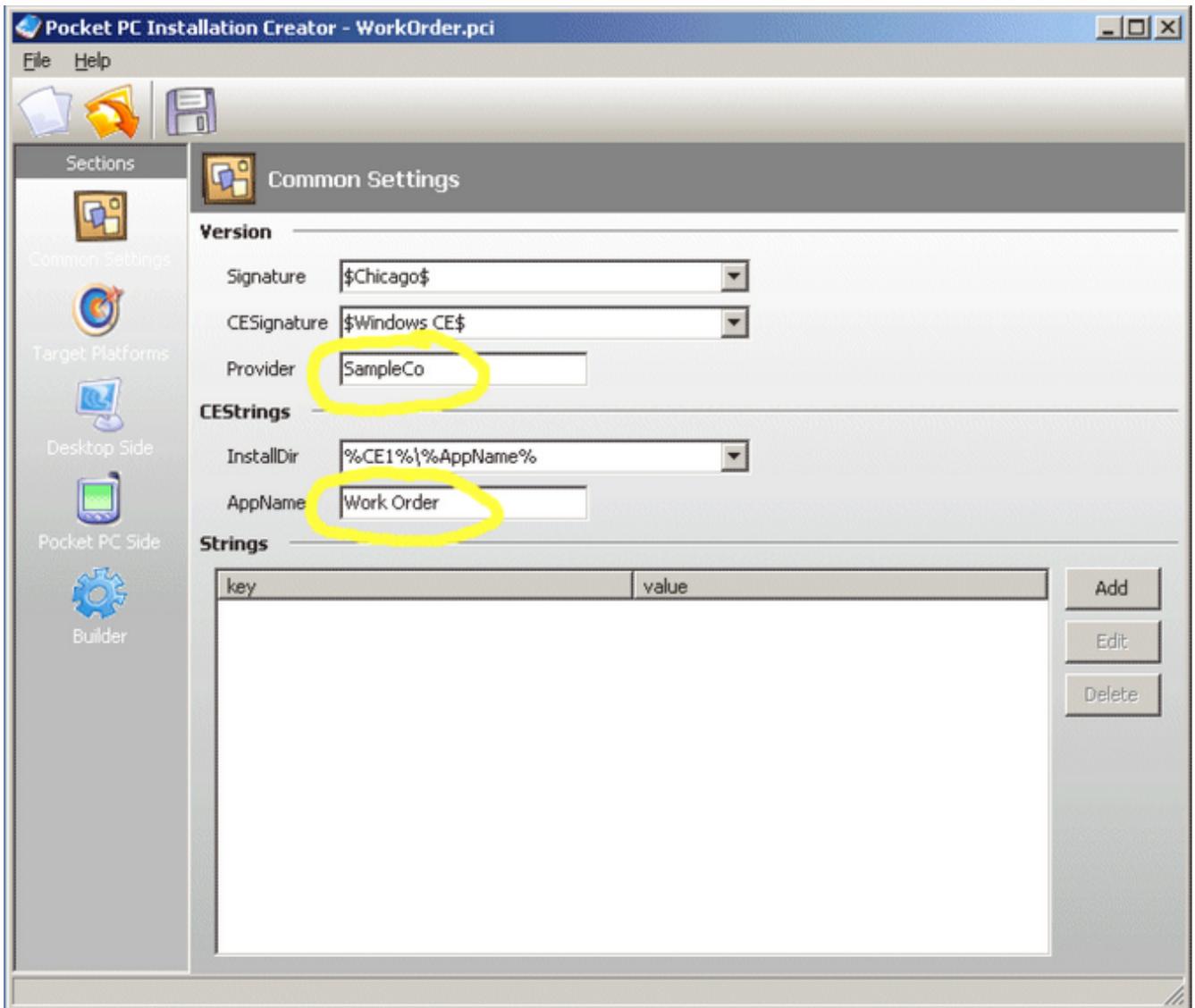
Insert lawyer-approved text here...

Copyright (C) 2006 SampleCo Corporation

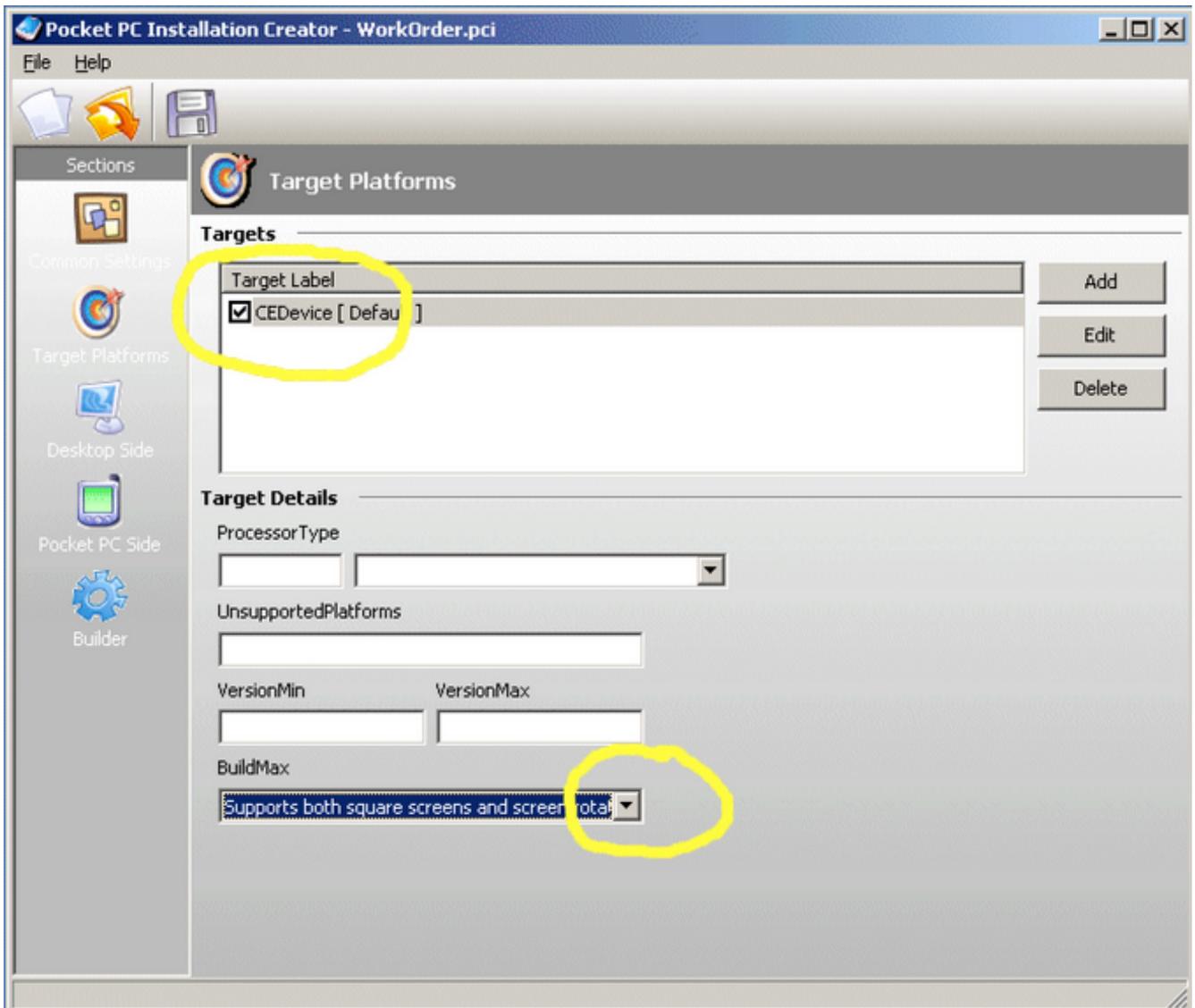
C. Step By Step Process

C.1 Start the PocketPC Installation Creator application. It will open to a blank screen that says "No project is loaded. Please open or create a new one.". Click on the new project icon in the toolbar (the white paper icon). Enter the name of your installer project, for this example enter WorkOrder and click on Save.

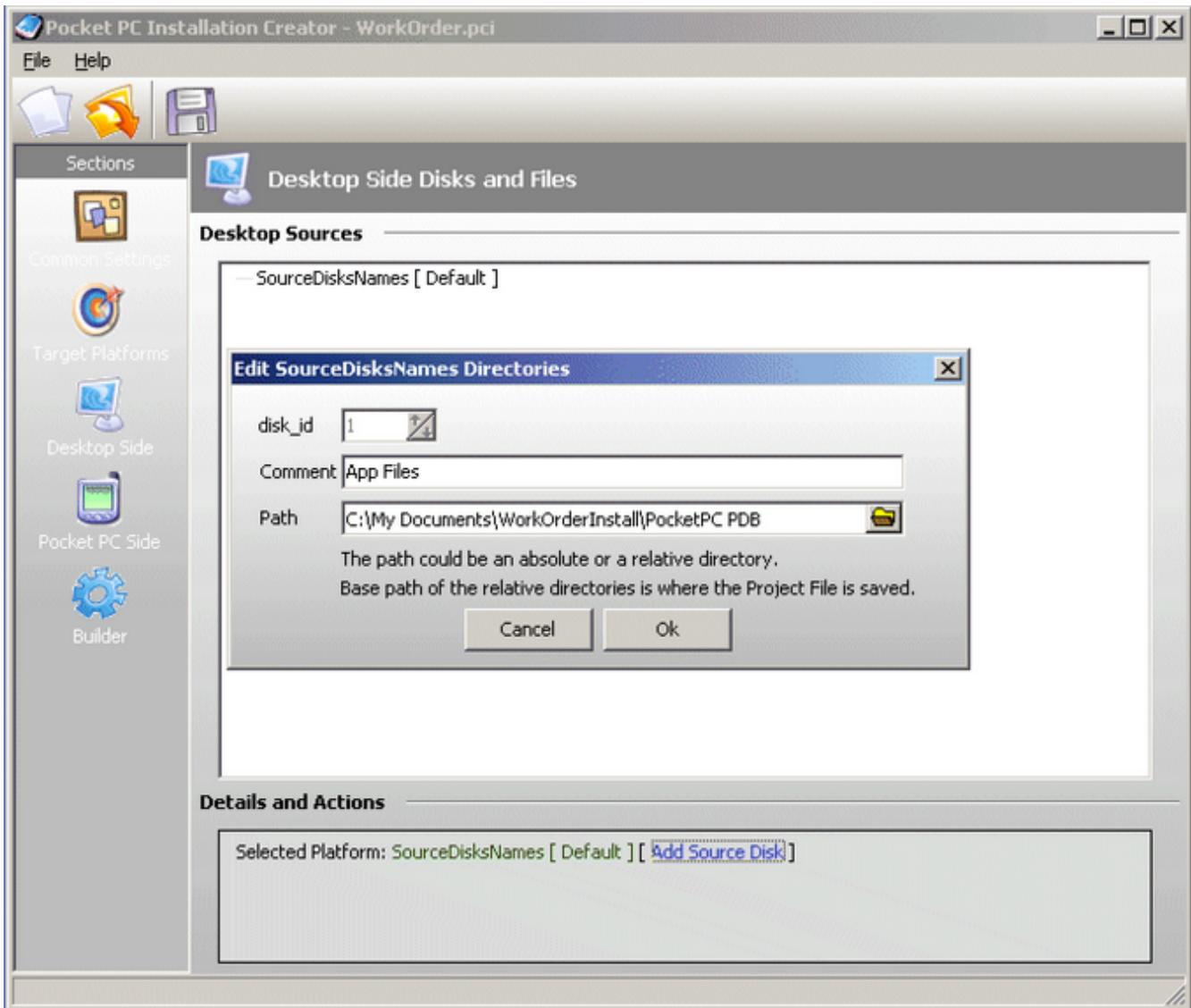
C.2. The Common Settings form is displayed. In the Provider field, enter your company name (in this example we are using SampleCo). In the AppName field enter your application name (in this example we are using Work Order). Leave the other fields unchanged. Click on the Target Platforms icon to go to that form.



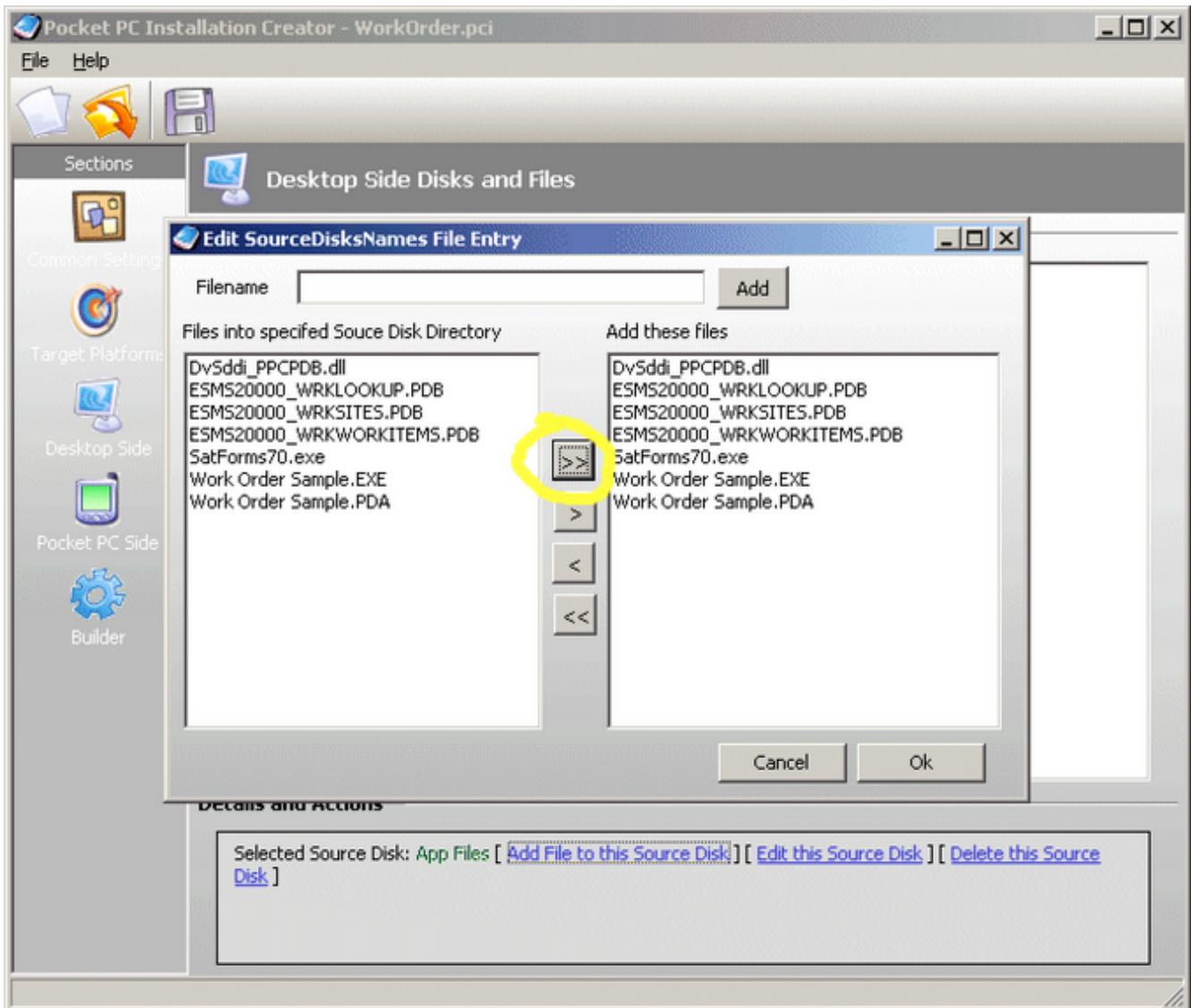
C.3. The Target Platforms form is displayed. Check the box next to CEDevice in the Target Label list so it is selected. Click on the BuildMax droplist and select "Supports both square screens and screen rotation". This ensures that you will not receive the "this app may not run on this version of Windows Mobile" warning when the app is installed. Click on the Desktop Side icon to go to that form.



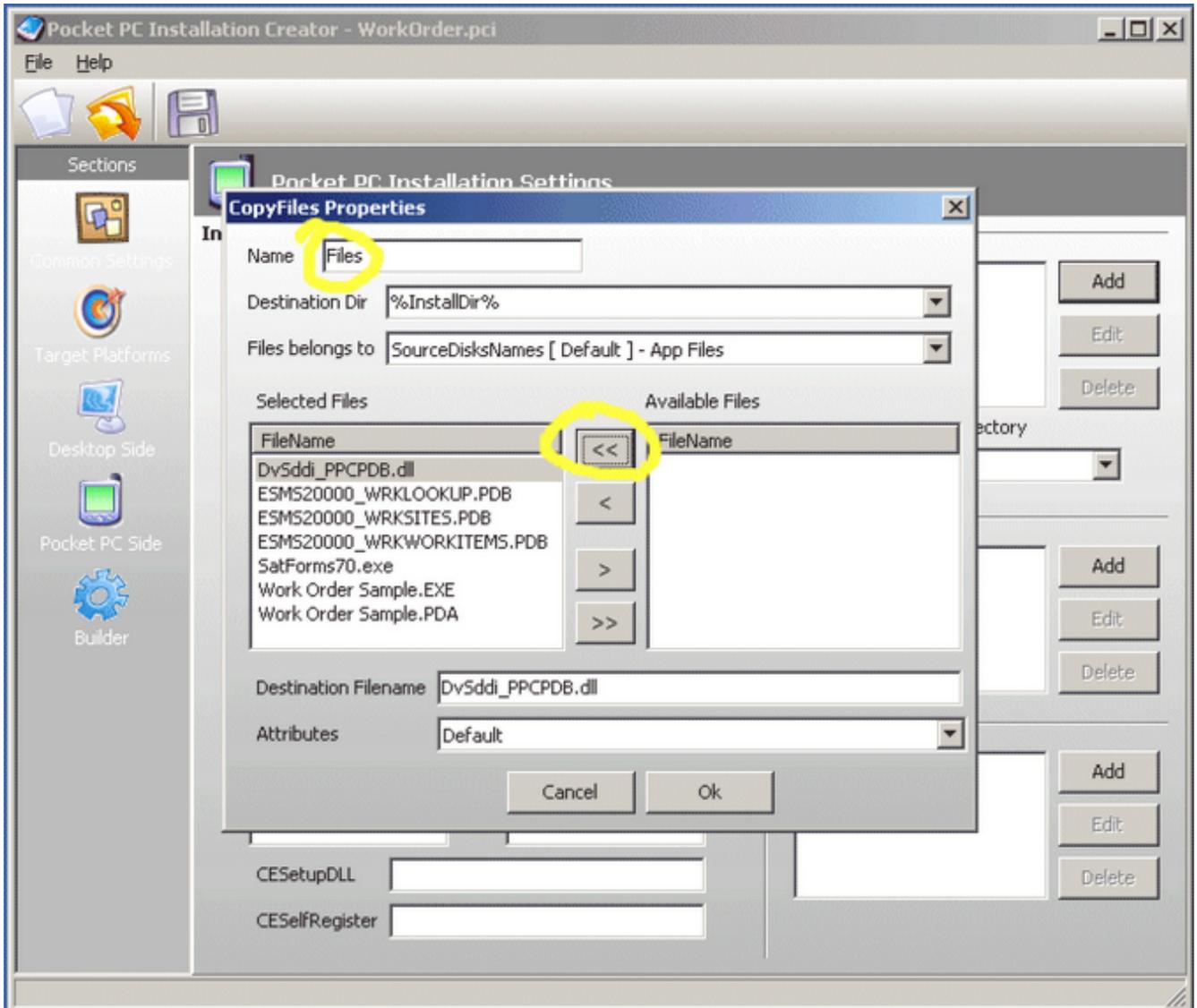
C.4. The Desktop Side Disks and Files form is displayed. This is where we will select all of the files to be installed to the PDA. In the Details and Actions box at the bottom of the form, click on Add Source Disk. The Edit SourceDiskNames dialog is displayed. In the Comment field enter App Files, and then browse to the folder where you have placed all of your PDA files (in this example C:\My Documents\WorkOrderInstall\PocketPC PDB), then click OK.



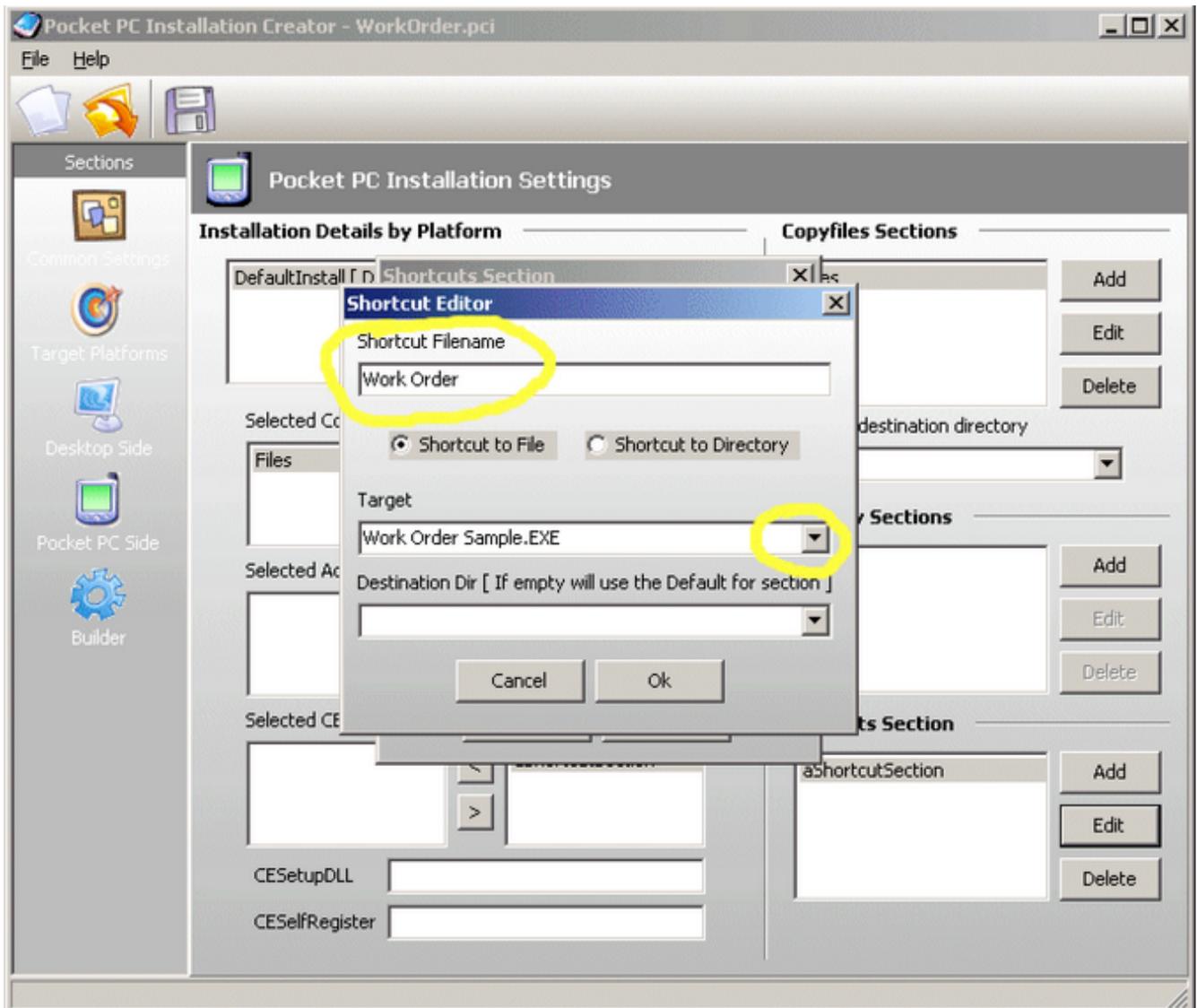
C.5. Now click on the [+] next to SourceDiskNames [Default] which expands to show App Files. Select App Files, then click on Add Files to this Source Disk in the Details and Actions box. The PDA files located in your app files folder should be shown in the list on the left hand side. Click on the >> button to copy all of those files to the list on the right hand side, then click OK. If you click the [+] next to App Files, you should see all of your PDA files listed. Now click on the Pocket PC Side icon to go to that form.



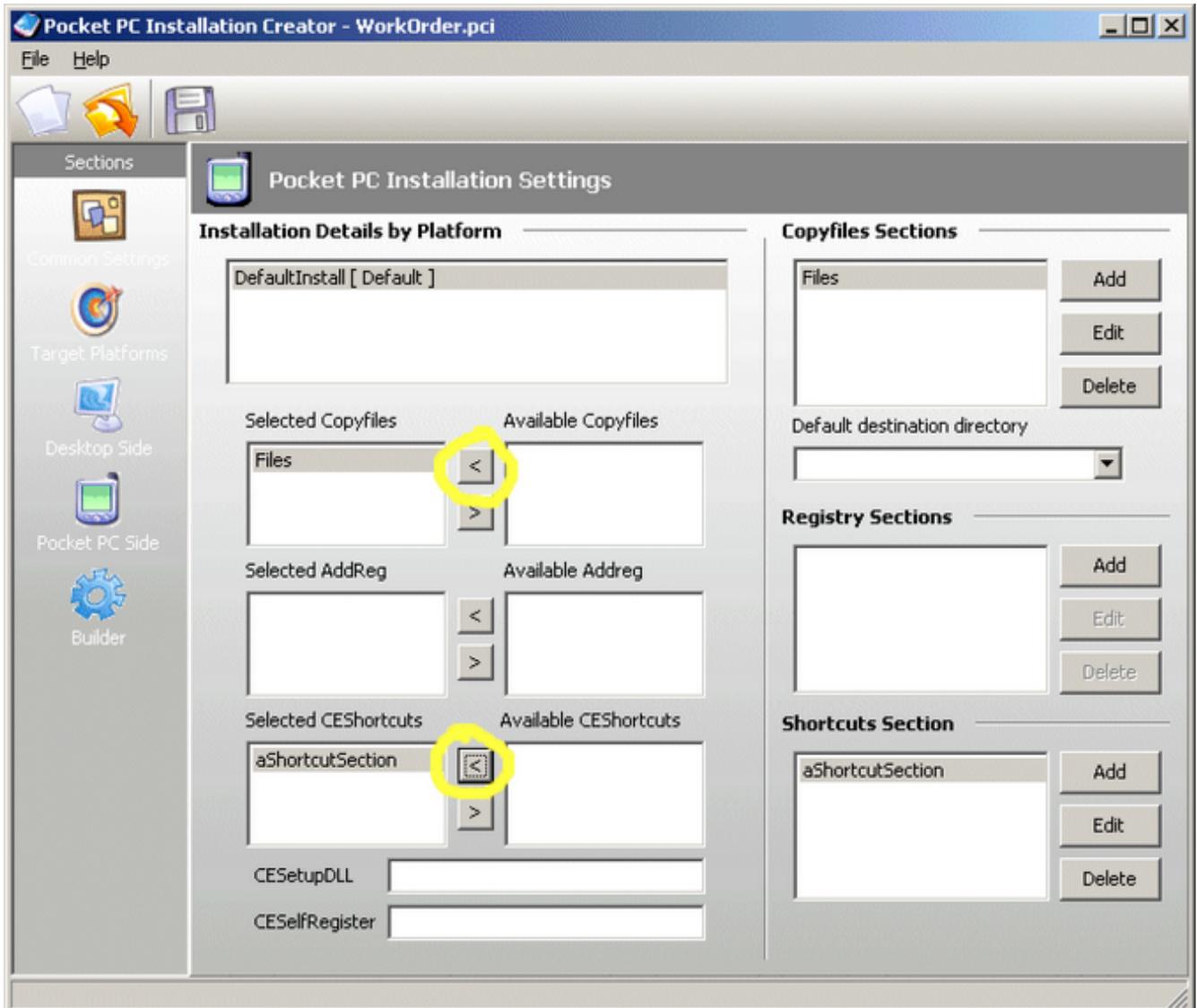
C.6. The Pocket PC Installation Settings form is displayed. In the CopyFiles Sections in the top right corner, click on Add. The CopyFiles Properties dialog is displayed. Enter Files in the Name field, then click on the << button to copy all of your PDA files from the list on the right hand side to the list on the left hand side. Click on OK to close that dialog.



C.7. Now, click on the Add button in the Shortcuts Section in the bottom right of the form, and the Shortcuts Section dialog is displayed. Click on the Add button in the Shortcuts Section dialog, and the Shortcut Editor dialog is displayed. Set the Shortcut Filename to your app name (eg. Work Order). This is the name that your app icon will have in the Programs folder on the PDA when the install is completed. Click the Target droplist, and select your application EXE file from the list (Work Order Sample.exe in this example). Click on OK to close the Shortcut Editor dialog, then click OK to close the Shortcuts Section dialog.



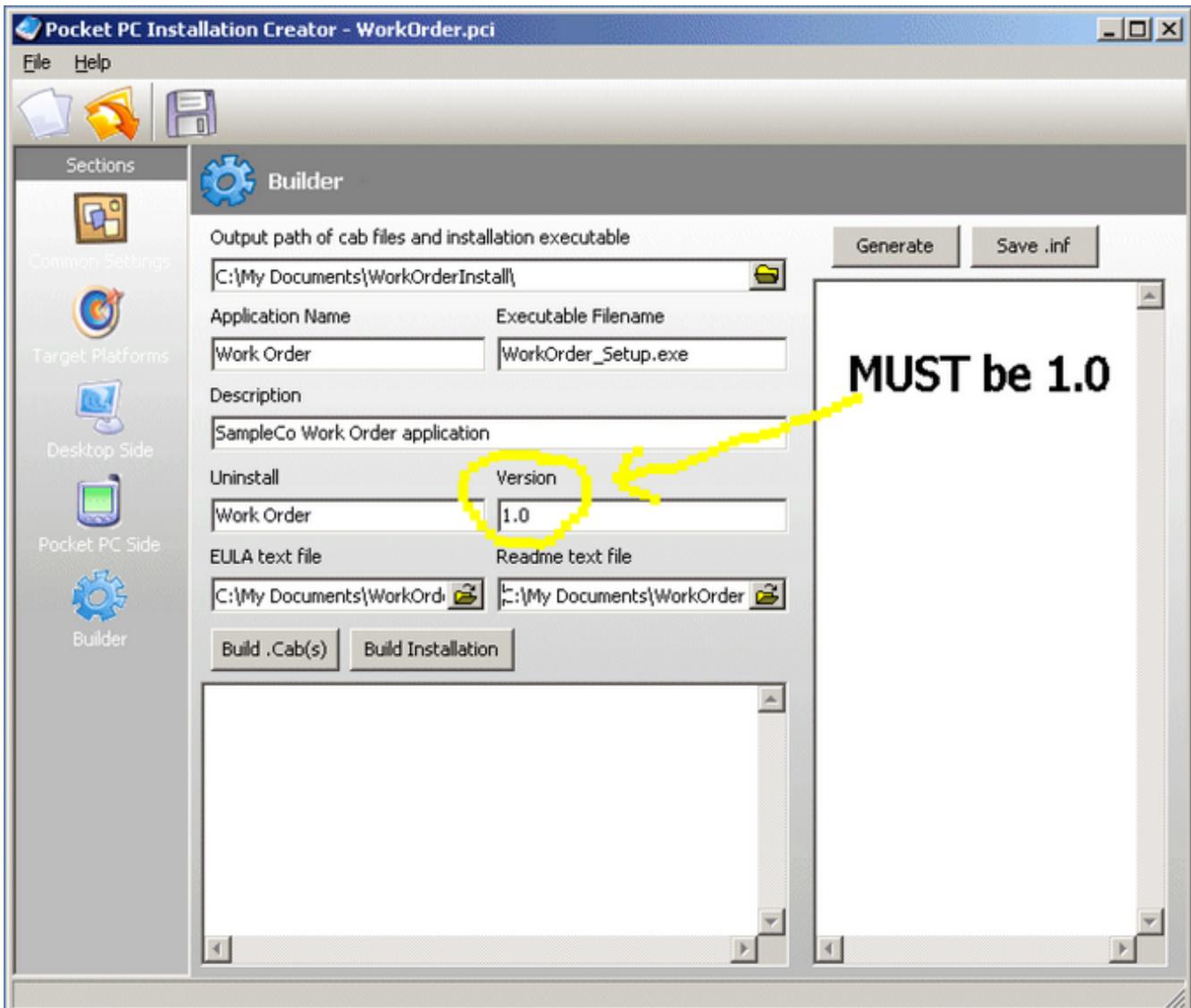
C.8. Next, click on the < button in the Available CopyFiles list to copy Files over to the Selected CopyFiles list. Then click on the < button by the Available CESHortcuts list to copy aShortcutSection over to the Selected CESHortcuts list. You have now completed this form, click on the Builder icon to go to the final Builder form.



C.9. The Builder form is displayed. Enter your Application Name (Work Order in this example), and enter a filename for your installer executable filename (eg. WorkOrder_Setup.exe). Enter a short Description (eg. SampleCo Work Order application). Enter the same application name in the Uninstall field (eg. Work Order).

IMPORTANT: The Version number field must ALWAYS be set to 1.0. This is the PocketPC CAB format version number, and NOT your application version number. Be sure it is set to 1.0 or the installation will fail!!

Browse to select your EULA text file (eg. EULA.txt) for your app license agreement, and your app readme text file (eg. setup_readme.txt). These will be displayed in the installer when it is run.



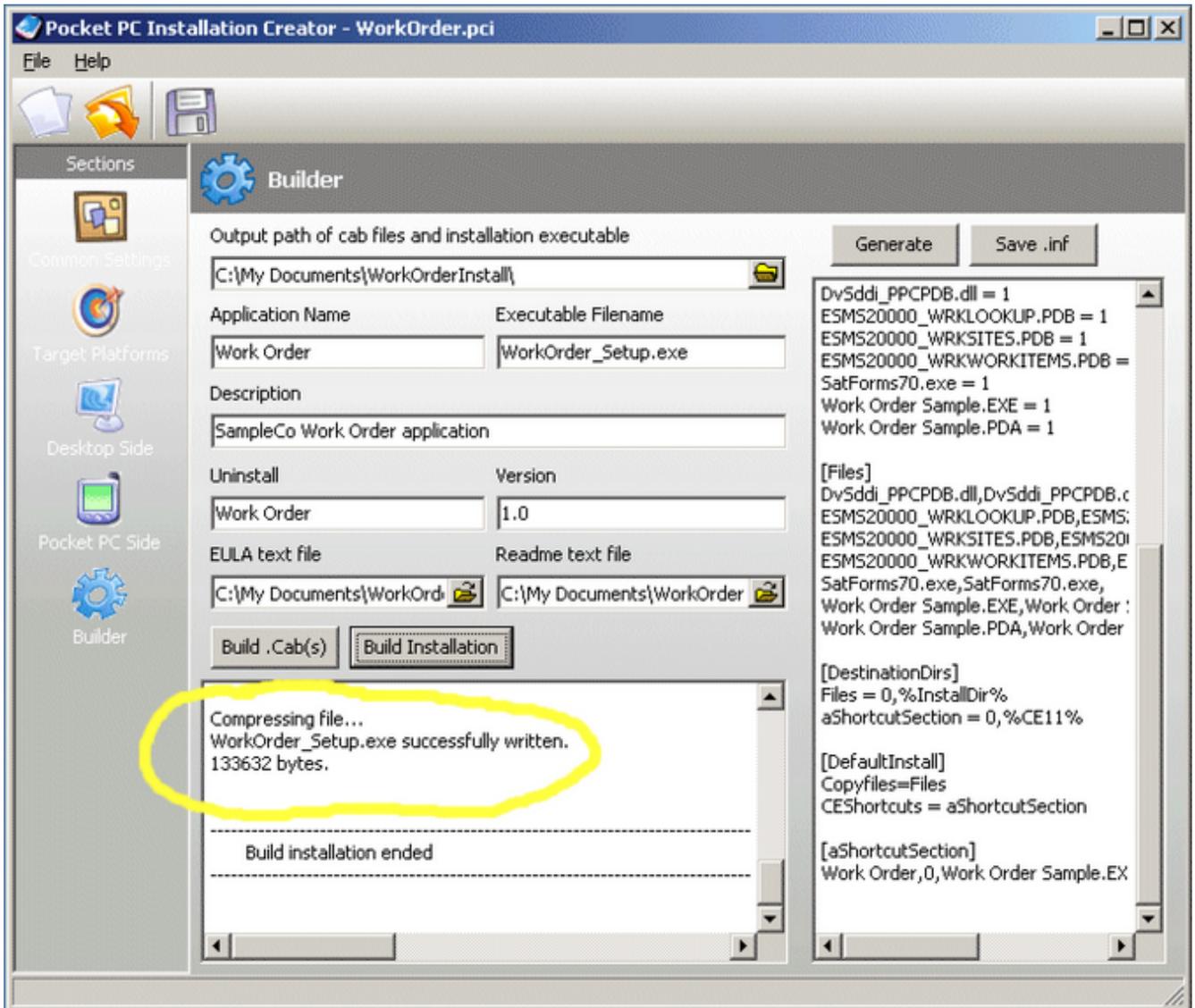
Almost finished! Now, let's save the data we have entered before building. Click on the Save disk icon on the toolbar to save your installer creation data file.

C.10. We're ready to build the CAB file. Click on Build .Cab(s) and the CAB file should be generated. You will see 2 warning messages that say "Warning: Section [DefaultInstall] key "AddReg" - there are no section entries to process", but they can be safely ignored. If you see any other error messages, then you will need to correct whatever errors they refer to before proceeding.

If you want to distribute just a CAB file that the user can install directly from the File Explorer on a PocketPC, you are done. However, most developers want to go the additional step of creating an installer that runs on the desktop PC. [Also, if you want to use a different final setup EXE builder tool that works from a CAB file, such as PocketSetupCreator, you can use the CAB you just created for that purpose.]

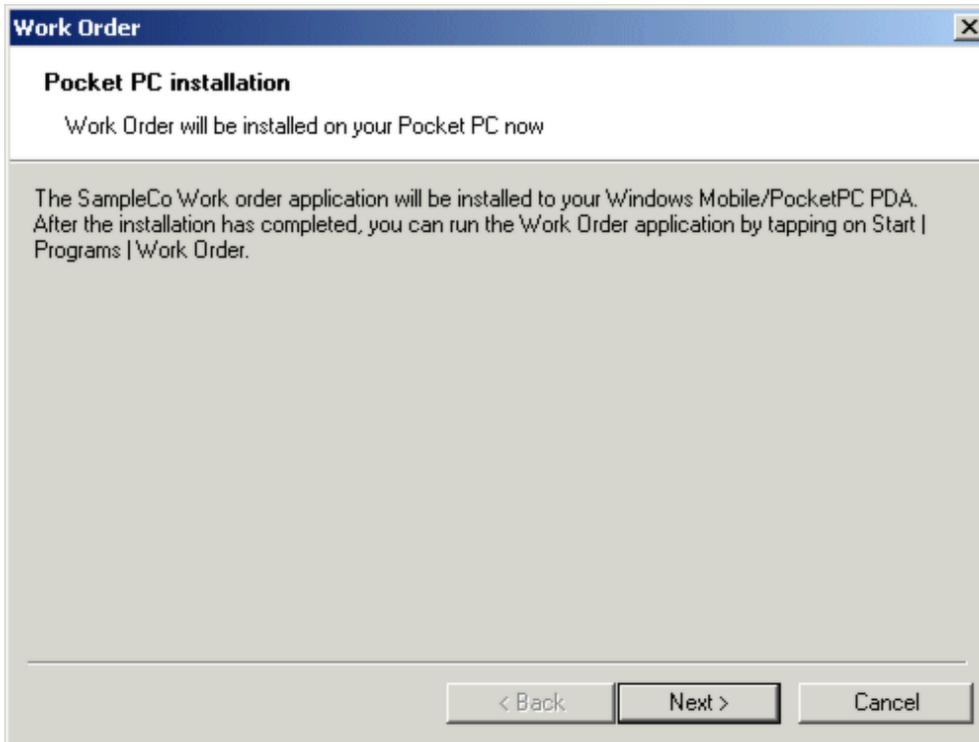
C.11. Now, the final step: click on Build Installation, and the CAB file will be used to generate your app setup exe file (eg. WorkOrder_Setup.exe). If all goes well, the status window should show the setup exe (eg. WorkOrder_Setup.exe) successfully written, and Build installation

ended.

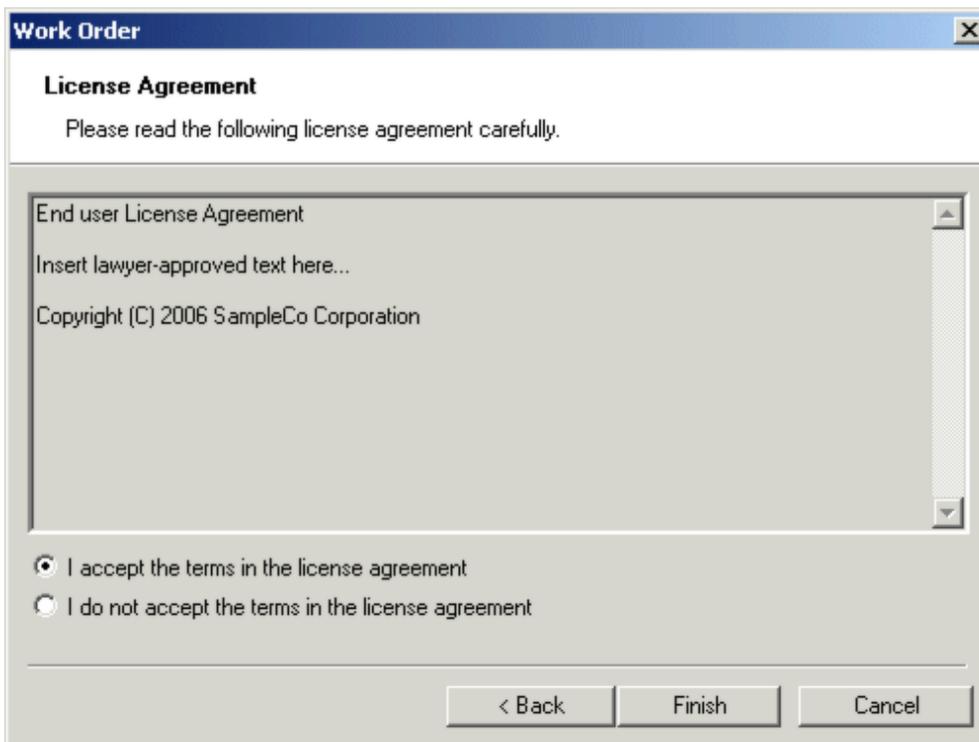


That's it, your installer is completed!

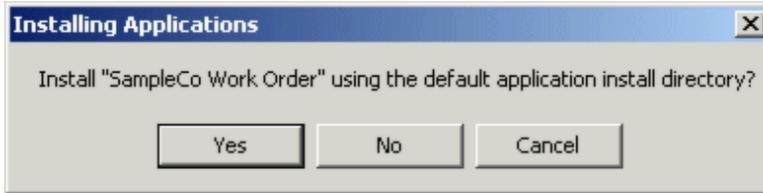
Now, let's test our new app installer. From the Windows Explorer, launch the setup exe (eg. WorkOrder_Setup.exe) that was generated. The initial installer screen is displayed, including the readme file contents.



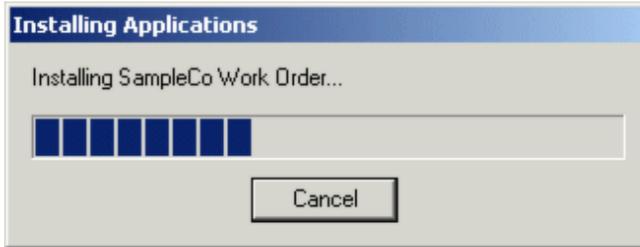
Click Next and the License Agreement screen is shown.



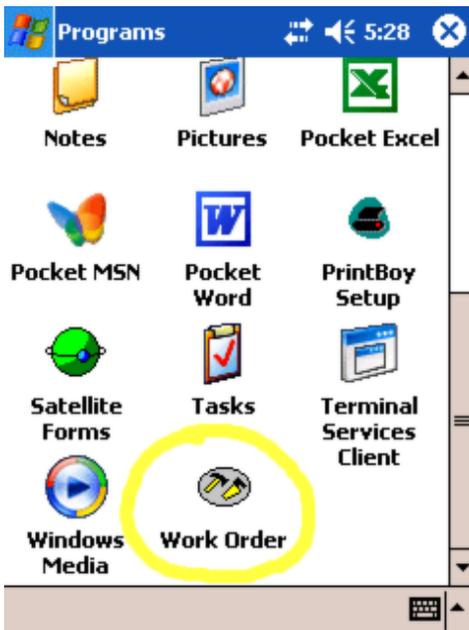
When the user accepts the terms of the license agreement and clicks Finish, the ActiveSync Add/Remove Programs dialog is displayed, and the Installing Applications prompt is shown, giving the user one last chance to cancel the install.



When the user affirms this prompt, the application is then installed to the PocketPC device.



The application can now be launched by tapping on Start | Programs | and then the application shortcut (eg. Work Order).



Success!

Keywords: PocketPC, Windows Mobile, install, installer, CAB, CABWiz, setup, installation, deployment, uninstall

KB ID: 10044
Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Create a shortcut to your PocketPC application

Problem: How To Create a shortcut to your PocketPC application

Updated: Starting with Satellite Forms 7.1, when you compile your PocketPC application and include an icon for it so that an EXE application is generated, MobileApp Designer will automatically create an application shortcut .Lnk file for you, and place it in the AppPkg folder. See the Satellite Forms 7.1 manual for more information. The information below for manually creating shortcut files is retained for Satellite Forms 7.0 and earlier.

Solution: This article explains how to programmatically create a shortcut to your application on the PocketPC, without using an installer. The shortcut appears in the Programs menu on the PocketPC.

A. Background Information

When you install your application to a PocketPC, you usually want to have a shortcut to your application appear in the programs screen on the PPC. This makes it easier for the user to start your application, by tapping on Start | Programs and then on your icon. The shortcut takes care of launching your app, wherever it happens to be located (eg. in \My Documents\YourApp or \Program Files\YourApp).

How do you create this shortcut? Usually, this task has been carried out by using an application installer to install your app to the PPC. The installer takes care of copying your application files to a specified folder on the PPC, and creating a shortcut to your app in the Programs screen. Examples of some software tools used to create PocketPC app installers include the freeware tools EzSetup, Pocket PC Installation Creator, PocketSetupCreator, and NSIS, as well as commercial tools like InstallShield, e-PocketSetup, Spb AirSetup, etc.

Users can also create shortcuts manually. However, this is something you would generally want to take care of programmatically, instead of expecting the user to carry out this task manually. Asking the user to create a shortcut manually does not make for a good user experience.

There is another way to create this shortcut for your user automatically, without using an installer. For some developers, this task of creating the shortcut may have been the only reason that they used an installer, as they could carry out the rest of the app installation tasks using the CeRDKInst tool included with Satellite Forms. Using this new approach, the shortcut file is created manually on the PC, and that shortcut file is sent to the PocketPC along with all of the other application files, using the CeRDKInst tool. No PocketPC application installer is required. This may be the preferred approach for corporate/enterprise deployments, where other features of an installer (eg. display of a license agreement and readme file) are not needed.

B. Creating the Shortcut on the Desktop PC

The key to understanding this approach is recognizing that the shortcut you see in the PPC programs folder is actually just a file with certain properties, placed in a certain location on the PocketPC. That file can be created on the desktop PC using any standard text editor, then copied to the right location on the PocketPC.

The shortcut file should be named as you want it displayed in the programs folder, with a .lnk file suffix. For example, if your app is called SuperDuperApp.exe, but you would like it to be listed in the programs folder as "Super Duper", you would create the file "Super Duper.lnk". The name of the shortcut does not have to match

the name of your application EXE file.

NOTE: Windows treats files with the .lnk suffix in a special manner, and considers them to be desktop shortcut files. This means it may be difficult for you to work with the file using Windows Explorer, as it does not treat the file like a regular text file. We find that opening your text editor, and then loading the shortcut .lnk file from within your text editor is easier than trying to work with the file in Windows Explorer.

The .lnk file itself is just a regular text file, containing the following text:

```
nn#"Path\To\Your App.exe"
```

nn is a decimal number equal to the length of the exe file path string, including the starting and ending quote chars

is a literal # character

"Path\To\Your App.exe" is the path to your application EXE file

So, let's use the example of creating a shortcut to the \My Documents\SuperDuperApp\SuperDuperApp.exe app, using the shortcut name "Super Duper". Create a text file named "Super Duper.lnk" which includes this single line of text:

```
47#"My Documents\SuperDuperApp\SuperDuperApp.exe"
```

Remember that 47 is the length of the "\My Documents\SuperDuperApp\SuperDuperApp.exe" path string. You now have a valid shortcut file for your application, and just need to place it in the right location on the PocketPC.

C. Placing the Shortcut file on the PocketPC

Now that you have created the shortcut .lnk file on the PC, all you need to do to make that shortcut visible on the PocketPC is to copy it to the correct folder on the PPC. The correct folder to place the shortcut is the "\Windows\Start Menu\Programs" folder. This is the folder that is displayed when the user taps on Start | Programs on the PPC. Simply copy the .lnk file into the "\Windows\Start Menu\Programs" folder, and the PocketPC will automatically find the icon for your application based on the path in the .LNK file. It will display your app icon, with the name of the lnk file. In our example above, the icon shown in the Programs screen would be Super Duper and it would show the icon contained within the "\My Documents\SuperDuperApp\SuperDuperApp.exe" file. Tapping on the Super Duper icon will start the application.

That is all there is to it!

D. A sample CeRDKInst install config file that includes a shortcut

The CeRDKInst utility included with Satellite Forms is used to install a list of files to the PocketPC. Using the new ["bundled runtime engine"](#) approach that is now available with [Satellite Forms 7 using PDB databases](#), the CeRDKInst tool can be used to install your application and runtime engine together. You can have CeRDKInst

install the shortcut file you created above also, for a complete PocketPC installation solution.

The documentation for the CeRDKInst tool is included in the main Satellite Forms help file, under the topic Deploying Your Application | Working with the Pocket PC install utility. The sample below shows a complete install ini file for an application, runtime engine, and shortcut file:

```
;Super Duper App config file for CeRDKInst
;
; Installer Section
[Installer]
EngWinTitle=Super Duper App Installer
;
; Files0 Section - app files and SF runtime
; installed to \Program Files\SuperDuperApp folder
[Files0]
DevDir=\Program Files\SuperDuperApp
Sync=FALSE
File0=..\PDAFiles\SuperDuperApp.PDA
File1=..\PDAFiles\SuperDuperApp.EXE
File2=..\PDAFiles\ESDAP0100_SDITEMS.PDB
File3=..\PDAFiles\ESDAP0100_SDSITES.PDB
File4=..\PDAFiles\ESDAP0100_SDLOOKUP.PDB
File5=..\PDAFiles\SatForms70.EXE
File6=..\PDAFiles\DvSddi_PPCPDB.DLL
;
; Files1 Section - shortcut file for SuperDuperApp
; installed to \Windows\Start Menu\Programs
[Files1]
DevDir=\Windows\Start Menu\Programs
Sync=FALSE
File0=..\PDAFiles\Super Duper.Ink
```

Keywords: PocketPC, install, CERDKInst, shortcut, Windows Mobile, setup, installer, deployment

KB ID: 10045
Updated: 2007-12-03

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Implement a Quick Find feature

Problem: How To Implement a Quick Find feature

Solution: This article describes a technique for implementing a Quick Find feature in your application, enabling your users to incrementally search through a large list. As the user enters characters one by one, the closest matching record will be located.

You've got a list of 5000 items in your application, sorted by name, and you want to make it easy for the user to select the right item in a flash. Forcing the user to scroll page by page through 5000 items won't cut it: this job calls for a Quick Find approach using a binary search!

The Binarysearch function requires that your table be sorted on the field that you are searching. If it is not sorted correctly, the Binarysearch function will return undefined results.

Add an edit control to the form labeled Find (the control name in this sample is edFind). Add a short script to the form's AfterChange event that binary searches the table for the Find text, and positions the form on that matching record, like this:

```
dim bFound, iFoundRow
bFound = Tables().BinarySearch("Product", True, edFind, iFoundRow)
forms().currentrecord = iFoundRow
```

Now as the user enters characters, the form will instantly jump down to the matching record, highlighted in the list! This technique works on the PalmOS & PocketPC platforms.

If an exact match is not found, the Binarysearch function returns the record number in which a match *should* be found if it was in the database (the correct sort position).

Want to spice it up a little more? You can add some additional code to show the complete matched value in the Find edit control, with the characters not yet entered shown as selected, and the input cursor positioned right after the last entered char.

eg: FIND **Smit**

You'll need to do a bit of extra work to keep track of how many chars were entered by the user, so you know how many to show as selected in the edFind control. Here's an updated AfterChange script with the additional code.

```
dim bFound, iFoundRow, iFindLen, iMatchLen
```

```
iFindLen = Len(edFind) 'how many chars entered by user
```

```
bFound = Tables().BinarySearch("Product", True, edFind, iFoundRow)
forms().currentrecord = iFoundRow
```

```
edFind = Tables().Fields("Product") 'show match in edFind
edFind.SetSelection(iFindLen, Len(edFind)) 'highlight non-entered chars and set cursor position
```

The trick here is that because you highlight the non-entered chars and set the cursor position after the chars that were entered, the next character that is entered overwrites the highlighted selection. This enables the `iFindLen = Len(edFind)` code in the AfterChange event to get the new length of entered characters, before the edFind field is updated again with the rest of the matched data.

Tip: You can change the length of time before the AfterChange event is fired (called the

AfterChange delay), either on an application wide basis or on an immediate runtime basis. To change the AfterChange delay (the default delay is 2 seconds) on a project wide basis, go to the Build | Options menu and modify the OnChange Delay value. To change that value as your application is running you can use the script function SetDelayToChangeEvent.

Keywords: Find, search, incremental, quick, lookup, binarysearch, binsearch, typeahead

KB ID: 10047

Updated: 2007-02-01

[Satellite Forms KnowledgeBase Online](#)

[Satellite Forms Website Home](#)

-0-

How To Use OnPenDown/OnPenUp Scripts to Detect Pen Taps on Controls

Problem: How To Use OnPenDown/OnPenUp Scripts to Detect Pen Taps on Controls

Solution: This article describes a technique using a global PenTapInControl script in conjunction with OnPenDown/OnPenUp form scripts to detect when controls are tapped on, for use on controls that do not have an OnClick method (such as edit controls, text labels, lookups, bitmaps, etc).

Let's assume that you want to detect pen taps in a control in a manner consistent with the standard controls that support OnClick events; that is, a valid "click" is when the user taps the pen down on the control, and then lifts the pen up while still on the control. Tapping down on a control, then sliding off the control and lifting the pen when not on the control does not count as a valid click.

This can be accomplished using a combination of global functions, global variables, and OnPenDown/OnPenUp scripts on a given form.

First the global vars that we need in order to track pen tap coordinates across multiple scripts:

```
'pen tracking global vars  
Dim gOPDX, gOPDY, gOPUX, gOPUY
```

[For those curious about the variable naming convention, they are named after Global OnPenDown X, Global OnPenDown Y, Global OnPenUp X, Global OnPenUp Y.]

Next the OnPenDown script on a form:

```
'record the pen down but don't track it around the form  
'save x/y coords to global vars for use in other scripts  
'note these save to the "D" or "Down" global vars  
GetPenStatus(gOPDX, gOPDY)
```

Next the OnPenUp script. This example tracks taps for two controls, Edit1 and Edit2.

```
'record the pen up position but don't track it around the form  
'save x/y coords to global vars for use in other scripts  
'note these save to the "U" or "Up" global vars  
GetPenStatus(gOPUX, gOPUY)
```

```
Dim cX, cY, cW, cH  
'test if pen tapped within bounds of Edit1 control  
Edit1.GetPosition(cX, cY, cW, cH)  
If PenTapInControl(cX, cY, cW, cH) then  
  msgbox("You tapped inside Edit1")  
  Exit 'leave the script  
Endif
```

```
'test if pen tapped within bounds of Edit2 control  
Edit2.GetPosition(cX, cY, cW, cH)  
If PenTapInControl(cX, cY, cW, cH) then  
  msgbox("You tapped inside Edit2")  
  Exit 'leave the script
```

Endif

Okay, so you can see that the OnPenUp event script calls a global function named PenTapInControl, passing it the coordinates of the control that you want to test the pen taps for. Remember that we consider it a tap ONLY if the pen taps down on the form and also lifts up from the form within the bounds of a given control. A user can tap down then slide away and lift off outside of a control and that is not considered a tap (just like the standard UI behaviour for controls). That PenTapInControl global function looks like this:

```
'check supplied coordinates against stored global PenUp and PenDown coords
'return True if pen tapped (down and up) in control or False if not
Function PenTapInControl( cX, cY, cW, cH )
  Dim dDX, dDY, dUX, dUY
  dDX = gOPDX - cX 'pen down x coordinate differential
  dUX = gOPUX - cX 'pen up x coordinate differential
  dDY = gOPDY - cY 'pen down y coordinate differential
  dUY = gOPY - cY 'pen up y coordinate differential
  'verify that pen differentials lie within width and height of control
  If ( (dDX <= cW) and (dDX >= 0) and _
    (dUX <= cW) and (dUX >= 0) and _
    (dDY <= cH) and (dDY >= 0) and _
    (dUY <= cH) and (dUY >= 0) ) then
    PenTapInControl = True
  else
    PenTapInControl = False
  Endif
End Function
```

That should do it. You can then expand it to other controls by adding additional PenTapInControl tests in the OnPenUp script.

Keywords: PenTapInControl, OnPenDown, OnPenUp, pen, tap, click, global, GetPosition, GetPenStatus

KB ID: 10059
Updated: 2007-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Limit Edit Control Input to Numeric Only

Problem: How To Limit Edit Control Input to Numeric Only

Solution: There are times when you would like to limit allowed input into certain controls to numeric characters only (the digits 0..9, and ".", ",", and "-" characters, as well as the backspace). While this can be accomplished in several different ways (for example, the AfterChange event that gets fired when a control changes its value, or the OnValidate event which fires before the form's current data is saved to the linked table), this example takes the approach of filtering input chars as they are entered. This can be accomplished using a global script that can be called from multiple forms, and the OnKey event script on any given form.

In the OnKey event of the form, we can test whether the input is numeric or not. If not, then we can beep and discard the input. Let's say we only want to perform this numeric input validation:

```
dim AKey, VKey, MKey
GetLastKey(AKey, VKey, MKey)

if (AKey > 255) then Exit 'allow virtual keys to be processed by the OS

'if input to edNumeric control then check for numeric input only
if Forms().GetFocus = edNumeric.Index then
  if not IsNumericInput(AKey) then
    Beep(1)
    Fail 'non-numeric entry disabled
  endif
endif
endif
```

As you can see, the OnKey script calls a global function named IsNumericInput and passes it the ASCII value of the character that was just received from the keyboard input queue. Let's create a Global function named IsNumericInput that examines that input char and returns True if it is a valid numeric char, or False if it is not. The backspace key is handled by the AKey = 8 test, which tests the ASCII code instead of testing the character as is done for the digits and numeric punctuation:

```
Function IsNumericInput(AKey)
  dim char
  char = CHR(AKey) 'get the character
  if ((char >= "0") and (char <= "9")) _
    or (char = ".") _
    or (char = ",") _
    or (char = "-") _
    or (AKey = 8) then
    IsNumericInput = true
  else
    IsNumericInput = false
  endif
End Function
```

That should do it. You can then expand it to other controls and/or forms by adding additional IsNumericInput tests in the OnKeyUp script.

Keywords: IsNumericInput, numeric, char, OnKey, GetLastKey, validate

KB ID: 10060

Updated: 2007-07-11

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Force Input To ALL CAPS

Problem: How To Force Input To ALL CAPS

Solution: On occasion you may wish to force input in an edit control to ALL CAPS. There are several different ways to accomplish this, for example using the AfterChange event that gets fired when a control changes its value, or the OnValidate event which fires before the form's current data is saved to the linked table. This article presents a method for filtering the characters as they are inputted via the OnKey event, converting them to ALL CAPS before they get posted into the edit control.

This can be accomplished using a global function in conjunction with the PasteChars function of the new SysUtils extension (introduced with Satellite Forms 7.1). Let's start with an IsUppercaseInput global function that just looks at the input and decides if it is uppercase text or not. For simplicity, this function treats "a".."z" as Not Uppercase, and everything else as Uppercase. You could of course extend these routines to handle the international accented chars too -- the example below is for just for unaccented "a".."z":

```
Function IsUppercaseInput(AKey)
'use AKey values directly not characters here because
'in SF script "a" = "A" when doing string comparison
if ((AKey >= 97) and (AKey <= 122)) then
  IsUppercaseInput = false
else
  IsUppercaseInput = true
endif
End Function
```

Now, in the form's OnKey event we can check if the input was uppercase and if not we can post the uppercase version of that key into the keyboard input queue (using PasteChars from the SysUtils extension), and discard the original lower case input. In this example, we are performing the IsUppercaseInput filtering on an edit control name edAllCaps only, and leaving input to other controls unfiltered:

```
dim AKey, VKey, MKey
GetLastKey(AKey, VKey, MKey)

'if input to edAllCaps control then check for uppercase input only
if Forms().GetFocus = edAllCaps.Index then
  if not IsUppercaseInput(AKey) then
    'input is from lowercase a..z so post uppercase key instead
    'uppercase letters ASCII values are 32 less than lowercase values
    SU_PasteChars(CHR(AKey - 32)) 'post uppercase key to input
    Fail 'cancel the lowercase key
  endif
endif
```

That should do it. You can expand it to other controls by adding additional IsUppercaseInput tests in the OnKey script, and you can expand the comparison to accented chars by modifying the IsUppercaseInput global function.

Keywords: IsUppercaseInput, upper case, OnKey, capitals, GetLastKey

KB ID: 10061

Updated: 2007-07-11

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Dial a Phone Number Using the LaunchURL Extension

Problem: How To Dial a Phone Number Using the LaunchURL Extension

Solution: The LaunchURL extension, included with Satellite Forms 7.0 and also available by download for Satellite Forms 6.x, is designed to enable your application to open a website URL in the device's default web browser. It can also be used to make phone-enabled devices dial a phone number, by using a special URL.

The special URL to dial a phone number is tel:nnn-xxx-xxxx where nnn-xxx-xxxx is the telephone number.

To dial a number, use the LaunchURL function from the LaunchURL extension to launch the tel: URL, like this:

```
dim err, URL
URL = "tel:1-800-555-1212"
err = LaunchURL( URL )
```

On PalmOS Treo smartphones, a dial confirmation dialog will appear, with the phone number displayed and the button choices Dial, Message, Cancel. If you tap Dial, your application is closed, the Phone app is launched, and the phone number will be dialed. If you tap Message, your app will close, and the SMS Messaging app will be launched with the To: phone number filled in. You can then type in the message and send it via SMS. If you tap Cancel, the dial confirmation dialog is dismissed, and you remain in the Satellite Forms application.

On PocketPC Phone Edition devices, the Phone application will be launched and the phone number will be dialed. Your Satellite Forms application will still be running in the background while the phone application is active.

Keywords: phone, dial, URL, launch, telephone, call, message, LaunchURL

KB ID: 10072

Updated: 2007-06-26

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Send an Email Message Using the LaunchURL Extension

Problem: How To Send an Email Message Using the LaunchURL Extension

Solution: The LaunchURL extension, included with Satellite Forms 7.0 and also available by download for Satellite Forms 6.x, is designed to enable your application to open a website URL in the device's default web browser. It can also be used to send an email message, by using a special URL.

The special URL to send an email message is the mailto: URL. The mailto: URL is a defined internet URL standard, like http:, designed for sending shorter email messages via the web browser. For complete details on the mailto: URL scheme, use Google to search for the relevant reference documents and samples.

To send a short email message, use the LaunchURL function from the LaunchURL extension to launch the mailto: URL, like this:

```
dim err, URL
URL =
"mailto:user@domain.com?subject=This%20is%20%20subject&body=This%20is%20the%20message%20bod
y."
err = LaunchURL( URL )
```

On PalmOS Treo smartphones, your SatForms application will close and the the default email client (usually VersaMail) will appear, with the mail address, subject, and body information filled out. You can edit the message and then send it.

On PocketPC Phone Edition devices, the default email application (usually Messaging) will be launched with the address and message information filled out. Your Satellite Forms application will still be running in the background while the email application is active.

Keywords: email, message, URL, LaunchURL, mail

KB ID: 10073
Updated: 2007-06-26

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Commit Table Data to Storage Immediately

Problem: How To Commit Table Data to Storage Immediately

Solution: Newer PalmOS devices, and Windows Mobile/PocketPC devices using the PDB database format, employ a system of in-memory caching of table data for improved performance. Changes to cached tables are automatically committed, or saved, back to storage when the application is closed. In order to prevent loss of data due to a system reset before the application is closed, you can force the cached data to be committed immediately on a table by table basis, either by using the `Tables("tablename").CommitData` method that is new in Satellite Forms 7.1, or by using the PalmOS `DmSyncDatabase` extension or the PocketPC `CommitDatabase` extension.

A. Overview

The original PalmOS memory system stored non-built-in applications and data in battery-powered RAM memory. Everything worked well as long as the PalmOS device had battery power, but if the batteries died then the contents of RAM (including applications and data) were lost. A standard reset would not lose any data, because the contents of RAM were not cleared, leaving apps and data intact.

A few years ago a new PalmOS memory system was introduced in which applications and data were stored in non-volatile memory, keeping applications and data intact even if the batteries go dead. The main impetus for this new system, called the Non-Volatile File System (NVFS), was to enable "on the go" swapping of removable batteries in Palm Treo smartphones, without losing data such as phone numbers, appointments, and so on. The overall advantages of the NVFS memory system were readily apparent, and so NVFS was quickly adopted on most other non-Treo PalmOS 5.x devices as well, becoming the new normal memory system.

While the goals of NVFS were certainly admirable, and in general terms it does work well, the actual implementation of NVFS has a few holes that make it necessary to consider data protection strategies in your application. In basic terms, the NVFS system caches database tables that are open (in use) in regular RAM memory, and then commits (saves) the modified databases back to non-volatile storage when the database is closed (when the application ends) and when the power is turned off. This is all performed automatically behind the scenes by the PalmOS. The problem comes when the device is reset (due to a pin reset or to a system crash): if the cached databases have not been committed to storage before the reset, then all of the modified data is lost when the device restarts.

For Windows Mobile/PocketPC devices, the Satellite Forms PDB database format uses a similar system of caching modified data in memory for enhanced performance. While the applications databases are open, changes are cached in RAM, and when the application closes the modified data is committed to storage. This process of committing data to storage is what causes the delay when a PocketPC PDB application is closed. Under this PDB data caching system, a device reset/crash can also cause data loss if the data has not been committed to storage before the reset.

The implication of this data loss on reset behaviour is that in order to ensure your modified application data is saved to non-volatile storage, your application may need to periodically force its data to be committed to storage before the application closes.

Starting with Satellite Forms 7.1, a new `Tables("tablename").CommitData` method is available to perform this task. At any point in your application you can immediately commit any specified table to storage by calling the `Tables("tablename").CommitData` method. For versions of Satellite Forms prior to 7.1, two extensions are available to provide this capability. The

DmSyncDatabase extension for PalmOS provides this capability with the DmSyncDatabase(Tables("tablename").Index) script method. It is available for download from the Products & Support | Support Files | Software Updates & Patches section of the Satellite Forms website, and is also included as a standard extension in SatForms 7.0. The CommitDatabase extension for PocketPC provides this same capability for the Windows Mobile/PocketPC platform, using the CommitDatabase(Tables("tablename").Index) script method. The CommitDatabase extension is also available for download from the Products & Support | Support Files | Software Updates & Patches section of the Satellite Forms website.

The natural question that comes to many developers' minds is "when or how often should I commit the table data in my application to prevent data loss?" The simplest answer is that it depends on the application. Because the actual process of committing a table to storage does take a little time and is not instantaneous, there is a tradeoff between ensuring complete data protection versus acceptable application performance. It is up to you as the developer to determine the optimal blend of data protection and performance for your application, and to use the CommitData table method accordingly. For example, an inventory scanning application may want to commit the data table after each new record is scanned, while a sales entry app may wish to wait until a multi-item order is completed before committing the data. A medical reference application may not need to commit data at all, and just rely on the automatic committing of data when the application is closed. There is no single recommended approach to committing application data for all applications: it is dependent on what your application does and how important it is to ensure that modified data is saved.

B. Usage

(a) Satellite Forms 7.1 and higher

The recommended approach when using Satellite Forms 7.1 and higher is to call the new Tables("tablename").CommitData method on your desired table(s). This method works the same on both the PalmOS and Windows Mobile/PocketPC platforms, enabling you to use the same script code on both platforms. Here is the method prototype:

CommitData

Tables(TableName).CommitData		
Commits (saves) the cached table to storage immediately.		
Parameter	<i>TableName</i>	Name of a table.
Return Value	None	

Comments	<p>CommitData is a method of the Table object. Use CommitData to commit the cached table data to storage immediately.</p> <p>Table data is cached when it is in use, and committed to storage automatically when the application closes. If the device is reset or power is lost before the application is closed, modified data in the cache is not written to storage, and is therefore lost when the device restarts. The CommitData method enables you to commit cached table data to storage immediately, in order to protect against data loss from a device reset.</p> <p>Note that the CommitData method is NOT affected by active filters. All records in a table will be committed regardless of whether they are currently filtered out of view.</p>
Example	<pre>'Example of CommitData method. 'Emps is a table. 'Add new employee and commit table to storage. Tables("Emps").CreateRecord Tables("Emps").MoveLast Tables("Emps").Fields("Fname") = "Joe" Tables("Emps").Fields("Lname") = "User" Tables("Emps").CommitData</pre>

(b) Satellite Forms versions prior to 7.1

For Satellite Forms versions prior to 7.1, the CommitData table method is not available. The table committing functionality is available through the DmSyncDatabase extension for PalmOS, and the CommitDatabase extension for PocketPC. In order to seamlessly integrate the table commit procedure into your pre-7.1 SatForms application for both the PalmOS and PocketPC platforms, follow this approach which is a combination of using the extension functions and global scripts. This allows your form level scripts to use the same script code for either/both platforms, and follows the approach presented in the KnowledgeBase article [How To use Global Functions & Subs to replace extension functions not available on the current target platforms](#).

- First, add the DmSyncDatabase extension to your PalmOS application target, using the Manage Extensions toolbar.
- Next, add the CommitDatabase extension to your PocketPC PDB application target, using the Manage Extensions toolbar.
- In the PalmOS build target, create a new Private (not Shared) Global Sub named CommitData:

```
'CommitData global sub forPalmOS uses DmSyncDatabase
'the table index is passed to this function as a string variable
Sub CommitData (tIndex)
    DmSyncDatabase(tIndex)
End Sub
```
- In the PocketPC build target, also create a new Private (not Shared) Global Sub named CommitData:

```
'CommitData global sub for PocketPC uses CommitDatabase
'the table index is passed to this function as a string variable
Sub CommitData (tIndex)
    CommitDatabase(tIndex)
End Sub
```

- In the form-level scripts for either platform, you can commit a specified table to storage by calling the CommitData global sub like this:
'save our updated table to storage right now
'using the CommitData global sub
CommitData(Tables("tablename").Index)

C. Additional Data Protection Strategies

If you want to add some additional data security to your application, you could copy data tables to an installed memory card (or in the case of the Palm Tungsten T5, the built in flash memory area, and in the case of the Palm LifeDrive, the built in hard drive). In that way, if the data files are lost due to whatever reason, they could be recovered from the memory card backup. There are many low cost or even free backup utilities available (Google for 'Palm memory card backup' for some examples), which are user-run utilities. To programmatically handle backups of data in your app to a memory card without user intervention, you can use the PalmDataPro SFTextFile extension for PalmOS (<http://www.palmdatapro.com/itm00116.htm>) or SFTextFile-PPC for PocketPC (<http://www.palmdatapro.com/itm00152.htm>).

Keywords: commit, save, write, cache, reset, dmsyncdatabase, commitdatabase, NVFS, storage, protection

KB ID: 10074
Updated: 2007-09-13

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Make PocketPC PDB Apps Close Faster

Problem: How To Make PocketPC PDB Apps Close Faster

Solution: Satellite Forms Windows Mobile/PocketPC applications that use the PDB database format (which is highly recommended over the CDB database alternative, see [How To use PalmDB \(PDB\) tables in a PocketPC application](#)) can experience delays when closing the application. This delay occurs when [cached table data is committed back to storage](#) as the application closes. There are some techniques you can employ to speed up the application closing process, explained below.

Overview: The key to speeding up the closing of PocketPC PDB applications is understanding how Satellite Forms uses Table Flags. Table Flags are attributes that you can assign to individual tables in your application that cause the SatForms runtime engine to treat the table in a special manner. The Table Flag is a numeric value that determines special behaviours of that table when it is in use on the PDA by the Satellite Forms runtime engine. For example, one of the possible table flag values indicates that the table is Read-Only, and the runtime engine therefore prevents any modifications/additions/deletions to the data in the table.

The Table Flags for each table are initially defined in the Table Editor (Layout tab) in App Designer, where you can set the available flags using checkboxes. *The NoAutoCommit flag option is new to the Table Editor in App Designer in SatForms 7.1, but support was added for this flag to the runtime engine and SFConvertPDB utility starting with SF Patch 70003.* When you compile your application, the table flag values corresponding to the selected options will be saved into the PDB table header. Additionally, flags can be specified using the -CreateFlag parameter of the SFConvertPDB utility. Each table can have different flags as they are assigned on a per-table basis. The current supported table flags include:

Flag value	Flag name	Flag description
0	No table flags	No special table behaviours - regular read/write table. The PocketPC runtime engine will automatically commit the table to storage when the app closes, regardless of whether any records were modified/created/deleted. This is the standard behaviour.
1	Backup	The table will be backed up at Hotsync (PalmOS behaviour only). This flag currently has no effect on the PocketPC platform.
2	Read-Only	The runtime engine will prohibit table record modifications/additions/deletions and will only permit read access to the table data. The PocketPC runtime engine will not commit read-only tables to storage on exit, and thus will close read-only tables more quickly than standard read/write tables.
4	Autoname	The desktop table name will automatically match the logical table name (this is the 'Link table name to filename' option in the App Designer table editor). This table flag is used by App Designer only and has no effect on the PDA.

64	NoAutoCommit	This is like a cross between a standard read/write table and a read-only table. The runtime engine treats this as a standard read/write table while the application is running, allowing you to modify/create/delete records. However, the table will not be automatically committed to storage when the app closes, thus any changes will not be saved. To save the data before closing, you must explicitly commit the table data to storage . This flag is implemented on the PocketPC platform only, and is ignored on the PalmOS platform (the PalmOS runtime treats it is a standard read/write table). The NoAutoCommit flag permits tables to close more quickly by not automatically committing them to storage when the app closes, like a read-only table.
----	--------------	---

Some table flag values can be combined, while others must be exclusive. For example, the Backup, Read-Only, and Autaname flags can all be present on a given table, and you would combine those flags by adding their flag values together. The Read-Only and NoAutoCommit flags must be exclusive to each other: do not combine them together.

Implementation: With an understanding of how the SatForms PocketPC runtime engine uses table flags, strategies to speed up the closing of applications become readily apparent.

1. The first and easiest approach is to make sure that any tables in your application that will never be modified are marked with the Read-Only table flag. The runtime engine will not commit read-only tables to storage when the app closes, because they logically should not have been modified in any way, thus read-only tables will close instantly. This is an especially important consideration for large information reference type applications (for example a medical reference database), because the length of time it takes to commit a table is directly proportional to the overall size of the table. It is also something to consider for smaller tables in non-reference type applications too, for example a list of non-modifiable items used for a droplist. Set the Read Only checkbox for the table in App Designer, and be sure to also set the proper -CreateFlag value if you use the [SFConvertPDB utility](#) in your desktop sync application.
2. The second approach is a bit more complex, and that is to use the NoAutoCommit table flag on tables that you may or may not make modifications to while using the application. A NoAutoCommit table may be modified by your application just like a regular read/write table, but the runtime engine will not save those modifications when your application closes. If you want to save the changes in that NoAutoCommit table, you MUST [commit the data using the Tables\("tablename"\).CommitData method](#) prior to the application closing. Since the runtime engine will not automatically commit the NoAutoCommit table data to storage when it is closed, it closes instantly just like a Read-Only table. NoAutoCommit tables therefore give developers an option that can reduce the delay when closing an app, in addition to the option of using Read-Only tables. For example, if your app includes 10 NoAutoCommit tables, but you have only updated 2 of those tables when your application goes to close, you can commit those changes on the two tables by calling the CommitData method, and leave the other 8 tables to close without being committed. Your application would then close more quickly than if all 10 tables were committed to storage on close. The added flexibility of using NoAutoCommit tables does come with added responsibility: you must explicitly handle the task of committing your modified data tables to storage if you want to preserve the changes.

Keywords: commit, table, database, PDB, speed, close, commitdata, SFConvertPDB, flag, createflag, read-only, noautocommit

KB ID: 10075
Updated: 2007-09-13

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How to Beam Files via IR or Bluetooth on PocketPC

Problem: How to Beam Files via IR or Bluetooth on PocketPC

Solution: It is possible to beam files via infrared or Bluetooth from a PocketPC handheld in your Satellite Forms application, by calling the standard PocketPC beam utility and passing it the path and name of the file you want to send.

To achieve this, use the SysUtils extension that was introduced with Satellite Forms 7.1, which includes an SU_LaunchApp function:

```
USAGE: result = SU_LaunchApp( strPath, strParam )
```

The system beam utility is "\Windows\beam.exe" for PocketPC 2003 and higher devices. For PocketPC 2002 devices, it is "\Windows\irsquirt.exe" (supporting infrared only). This is the value you need to pass in the strPath parameter for the SU_LaunchApp function.

The strParam parameter should contain the full path and name of the file you want to beam, for example "\My Documents\Work Order Sample\ESMS20000_WRKSITES.PDB".

An example to beam the WrkSites table would go like this:

```
dim strBeamUtilPath, strPDBPathname, result
strBeamUtilPath = "\Windows\beam.exe"
strPDBPathname = GetAppPath & "ESMS20000_WRKSITES.PDB"

result = SU_LaunchApp( strBeamUtilPath, strPDBPathname )
```

When this function is run, the standard Beam utility progress screen will appear ovetop of your application, listing the file to send and the devices found within range. The user must tap on the device they wish to send to from the list presented. When the transfer is done, the user needs to tap on the OK button in the upper right of the screen and that will close the Beam utility and return them to your application.

You can beam files to any accepting devices, including other PocketPC PDAs, PCs, PalmOS PDAs, mobile phones, etc.

Because Satellite Forms 7.x uses the same Palm Database PDB for application tables on both the PalmOS and PocketPC platforms, you can beam a PDB from a PocketPC to a PalmOS device and use it directly without any need for conversion.

When receiving a beamed file on a PocketPC device, files are placed into the \My Documents folder by default. If you need to have that file in a different folder on the PocketPC (for example your application folder), you'll need to devise a method to do that. [The PalmDataPro.com SFTxtFile-PPC extension (<http://www.palmdatapro.com/itm00152.htm>) does provide the capability to copy files programmatically.]

Note that this method is not applicable to the Windows CE OS, which does not have a standard beam utility like the Windows Mobile/PocketPC OS does.

Keywords: Beam, Send, Bluetooth, IR, Infrared

KB ID: 10084

Updated: 2008-05-16

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Sync Satellite Forms Data to a Linux Server With jSyncManager

Problem: How To Sync Satellite Forms Data to a Linux Server With jSyncManager

Solution: An open source Java based server sync solution called jSyncManager (<http://www.jsyncmanager.org>) is available for Linux and other Java-capable environments. Using the SatelliteForms contrib library written by Michael Schwarz of Technodane Software & Systems, this solution can be used to sync Satellite Forms data on Palm devices. This article by Michael Schwarz describes this solution.

Using the SatelliteForms contrib library

Michael Schwarz
TechnoDane Software & Systems, LLC
<http://www.technodane.com>

1. Introduction

I wrote a Java library to support reading and writing SatelliteForms data using an open source Java-based Palm sync server called jSyncManager. jSyncManager is a full platform for syncing PalmOS devices. It supports serial, USB, and network syncing. It has a single user GUI mode and a multi-user server mode. This document doesn't attempt to explain jSyncManager. Read other documentation and support information in the jSyncManager project for that. I will say that the way I set things up was to unpack the jSyncManager source code and I just added my conduit code to it. A conduit has to be in the org.jsyncmanager.Conduit package and has to extend AbstractConduit. One thing that threw me at first was that you have to use javax.swing even if you are running in "server" mode only because AbstractConduit requires you to write a method called "constructConfigPanel" (to support the GUI mode) that returns a JPanel. Mine just returns a null and it works both for the GUI and Server mode.

This library was written to support a specific client. We built a field force automation application for PalmOS using SatelliteForms 6.1.1 and we had over 100 users using network sync through jSyncManager using this library.

The library has been released under the LGPL and included in the "contrib" folder of the jSyncManager project. It was written by Michael A. Schwarz and is "owned" by his company, TechnoDane Software & Systems, LLC.

We are happy to provide assistance in using the library. Contact Michael at mschwarz at technodane.com. Answering specific questions we're happy to do for free. We also offer design and development services at competitive rates. We can also offer fixed bids.

2. Getting Started

Every class in the library has javadoc comments. The two most important classes for most users are SFTTable and SFColumn. The library is found in the "contrib" directory of the jSyncManager source tree. As of this writing, the library can only be obtained via CVS.

The first thing you need to know is how SatelliteForms data tables end up being named. You will need an official creator ID for your application. The data tables end up with the name Eccccvvv_SFTTableName, where "cccc" is the four character creator ID that you register with Palmsoft. "vvvv" is a table version number (which IIRC is set in SatelliteForms to allow you to differentiate between versions of your tables. I never took advantage of this feature, we always upgraded everyone at the same time).

Execution of your conduit starts at the startSync method. It is called with an instance of ConduitHandler, which is the class through which you drive the sync session. For example, we call the getUserInfo method on the ConduitHandler object to obtain the sync name of the handheld being synced. This was how we identified the user. Be aware that when working in server mode, several invocations of your conduit may be running simultaneously. We chose to create an inner class called "Session" to hold separate data structures for each active user. See jSyncManager project code and documentation for more details on concurrency issues.

Use the openDatabase method on the ConduitHandler to open the table you want to sync. This returns a handle you use for subsequent operations. Now we get to the meat of using the SatelliteForms library.

3. The SFTable class

PalmOS databases have a "App Block," which is a "one per database" structure. SatelliteForms uses this to store the structure of the data that is in the actual records of the database (in a traditional PalmOS application, records are just blocks of bytes that are usually mapped into a C struct). In an SF application, this AppBlock describes the names, types, and sizes of columns.

This is where our library starts to help you. You do not need to parse this structure, or encode and decode records yourself. Use the ConduitHandler getApplicationBlock method to obtain this AppBlock record. You then pass the resulting structure to the constructor for our SFTable class. It might look like this:

```
byte mode = DLPDatabase.READ_MODE;
mode |= DLPDatabase.WRITE_MODE;
mode |= DLPDatabase.SHOW_SECRET;
try {
    dbHandle = conduitHandler.openDatabase("EAIac0000_CUSTOMERS", mode);
} catch (Exception e) {
    // Intercept exception for tracing, then "re-throw" for
    // normal error handling.

    e.printStackTrace(System.out);
    DBUtility.logEvent(s.db, s.syncID, "ABOVE THROWN for DB [" + databases[x] + "]);
    throw e;
}

int count = conduitHandler.getOpenDatabaseInfo(dbHandle);
conduitHandler.postToLog("Database claims " + count + " records.");

DLPBlock appBlock = conduitHandler.getApplicationBlock(dbHandle);

SFTable sft = new SFTable(appBlock);
```

Now that you have the SFTable instance, you can use it to get and set data from rows of that table. There are a couple of common ways to examine records from the handheld. One is to iterate over all of them. The other is to look only at those marked "dirty," which means they were either created or modified on the handheld since the last sync.

If the table is always very small, iterating all of them may make sense. But it is usually much more efficient to examine only the dirty records. Here's how you do that:

```
try {
    DLPRecord dlpRec = chdlr.readNextModifiedRecord(dbHandle);
```

```
cols = sft.getColumns(dlpRec);
SFColumn custname = (SFColumn) cols.get("CUSTNAME");
SFColumn custaddr = (SFColumn) cols.get("CUSTADDR");
SFColumn custcity = (SFColumn) cols.get("CUSTCITY");
SFColumn custstate = (SFColumn) cols.get("CUSTSTATE");
SFColumn custzip = (SFColumn) cols.get("CUSTZIP");
cHdr.deleteRecord(dbHandle, (byte) 0, dlpRec.getRecordID());
} catch (NotConnectedException nce) {
    throw new OurConduitException(nce);
}
```

Note that everything is cast to SFColumn. This is the abstract base class of all the SatelliteForms datatypes. At this level, everything is a String. You call the SFColumn methods getValue or setValue to read and write the data.

Details to keep in mind

- Think of SFTable as a buffer. Getting and setting values doesn't have any real effect until you actually put something in it from a DLPRecord or get something out of it as a DLPRecord.
- Another thing to keep in mind. The getValue method is always going to return the value it was given by the getColumns(DLPRecord) method. In other words, calling setValue doesn't change what will be returned by getValue. The setValue method only affects what you get out of the SFTable.getDataRecord method. This was driven by the requirements of the application for which this library was developed.
- You can also call setValue on SFTable. There you provide the key and the value. Once you have set values in a column or columns, you can construct a DLPRecord that is ready to be written to the handheld with the SFTable.getDataRecord method.

4. Conclusion

This is a fairly light treatment of the jSyncManager SatelliteForms library. We hope to improve the documentation with your help. However, be aware that we do have Javadoc comments on our source code which we always try to improve. And the code itself is, of course, the most accurate documentation you could want. Contact us with your comments and questions. We would love to see other people get use out of this library.

Keywords: jSyncManager, Java, Linux, Unix, open source, Technodane, Michael Schwarz

KB ID: 10085
Updated: 2008-05-16

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Use the MSR Attachment with Janam XP Scanners in Satellite Forms

Problem: How To Use the MSR Attachment with Janam XP Scanners in Satellite Forms

Solution: A Magnetic Stripe Reader (MSR) attachment is available for the Janam XP20/XP30 PalmOS barcode scanners. Use the following instructions to utilize the Janam MSR attachment within your Satellite Forms applications.

1. The Janam MSR attachment is a serial port device, and you communicate with the MSR via the serial port. The simplest way to utilize the Janam MSR in Satellite Forms is to use the Serial Bar Code control (also known as the Bar Code Reader control). This control is not the same as the Symbol Integrated Scanner control used to control the barcode scanner on Janam and Symbiol PalmOS scanners, but does present similar options and events. Using the Bar Code Reader control to manage the MSR attachment, a "scan" event is fired when the MSR reads a card, providing you easy access to the swiped card data. You can test this with the sample project named "BarCode" (in the \Satellite Forms 7\Samples\Projects\Serial BarCode folder).
2. An alternative to using the Bar Code Reader extension would be to read the serial port data directly using the SerialPort extension. The Janam MSR is a transmit-only device that communicates with the handheld on Port 0 at 9600bps N-8-1.
3. In order for the MSR to work, it must be powered by the Janam scanner. The 5V Power Out feature on the scanner must be enabled in order to power the MSR. There are two ways that you can achieve this:
 - o In the Palm Prefs > Settings panel, set External 5V Power to "Always On". You should notice the green light on the MSR blink about once per second when you do this, indicating that the reader is powered up and ready to read a card.

OR

- o Starting with Satellite Forms 7.2, a new JanamUtils extension is included that provides access to manage some of the Janam-specific hardware features like keypad backlight, LEDs, Bluetooth power, and the 5V Power Out setting. Using the JXP_5VPowerOut function, you can toggle the 5V Power Out setting on or off from within your application. Therefore, you can extend battery life by enabling the power to the MSR only when you need it, and by turning off the 5V Power Out when it is not needed.

Keywords: MSR, Janam, XP, card, Magnetic Stripe Reader, barcode, Bar Code Reader, serial, 5V Power Out

KB ID: 10086

Updated: 2008-05-16

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-o-

How To Insert New Records Into a Sorted Table

Problem: How To Insert New Records Into a Sorted Table

Solution: For optimal performance when working with tables that have large numbers of records (for example a table of 10000 barcoded inventory items), it is advantageous to keep the table records in sorted order, so that the Tables().BinarySearch function can be used to locate a specific record quickly. If you create a new record in that table, a standard approach is to then re-sort the table using Tables().QuickSort so that the new record is placed into the correct sorted position, once again enabling the use of the BinarySearch function to locate records quickly. The QuickSort function takes more and more time as the number of records increases. This article describes an alternate approach to keep your table sorted when adding new records without requiring a QuickSort, thereby yielding even better performance.

The general approach for this method is to (1) find where the new record should go into the sorted table first, (2) create the new record at the end of the table, then (3) move that new record into the correct sorted position.

How do we easily determine where the new record should go in the sorted table? Simple: the BinarySearch function will tell us! The BinarySearch function returns True or False to indicate whether a match was found. If a match is found, the RowNum parameter will be set to the row number where the match was found, but if a match is not found, then RowNum will be set to the row number where a match *should* be found if there was one! This is the correct sorted position for that item if it is added to the table. That is where the item would go if it was added to the table and then a QuickSort was called. We can take advantage of that information to put a new record into the right location without needing to sort the table again.

Let's use a simple example where you are taking inventory by scanning barcodes of items on a shelf. If that item already exists in the database, then scanning it will increment the quantity by one. If that item is not located in the database, then a new record is created and the quantity is set to 1. A real inventory application might be a little more complex, but this will do for the illustration of the concept.

We want the best performance when taking inventory, so we'll keep our item database sorted on the barcode number, allowing us to use the fast BinarySearch to locate scanned items in the table. So, we scan in the item barcode, then call the update/create script code in our barcode handler event, like this:

```
'search for the matching item (in strBarcode variable) in our database (ItemCode field in tItems table)
'BinarySearch on sorted table yields best search performance
dim bFound, RowNum
bFound = Tables("tItems").BinarySearch("ItemCode", true, strBarcode, RowNum)

if bFound = True then
    'position to the matching row in table
    Tables("tItems").Position = RowNum
    'increment item quantity (ItemQty field)
    Tables("tItems").Fields("ItemQty") = Tables("tItems").Fields("ItemQty") + 1
    'play loud boop-beep audible confirmation tone
    Tone(1500, 75, 64)
    Tone(2500, 75, 64)
else
    'create new item record and set Qty to 1
    Tables("tItems").CreateRecord
    Tables("tItems").MoveLast
    Tables("tItems").Fields("ItemCode") = strBarcode
    Tables("tItems").Fields("ItemQty") = 1
```

```
'move new record into correct sorted position without needing to QuickSort!  
Tables("tItems").MoveRecord(RowNum, Tables("tItems").Position)  
'play loud boop-bee-bee-beep audible confirmation tone  
Tone(1500, 75, 64)  
Tone(2500, 50, 64)  
Tone(2500, 50, 64)  
Tone(2500, 75, 64)  
endif
```

Since you've either updated an existing item quantity or added a new item and placed it into the right sorted position, there is no need to sort the table again, and you can quickly get on with the job of scanning the next item.

Keywords: QuickSort, sort, BinarySearch, search, find, locate, speed, performance, MoveRecord, inventory

KB ID: 10087
Updated: 2008-06-26

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Use Google Maps for Windows Mobile from your Satellite Forms application

Problem: How To Use Google Maps for Windows Mobile from your Satellite Forms application

Solution: This article describes how to have your Satellite Forms application for Windows Mobile launch Google Maps with a specific destination (address or landmark or GPS lat/lon position), business search, or route request!

This method lets you show the desired location in Google Maps Mobile for WM (let's call it GM for short), and be able to have live interaction to move around, zoom, etc., just as if you had started GM manually. Your app stays running in the background, so when you close GM, your app returns to the foreground.

The techniques described here are demonstrated in the sample project GoogleMaps, in the PocketPC target, included with Satellite Forms 8. That sample also demonstrates the GoogleMaps extension for Palm OS, which provides similar features.

The key to using GM from your Satellite Forms Windows Mobile app is to launch GM with a specially formatted commandline that tells it to show a location, search for a business, or plot a route.

1. Launch GM using the SU_LaunchApp function in the SysUtils extension, providing the full path to the GoogleMaps.exe program in the strPath parameter, normally: \Program Files\GoogleMaps\GoogleMaps.exe
2. Pass the special commandline parameters for the location, business search, or route in the strParam parameter. The parameter string you pass should take this form:

Maps Usage Type	Parameter Format	Example
Location	-URL "?action=locn&a= <i>[location]</i> "	-URL "?action=locn&a=1 Microsoft Way Redmond WA"
Business Search	-URL "?action=busi&q= <i>[search]</i> "	-URL "?action=busi&q=pizza near wall st new york ny"
Route Request	-URL "?action=rout&start= <i>[place]</i> &end= <i>[place]</i> "	-URL "?action=rout&start=Vancouver BC&end=@latlon:50.34,-112.23"
View mode [optional]	&view= <i>[mapv/satv]</i>	&view=mapv for map view mode, &view=satv for satellite view mode

The parameter string includes several parts.

A. To show a Location, use these guidelines:

?action=locn

- action=locn means the action is to show a Location, with the location to follow next:

&a=1 Microsoft Way Redmond WA

- the "a" means "address" and the address in this example is 1 Microsoft Way Redmond WA
- the address can be a street address or city or even a landmark (eg. Statue of Liberty)
- the address can also be a GPS latitude/longitude, specified in decimal degrees format with a prefix of @latlon: for example a=@latlon:50.34,-112.23
- the address can also be a special string "@GPS" which means current GPS coordinates
- an optional View mode can be specified where you can choose the map view or satellite view
 - &view=mapv specifies map view
 - &view=satv specifies satellite view
- example sat view param string is "?action=locn&view=satv&a=Vancouver BC"

B. To search Google Maps for businesses, use these guidelines:

?action=busi

- action=busi means the action is to search for businesses, with the location and search term to follow next:
 - &q=pizza near wall st new york ny
- the "q" means "query" and the search query in this example is pizza near wall st new york ny
- use a natural language search query that combines something to search for in or near somewhere
- as with the Location action, search queries can be specified using a street/city address, landmark, GPS lat/lon coordinates using the @latlon: prefix, or the special "@GPS" keyword which means current GPS coordinates

C. To show a Route between two points, use these guidelines:

?action=rout

- action=rout means the action is to show us a Route instead of a location, with the start & end points to follow next:
 - &start=Vancouver BC&end=Whistler BC
- use "start=" and "end=" to specify the beginning and ending locations for the route
- as with the Location action, Start and End addresses can be specified as a street/city address, landmark, GPS lat/lon coordinates using the @latlon: prefix, or the special "@GPS" keyword which means current GPS coordinates

- example route combining address and GPS location is "?action=rout&start=Vancouver BC&end=@latlon: 50.34,-112.23"
- example route combining address and current GPS location is "?action=rout&start=Chicago&end=@GPS"

D. General tips:

- remember to begin and end the param string with a double quote char to make sure any spaces in the params are handled properly
- if you are building the strparam string, you might want to use chr(34) to put the quote chars into the string, like this:

```
dim strParam
strParam "-URL " &chr(34) &"?action=locn&a=Vancouver BC" &chr(34)
```

- to use this solution, you must of course also install Google Maps for Windows Mobile on the handheld, which you can obtain from <http://google.com/gmm>
- you can query the registry using SU_RegReadKey to find the location of the GM exe file, rather than hardcoding, in this reg key: HKLM\Software\Google\GoogleMaps\InstallPath

Keywords: Google Maps, map, satellite, route, directions, search, business

KB ID: 10088

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Use a Specific Connection using ConnectionMgr

Problem: How To Use a Specific Connection using ConnectionMgr

Solution: An improvement to the ConnectionMgr extension for Windows Mobile allows you to find the available connections on a device, and select which connection to use, instead of just taking the default as selected by the system.

The ConnectionMgr extension for Windows Mobile is used to request that the system initiates a network connection, so that a network resource can be used. Sometimes, on devices with multiple types of connections available, you may wish to specify which connection to use, instead of accepting the default connection selected by the system. The new CM_GetConnectionName and CM_ConnectByIndex functions provide this capability.

In order to choose a specific connection, you first need to find out what the available connections are to choose from. To do this, use the CM_GetConnectionName function in a loop until you have found all of the connections. The system stores available connections by name, with a unique index number for each one. Start by requesting the name of the connection with an index of 0, store that name, then increment the index and retrieve the next name, until you have retrieved all available connections. Once you have the list of connections and their indexes, you can initiate the connection using the index of the connection you want to use.

For example (lists all connections in a paragraph control named edConnName):

```
dim connidx, connname, done
connidx = 0
done = false
while not done
    connname = CM_GetConnectionName( connidx )
    if connname = "" then
        done = true
    else
        edConnName = edConnName &connidx &" - " &connname &chr(10)
        connidx = connidx + 1
    endif
wend
```

Sample output might look like this:

```
0 - My Work Network
1 - My ISP
2 - Work
3 - Secure WAP Network
4 - The WAP Network
5 - The Internet
```

Now, once you have the connection index of the connection you want to use, you can initiate that connection using CM_ConnectByIndex, like this:

```
'connect using The Internet
result = CM_ConnectByIndex(5)
```

Here's a more complete sample that will search all connections for one named "The Internet" and either connect with it or display an error message:

```
'connect to "The Internet"
dim connidx, connname, done
connidx = 0
done = false
while not done
    connname = CM_GetConnectionName( connidx )
    if connname = "" then Exit While
    if connname = "The Internet" then
        done = true
        Exit While
    else
        connidx = connidx + 1
    endif
wend
if done then
    result = CM_ConnectByIndex(connidx)
else
    MsgBox("The Internet connection not found.")
endif
```

Keywords: connection, dialup, WLAN, WWAN, internet

KB ID: 10089

Updated: 2010-06-21

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Enable Newer 2D Barcode Types on Janam Scanners

Problem: How To Enable Newer 2D Barcode Types on Janam Scanners

Solution: The Janam XP (Palm OS) and XM (Microsoft Windows Mobile OS) barcode scanners are designed to be compatible with scanning software written to the Symbol scanner APIs, enabling easy hardware migration. Thus, in Satellite Forms, the Symbol Integrated Scanner control can be used to manage barcode scanning on both Symbol and Janam scanners.

The Janam XP & XM scanners are also capable of scanning several newer 2D barcode types that were not included in the design of the Symbol Integrated Scanner control, including Data Matrix, Aztec Code, QR Code, and Maxicode. This article explains how to enable or disable those newer barcodes types in your scanning application.

NOTE: The capability to decode 2D barcodes on Janam scanners is a paid option, and is unlocked with a password that must be entered into the device. In order to decode 2D barcodes as described in this article, you need to unlock the 2D barcode capability on the Janam scanner first. Consult <http://www.janam.com> for more details on the 2D barcode unlock feature if needed.

Enabling 2D Barcodes with the SetBarExtra method

The Symbol Integrated Scanner control included with Satellite Forms version 8 has a new SetBarExtra method to let you enable or disable these newer 2D barcode types. The prototype for SetBarExtra looks like this:

```
result = Barcode1.SetBarExtra(bartype, enabled)
```

Where *bartype* is a barcode type number indicating the barcode type to enable or disable, and *enabled* is true to enable or false to disable.

So, now you are thinking that all you need to have is the bartype number for the symbology you want to enable, and you are all set. That is true, however, there is a catch: the barcode scanner manufacturers unfortunately decided to use different bartype codes on Palm OS powered scanners than on the Windows Mobile scanners. So, the bartype you need will differ depending on which OS platform your app is running on.

Here is the list of bartypes you can enable or disable with SetBarExtra.

Table of Barcode Types and Type Numbers for SetBarExtra

Barcode Type Name	Bartype Number for Palm OS (Janam XP)	Bartype Number for Windows Mobile (Janam XM)
* Enable/Disable ALL Barcode Types *	255	N/A
CODE39	0	55
UPCA	1	50
UPCEO	2	48
EAN13	3	53
EAN8	4	52

D2OF5	5	56
I2OF5	6	57
CODABAR	7	54
CODE128	8	60
CODE93	9	59
MSI_PLESSEY	11	51
UPCE1	12	49
TRIOPTIC39	13	66
UCC_EAN128	14	N/A
BOOKLAND	83	N/A
ISBT128	84	N/A
COUPON	85	N/A
Code32	86	70
AZTEC	192	34
CODE11	193	58
CODE49	194	36
COMPOSITE	195	86 or 87
DATAMATRIX	196	73
MAXICODE	197	72
MICROPDF	198	69
OCR	199	37
PDF417	200	64
POSTNET	201	97
QR	202	74
RSS	203	76
BPO	204	33
CANPOST	205	103
AUSPOST	206	101
IATA25	207	62
CODABLOCK	208	38
JAPOST	209	100
PLANET	210	98
DUTCHPOST	211	102
TLCODE39	212	88
MATRIX25	213	104
CHINAPOST	214	40
KOREAPOST	215	41
TELEPEN	216	42
CODE16K	217	43
POSICODE	218	44

USPS4CB	219	45
IDTAG	220	46
RSS_LIM	221	77
RSS_EXP	222	78
MSI	160	51
Plessey	161	39
WEBCODE	N/A	84
CUICODE	N/A	85
MACROPDF	N/A	71
STRT25	N/A	105
MESA	N/A	35

Keywords: barcode, Janam, bartype, Symbol, 2D, scanner

KB ID: 10090

Updated: 2010-06-22

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

How To Change The CreatorID of a Palm Extension PRC To Match Your App

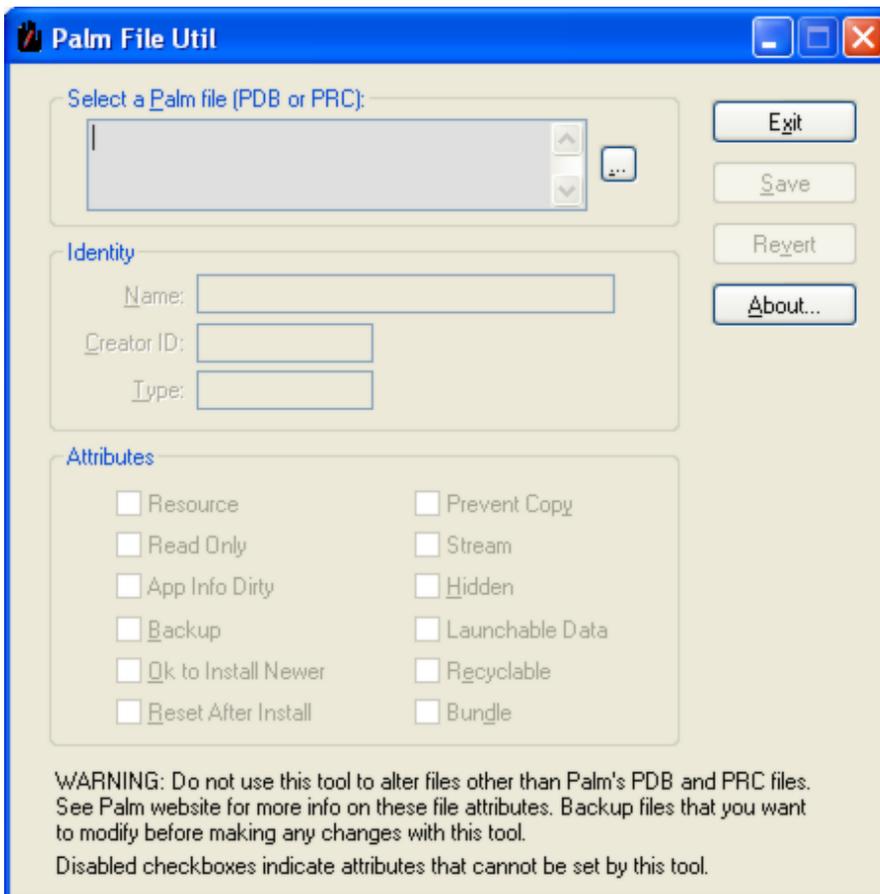
Problem: How To Change The CreatorID of a Palm Extension PRC To Match Your App

Solution: The Palm File Util tool included with Satellite Forms can be used to change the creatorID of an extension PRC file. If you use the new Integrated Runtime Engine feature of Satellite Forms 8, the runtime engine is changed to use your application creatorID. As a result, extension PRC files may appear on the Palm Info/Delete list, instead of being hidden by the runtime engine. To prevent this, you can change the creatorID of the extension PRC files to match your application, resulting in them being hidden on the Palm Info/Delete list.

When you use the integrated runtime engine feature in Satellite Forms 8, the extension PRC files used by your application are copied into the target's AppPkg folder, for easier deployment. Using the Palm File Util, you can change the creatorID of these extension PRC files in your AppPkg folder, without affecting the original extension PRC files in the \Satellite Forms 8\Extensions folder.

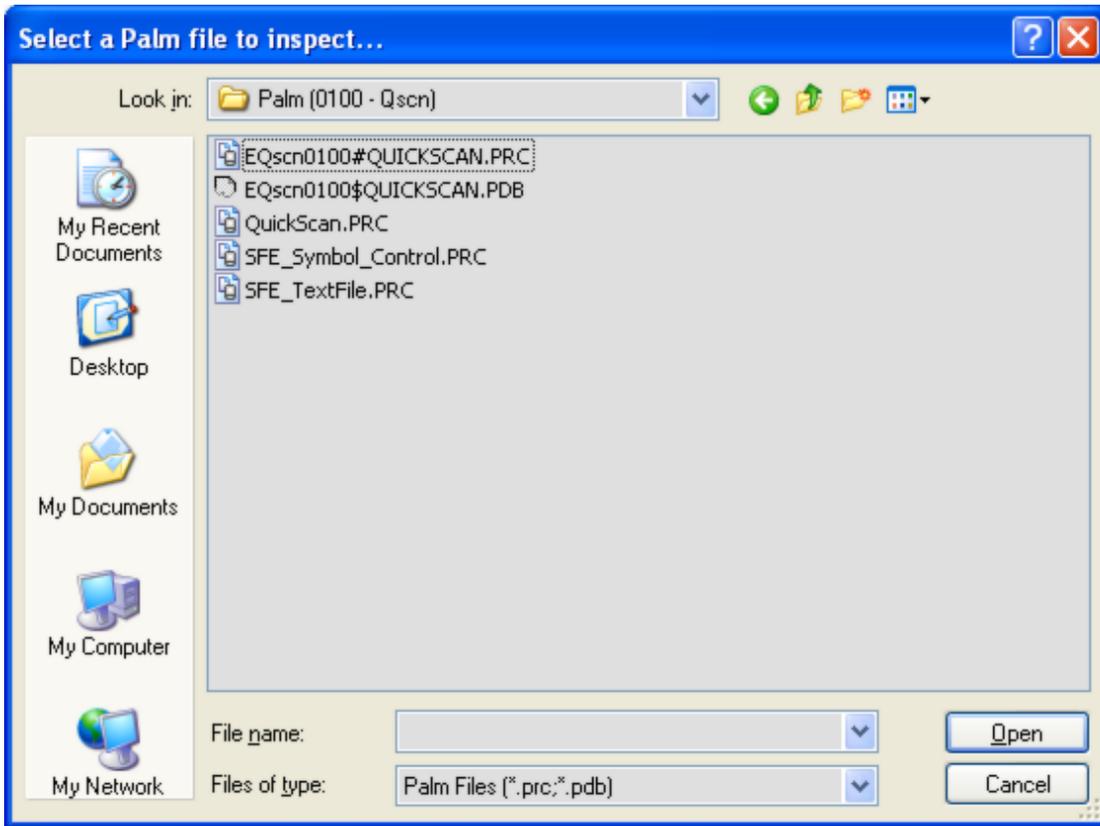
Here are the steps required.

1. To begin, launch the Palm File Util from the Windows Start menu on your development PC, via Start > Programs > Satellite Forms 8 > Palm File Util. You should see the Palm File Util dialog box like this:

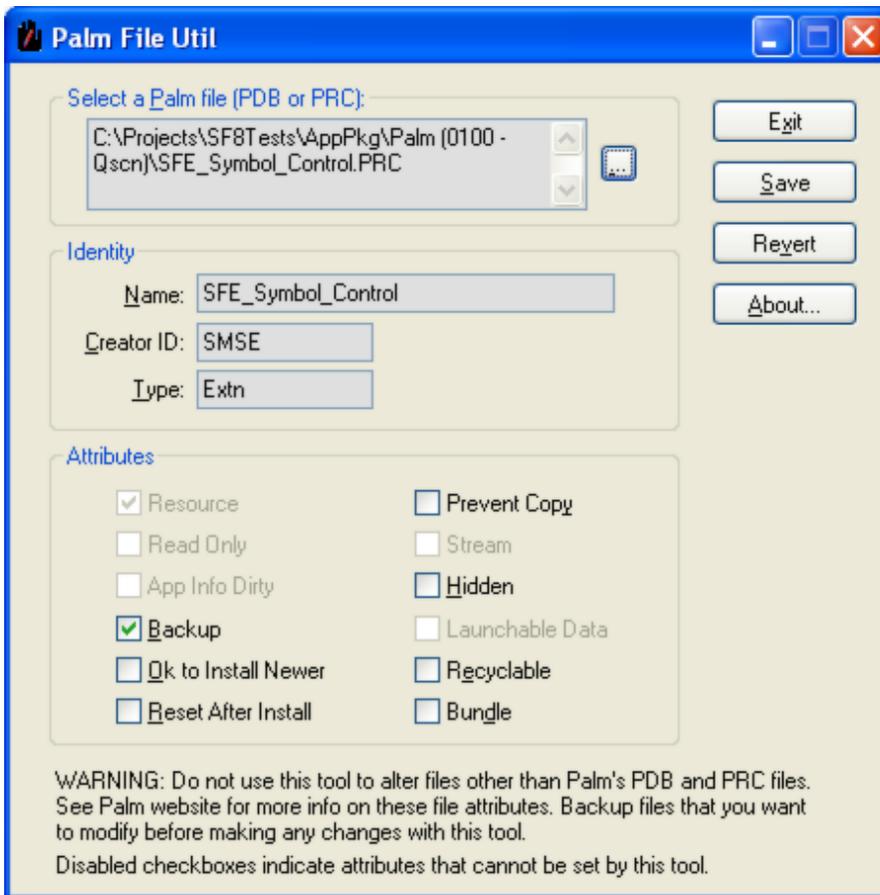


2. Next, click on the [...] button to browse for files. Locate the AppPkg target folder for your application, which should have copies of the extensions PRC files there from the build

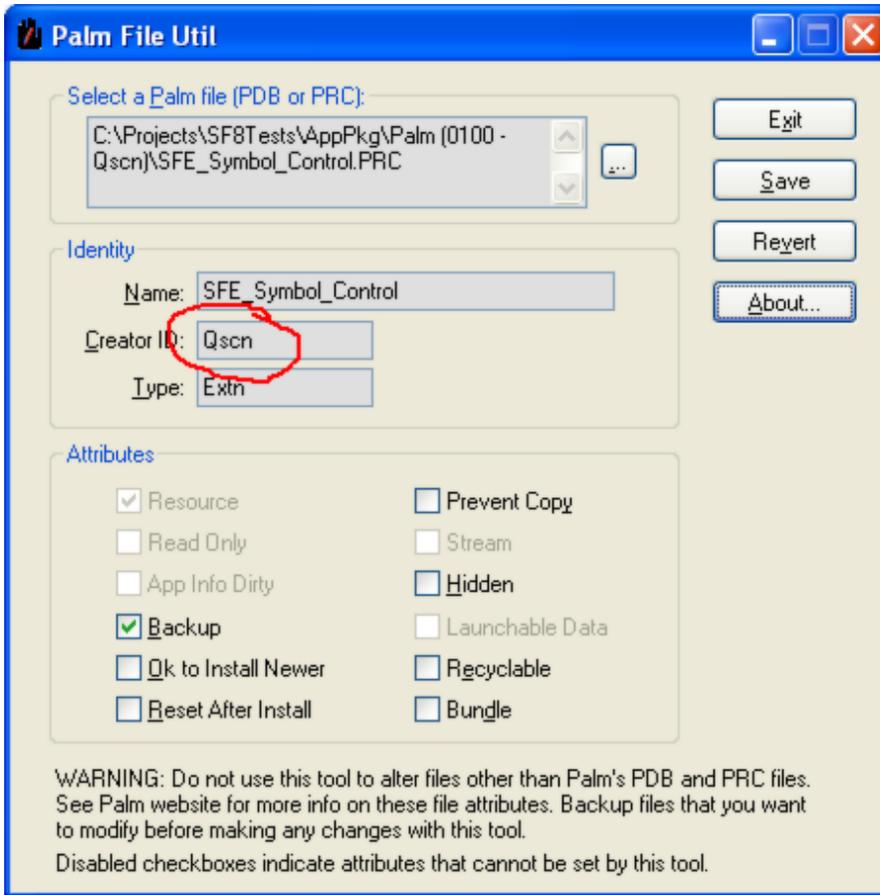
process. The browse window will look like this:



3. Select the first extension PRC file that you want to modify, in this example it is the SFE_Symbol_Control.PRC file. When you click on Open, the Filename, Identity, and Attributes information will be shown in the Palm File Util screen:



4. Now, change the Creator ID value to match (case sensitive) your application. You can find the creatorID from the Project Properties screen in MobileApp Designer. In this example, the creatorID of our application is Qscn, so we'll set that value like this:



5. There is no need to change any other values. Now, finally, click on the Save button to save the changes to the PRC file.
6. Repeat steps 2..5 for all remaining extension PRC files that you wish to modify.

That's it! Now those extension PRC files will remain hidden on the Palm Info/Delete lists.

Keywords: Palm, PRC, extension, info, delete, hidden, Palm File Util, creatorID

KB ID: 10091
Updated: 2010-06-29

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

QuickStart Guides

Barcode Scanning on Janam XP PalmOS Scanners

QuickStart Guide to: Barcode Scanning on Janam XP PalmOS Scanners

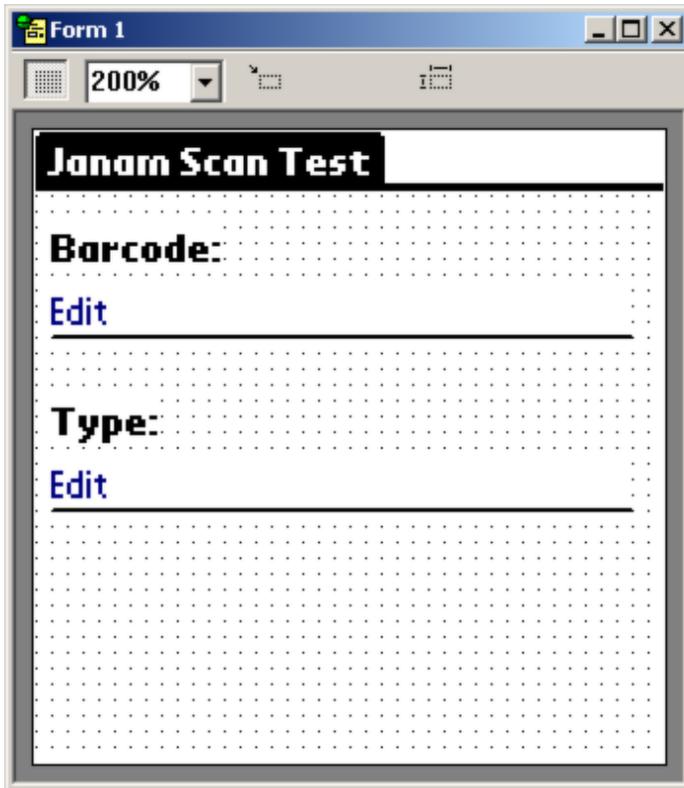
The Janam XP20 and XP30 scanners from [Janam Technologies](#) are rugged PalmOS handhelds with integrated barcode scanning capability. This QuickStart Guide shows you how to add Janam XP barcode scanning support to your Satellite Forms application, quickly and easily.

This QuickStart Guide contains plenty of screenshots to guide you through the process step by step, but don't let the length of this article scare you: the entire process of building the barcode scanning test application for the Janam XP scanners should only take about 15 minutes.

In order to control the Janam barcode scanner, we'll utilize the Symbol Integrated Scanner control extension that is included with Satellite Forms. Why the *Symbol* scanner control and not a *Janam* scanner control? Well, the Janam engineers brilliantly implemented their barcode support to be completely compatible with applications written to use the Symbol PalmOS barcode scanners, and therefore the Satellite Forms Symbol Integrated Scanner control works perfectly with the Janam scanners! This means that your application can support both the modern Janam XP scanners as well as the older Symbol SPT PalmOS scanners without any extra coding.

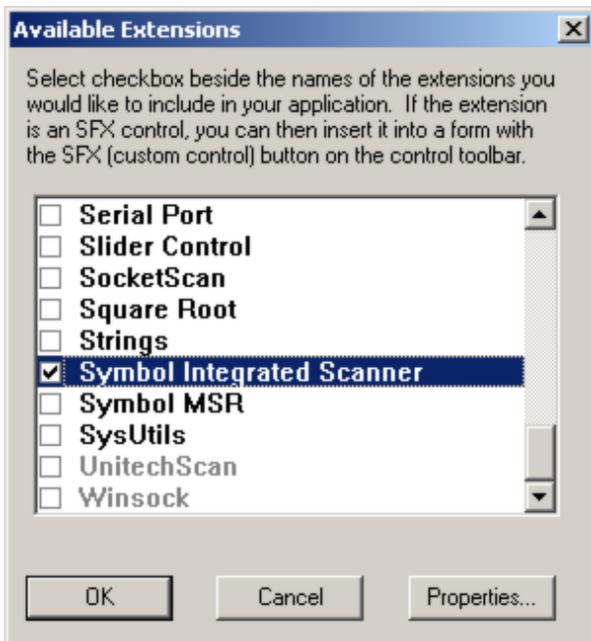
Okay, let's build a Janam Scan Test sample application step by step:

Step 1. Start a new project in Satellite Forms MobileApp designer, and select the default Palm platform target. A default form named Form 1 will be created, ready for you to add controls to. Click in the middle of the form, so that the Control Palette Toolbar becomes active. Let's add a form Title control, with the title text set to Janam Scan Test. Next add a text control with the label Barcode: and below that add an edit control named edBarcode that stretches across the width of the form. Let's add another text control below that labeled Type:, and below that another edit control named edBarcodeType that also stretches across the form. Your form should look like this:



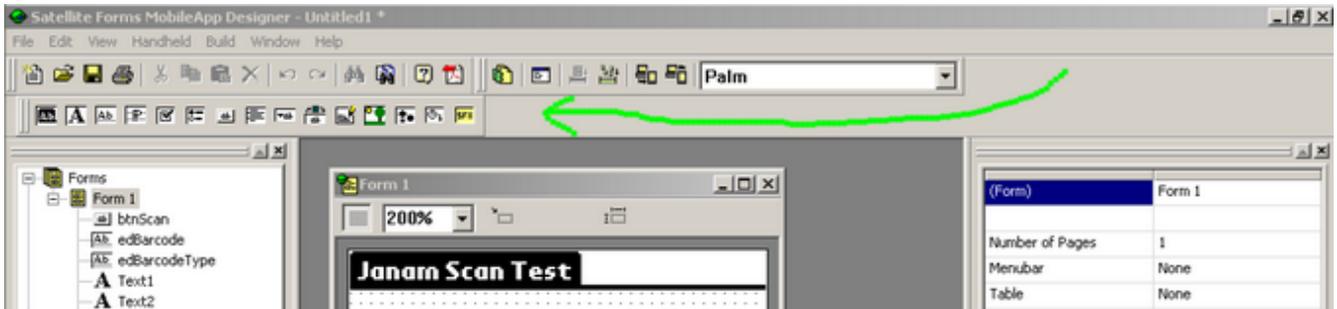
Step 2. The next step is to add the Symbol Integrated Scanner control extension to our project.

Click on the Manage Extensions toolbar icon which looks like this:  A list of available extensions is displayed. Do not select the one at the top named Bar Code Reader. Scroll down to the bottom of the list, and select the Symbol I ntegrated Scanner extension, then click OK:



Step 3. The Symbol Integrated Scanner extension is a custom SFX control, and not just a plugin script extension. Although it is a custom control, the control is not actually visible to the end user: it is only visible in form design view of MobileApp Designer. Because it is a custom control, an icon is added to the Control Palette toolbar so that you can click on the icon to add the control

to your forms. The SFX icon looks like this  and is added to the right end of the Control Palette toolbar. If you do not see it, there's a good chance that part of the Control Palette toolbar is simply being cut off because the MobileApp Designer window is not wide enough. That's easy to solve, we'll just grab the Control palette toolbar "handle" and drag it down to the next line in the MobileApp Designer desktop. Now we can see all the Control palette toolbar icons, like this:

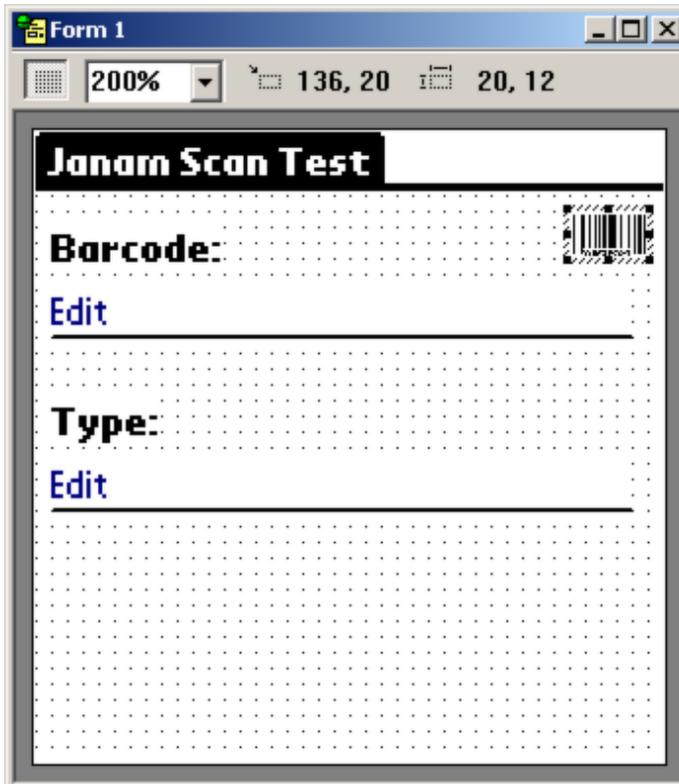


Okay good, we can see all the toolbar icons now.

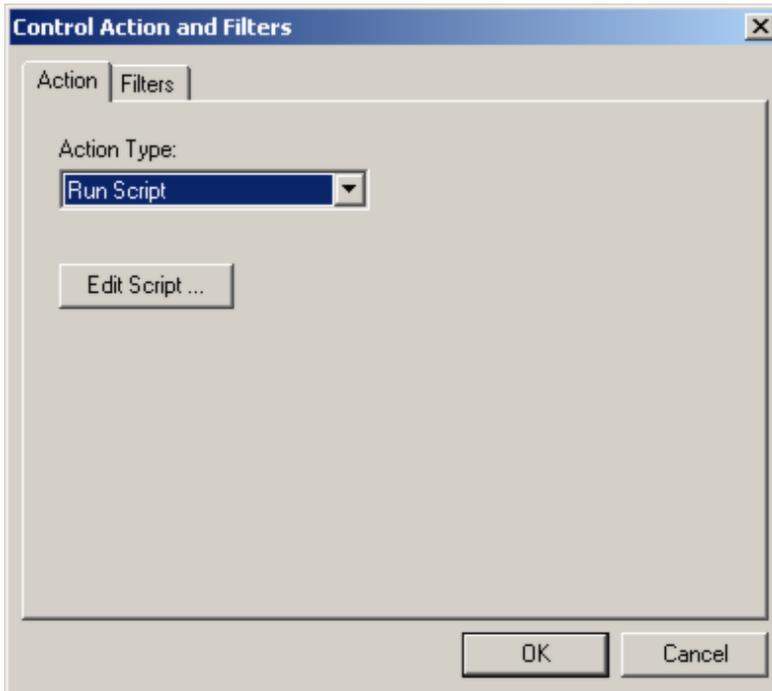
So, click on the SFX custom control toolbar icon so that we can add it to our form. It's possible you might have more than one custom control on a form and they all share the same toolbar icon, so a selection box appears to let you choose the custom control you want, like this:



Click OK to select it, and the control then appears on our form as a little barcode symbol. Remember, this custom control is not visible to the end user. Let's move it out of the way so it does not overlap other controls, so your form should now look like this:



The default name that appears for the scanner control is BarCode1, and we'll leave it at that default. With the scanner control selected, click the Edit Action button in the control properties, and select Run Script from the Action Type droplist. This is the OnClick script for the scanner control, and this event will be fired when a barcode is scanned.



Step 4. Alright, now we're getting into the thick of things: here is where we decide what to do when a barcode is scanned. Click on Edit Script and the blank script editor window will open up. In this script, we are going to use script methods of the scanner control to obtain the barcode data that was scanned and place it into the edBarcode edit control on the form. Let's display the barcode type into the edBarcodeType control as well, so we can see what kind of barcode was scanned. We'll play a high pitched confirmation beep to indicate the successful scan, or a low frequency buzz if the scan was not successful. Here's the script code to type in (feel free to copy & paste right from this QuickStart Guide article):

```
'the OnClick event is fired when a barcode is scanned
'if a good scan is received then put it into the
'edBarcode edit field else sound a failure tone

if BarCode1.TermRecd then
  'put barcode into edit control
  edBarcode = BarCode1.GetScan(0)
  'what type of barcode was it
  edBarcodeType = BarCode1.GetType
  'good read tone
  Tone(3800, 150, 64)
else
  'bad read tone
  Tone(220, 500, 64)
endif
```

Step 5. Okay, that is the bulk of it, but we need to do a couple more things to make barcode scanning work. What we need to do is to make sure the device we're running on is really a Janam (or Symbol) barcode scanner, and if so, enable the scanner to be used. We also need to disable the scanner when we leave the form. So, in the form AfterOpen event script that gets fired when the form is opened, enable the scanner like this:

```
'if device is Symbol or Janam scanner then
'enable scanner in the AfterOpen event and
'disable scanner in the BeforeClose event

if IsSymbolUnit = True then
  'enable scanning
  BarCode1.EnableScanner
else
  'warn user about no scanner
  edBarcode = "Barcode scanner not found"
endif
```

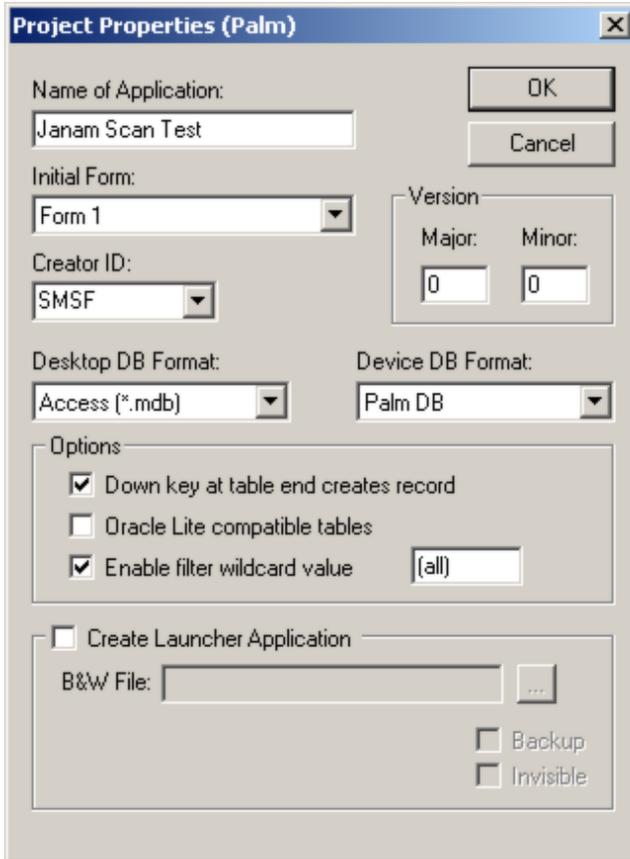
And in the form BeforeClose event, disable the barcode scanner like this:

```
'if device is Symbol or Janam scanner then
'disable scanner in the BeforeClose event

if IsSymbolUnit = True then
  'disable scanning
  BarCode1.DisableScanner
endif
```

That's about it for the barcode scanning stuff. Let's give the application a name, compile it, and test it out on the handheld.

Step 6. Click on the Edit > Project Properties menu, and give our application the name Janam Scan Test, like this:



Click OK, and then save your project. Name it Janam Scan Test.sfa.

Now, compile the application by pressing the F7 hotkey (Rebuild All). If the compiler finds any typos, fix them, and then Rebuild All again. Next, use the Handheld > Download App & Tables menu option (or press the F5 hotkey) to send the application to your Janam scanner device. Connect the Janam XP to the cradle or sync cable, and start the hotsync from the handheld. Assuming you've already installed the Satellite Forms SDK runtime engine on the Janam, your application will be sent to the device during the hotsync. Launch the SatForms SDK app on the handheld, select Janam Scan Test from the list, and you are ready to scan.

Point the scanner at a nearby barcode, press one of the front or side scan buttons on the Janam XP unit, and you the scanner should read the barcode and play a confirmation beep. Here's a sample screenshot from the Janam XP30 color screen:



Congratulations! You've just learned how to scan barcodes on Janam XP scanners with Satellite Forms!

There are plenty of additional functions in the scanner control to give you even more detailed control over the scanning process, but you've got the basic scanning function covered just by applying what you learned in the QuickStart Guide above.

Additional Sample Projects:

\Satellite Forms 7\Samples\Projects\Symbol Control
App\ScannerDemo\SimpleScannerDemo.sfa
\Satellite Forms 7\Samples\Projects\Symbol Control App\Barcode\SymbolBarcode.sfa
\Satellite Forms 7\Samples\Projects\Symbol Control App\Function
Test\SymbolFunctionTest.sfa

Keywords: Janam, XP, XP20, XP30, Symbol, barcode, scanner, quickstart, PalmOS

KB ID: 10077

Updated: 2008-05-27

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Barcode Scanning on Symbol Windows Mobile/PocketPC Scanners

QuickStart Guide to: Barcode Scanning on Symbol Windows Mobile/PocketPC Scanners

[Symbol Technologies](#), now a unit of Motorola, manufactures numerous models of Windows Mobile/PocketPC rugged handhelds with integrated barcode scanning capability. The Symbol SPT family of PalmOS powered barcode scanners is no longer manufactured, but there are thousands of those units in use and the same techniques shown here can also be applied to those SPT scanners. This QuickStart Guide shows you how to add Symbol Windows Mobile/PocketPC barcode scanning support to your Satellite Forms application, quickly and easily.

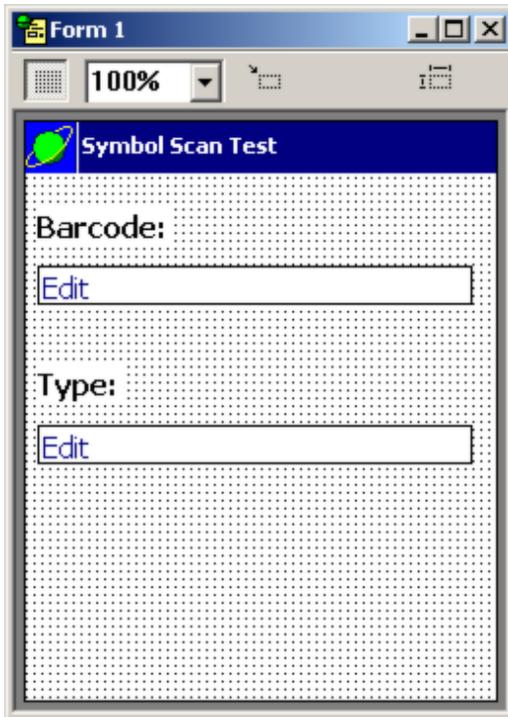
NOTE: The Symbol Integrated Scanner control demonstrated in this article can also be used to control barcode scanning on the Janam XM60 Windows CE scanner. [Janam Technologies](#) implemented their barcode scanning system so that it is compatible with the Symbol scanning functions, and therefore the same Symbol Integrated Scanner control in Satellite Forms can be used to control scanning on both Symbol scanners and Janam scanners!

This QuickStart Guide contains plenty of screenshots to guide you through the process step by step, but don't let the length of this article scare you: the entire process of building the barcode scanning test application for the Symbol PocketPC scanners should only take about 15 minutes.

In order to control the Symbol barcode scanner, we'll utilize the Symbol Integrated Scanner control extension that is included with Satellite Forms. This control is available for the Symbol devices on both the Windows Mobile/PocketPC and PalmOS platforms, so you can create cross platform barcode scanning applications with ease. This guide demonstrates the Windows Mobile/PocketPC platform only.

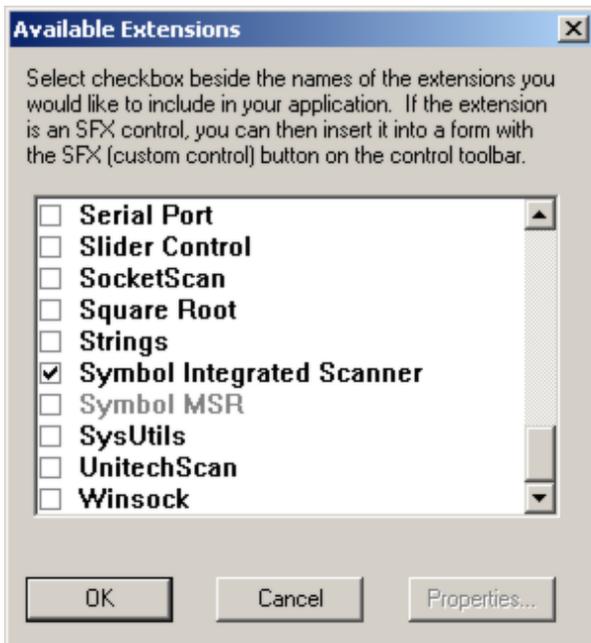
Okay, let's build a Symbol Scan Test sample application step by step:

Step 1. Start a new project in Satellite Forms MobileApp designer, and select the PocketPC platform target. A default form named Form 1 will be created, ready for you to add controls to. Click in the middle of the form, so that the Control Palette Toolbar becomes active. Let's add a form Title control, with the title text set to Symbol Scan Test. Next add a text control with the label Barcode: and below that add an edit control named edBarcode that stretches across the width of the form. Let's add another text control below that labeled Type:, and below that another edit control named edBarcodeType that also stretches across the form. Your form should look like this:



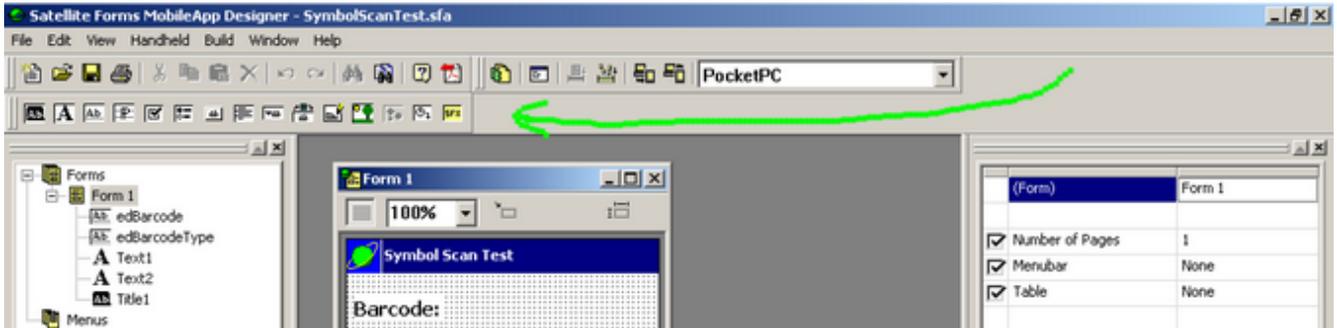
Step 2. The next step is to add the Symbol Integrated Scanner control extension to our project.

Click on the Manage Extensions toolbar icon which looks like this:  A list of available extensions is displayed. Do not select the one at the top named Bar Code Reader. Scroll down to the bottom of the list, and select the Symbol Integrated Scanner extension, then click OK:



Step 3. The Symbol Integrated Scanner extension is a custom SFX control, and not just a plugin script extension. Although it is a custom control, the control is not actually visible to the end user: it is only visible in form design view of MobileApp Designer. Because it is a custom control,

an icon is added to the Control Palette toolbar so that you can click on the icon to add the control to your forms. The SFX icon looks like this  and is added to the right end of the Control Palette toolbar. If you do not see it, there's a good chance that part of the Control Palette toolbar is simply being cut off because the MobileApp Designer window is not wide enough. That's easy to solve, we'll just grab the Control palette toolbar "handle" and drag it down to the next line in the MobileApp Designer desktop. Now we can see all the Control palette toolbar icons, like this:

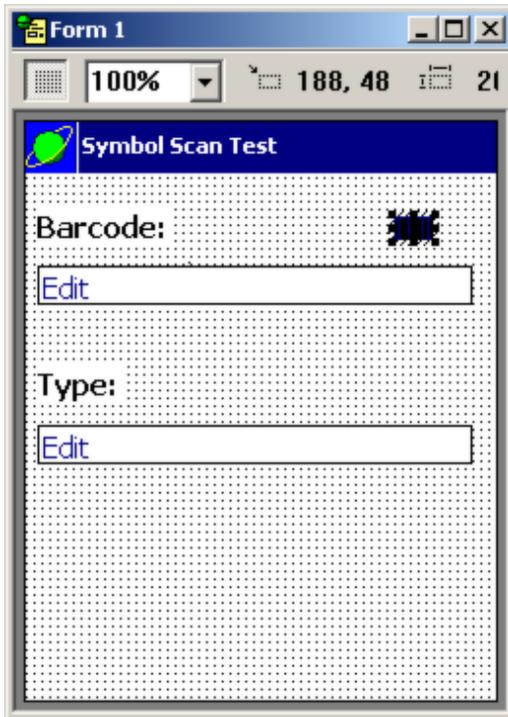


Okay good, we can see all the toolbar icons now.

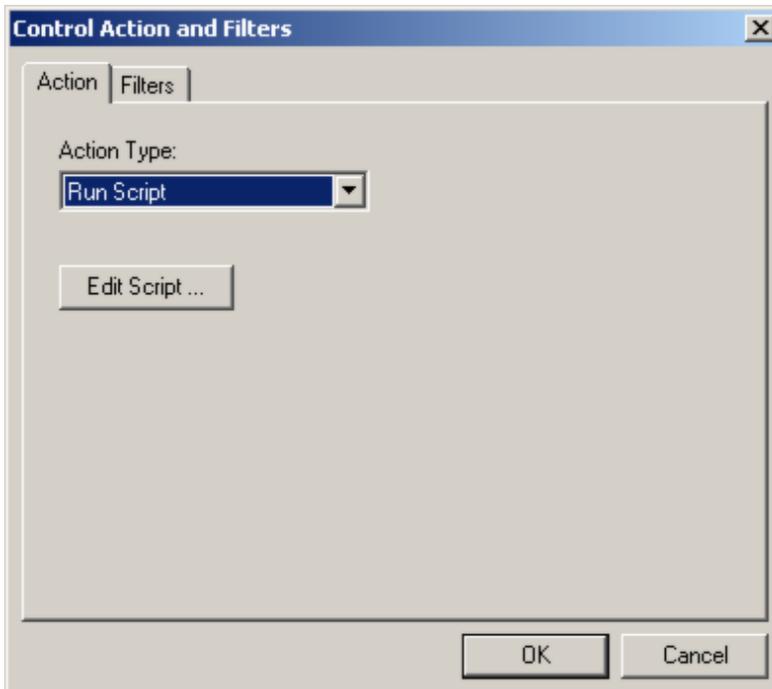
So, click on the SFX custom control toolbar icon so that we can add it to our form. It's possible you might have more than one custom control on a form and they all share the same toolbar icon, so a selection box appears to let you choose the custom control you want, like this:



Click OK to select it, and the control then appears on our form as a little barcode symbol. Remember, this custom control is not visible to the end user. Let's move it out of the way so it does not overlap other controls, so your form should now look like this:



The default name that appears for the scanner control is BarCode1, and we'll leave it at that default. With the scanner control selected, click the Edit Action button in the control properties, and select Run Script from the Action Type droplist. This is the OnClick script for the scanner control, and this event will be fired when a barcode is scanned.



Step 4. Alright, now we're getting into the thick of things: here is where we decide what to do when a barcode is scanned. Click on Edit Script and the blank script editor window will open up. In this script, we are going to use script methods of the scanner control to obtain the barcode data that was scanned and place it into the edBarcode edit control on the form. Let's

display the barcode type into the edBarcodeType control as well, so we can see what kind of barcode was scanned. We'll play a high pitched confirmation beep to indicate the successful scan, or a low frequency buzz if the scan was not successful. Here's the script code to type in (feel free to copy & paste right from this QuickStart Guide article):

```
'the OnClick event is fired when a barcode is scanned
'if a good scan is received then put it into the
'edBarcode edit field else sound a failure tone

if BarCode1.TermRecd then
  'put barcode into edit control
  edBarcode = BarCode1.GetScan(0)
  'what type of barcode was it
  edBarcodeType = BarCode1.GetType
  'good read tone
  Tone(3800, 150, 64)
else
  'bad read tone
  Tone(220, 500, 64)
endif
```

Step 5. Okay, that is the bulk of it, but we need to do a couple more things to make barcode scanning work. What we need to do is to make sure the device we're running on is really a Symbol barcode scanner, and if so, enable the scanner to be used. We also need to disable the scanner when we leave the form. So, in the form AfterOpen event script that gets fired when the form is opened, enable the scanner like this:

```
'if device is Symbol barcode scanner then
'enable scanner in the AfterOpen event and
'disable scanner in the BeforeClose event

if IsSymbolUnit = True then
  'enable scanning
  BarCode1.EnableScanner
else
  'warn user about no scanner
  edBarcode = "Barcode scanner not found"
endif
```

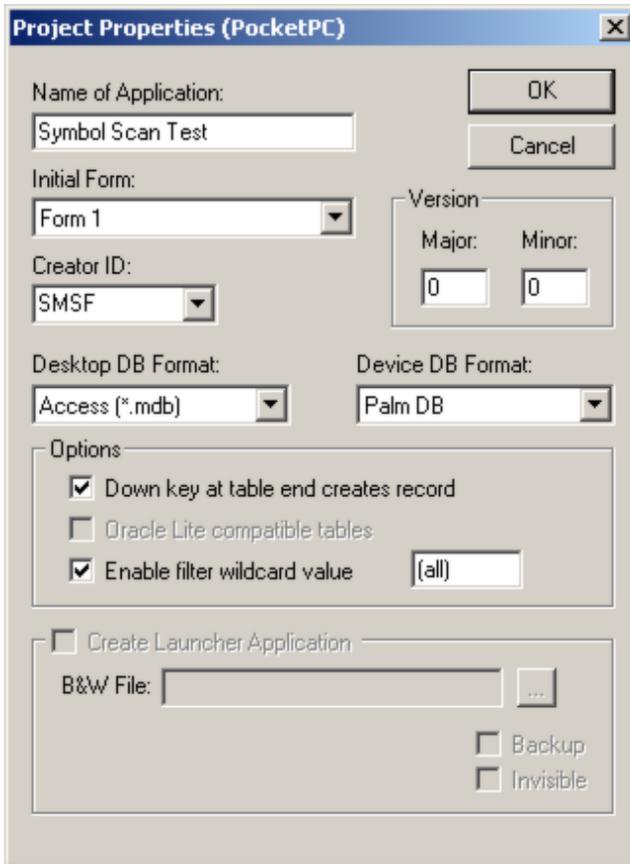
And in the form BeforeClose event, disable the barcode scanner like this:

```
'if device is Symbol barcode scanner then
'disable scanner in the BeforeClose event

if IsSymbolUnit = True then
  'disable scanning
  BarCode1.DisableScanner
endif
```

That's about it for the barcode scanning stuff. Let's give the application a name, compile it, and test it out on the handheld.

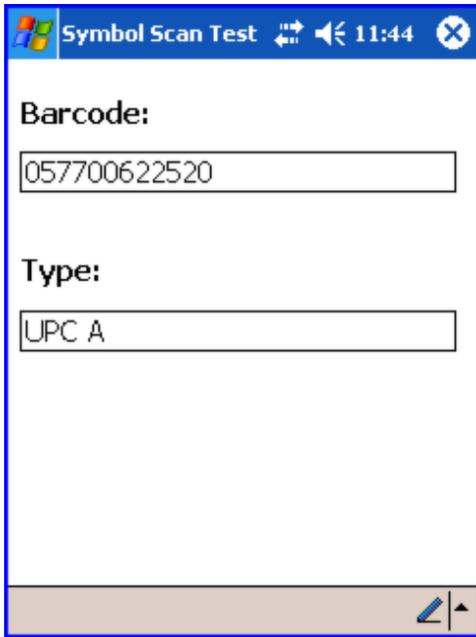
Step 6. Click on the Edit > Project Properties menu, and give our application the name Symbol Scan Test, like this:



Click OK, and then save your project. Name it Symbol Scan Test.sfa.

Now, compile the application by pressing the F7 hotkey (Rebuild All). If the compiler finds any typos, fix them, and then Rebuild All again. Next, connect the Symbol scanner to the cradle or sync cable, and use the Handheld > Download App & Tables menu option (or press the F5 hotkey) to download the application to your Symbol Windows Mobile/PocketPC scanner device. Launch the SatForms SDK app on the handheld, select Symbol Scan Test from the list, and you are ready to scan.

Point the scanner at a nearby barcode, press one of the scan buttons on the Symbol unit, and you the scanner should read the barcode and play a confirmation beep. Here's a sample screenshot from a Symbol MC50 screen:



Congratulations! You've just learned how to scan barcodes on the Symbol Windows Mobile/PocketPC scanners with Satellite Forms!

There are plenty of additional functions in the scanner control to give you even more detailed control over the scanning process, but you've got the basic scanning function covered just by applying what you learned in the QuickStart Guide above.

Additional Sample Projects:

 \Satellite Forms 7\Samples\Projects\Symbol Control
App\ScannerDemo\SimpleScannerDemo.sfa
 \Satellite Forms 7\Samples\Projects\Symbol Control App\Barcode\SymbolBarcode.sfa
 \Satellite Forms 7\Samples\Projects\Symbol Control App\Function
Test\SymbolFunctionTest.sfa

Keywords: Symbol, barcode, scanner, quickstart, PocketPC, Windows Mobile, MC50, MC70, Janam, XM60

KB ID: 10079
Updated: 2008-05-27

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Barcode Scanning on Aceeca Meazura PalmOS Scanners

QuickStart Guide to: Barcode Scanning on Aceeca Meazura PalmOS Scanners

The Aceeca Meazura MEZ1000 RDA (Ruggedized Digital Assistant) from [Aceeca International](#) is an ultra-rugged waterproof PalmOS handheld designed for use in tough environments. It is available with an optional barcode scanner module, with both CCD and laser scanner options. This QuickStart Guide shows you how to add Aceeca Meazura barcode scanning support to your Satellite Forms application, quickly and easily.

This QuickStart Guide contains plenty of screenshots to guide you through the process step by step, but don't let the length of this article scare you: the entire process of building the barcode scanning test application for the Aceeca Meazura scanners should only take about 15 minutes.

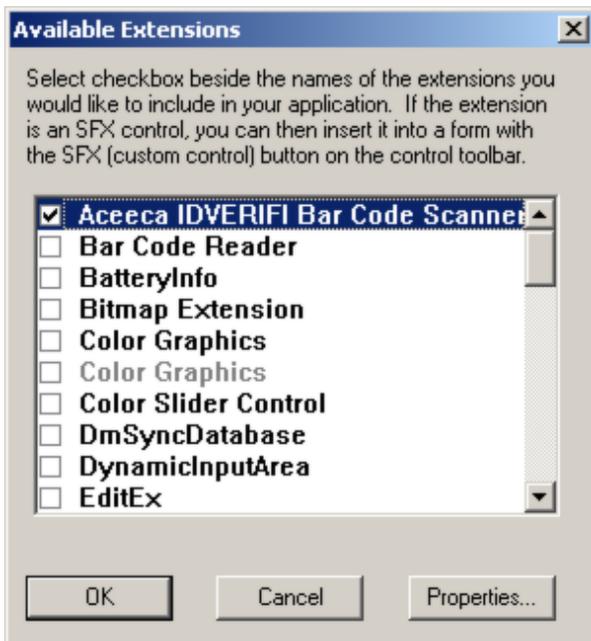
In order to control the Aceeca barcode scanner, we'll utilize the Aceeca IDVERIFI Bar Code Scanner plugin extension that is included with Satellite Forms. The extension works with either the Aceeca BCS1 CCD scanner module, or the BCS2 laser scanner module. We'll demonstrate the BCS2 laser scanner module here.

Okay, let's build an Aceeca Scan Test sample application step by step:

Step 1. Start a new project in Satellite Forms MobileApp designer, and select the default Palm platform target. A default form named Form 1 will be created, ready for you to add controls to. Click in the middle of the form, so that the Control Palette Toolbar becomes active. Let's add a form Title control, with the title text set to Aceeca Scan Test. Next add a text control with the label Barcode: and below that add an edit control named edBarcode that stretches across the width of the form. Let's add a button control named btnScan, labeled SCAN, using the Bold 12 font, and enlarge it to make it easy to tap on with a finger. Leave the button action set to No Action for now. Your form should look like this:



Step 2. The next step is to add the Aceeca IDVERIFI Bar Code Scanner plugin extension to our project. Click on the Manage Extensions toolbar icon which looks like this:  A list of available extensions is displayed. Select the Aceeca IDVERIFI Bar Code Scanner at the top of the list, then click OK:

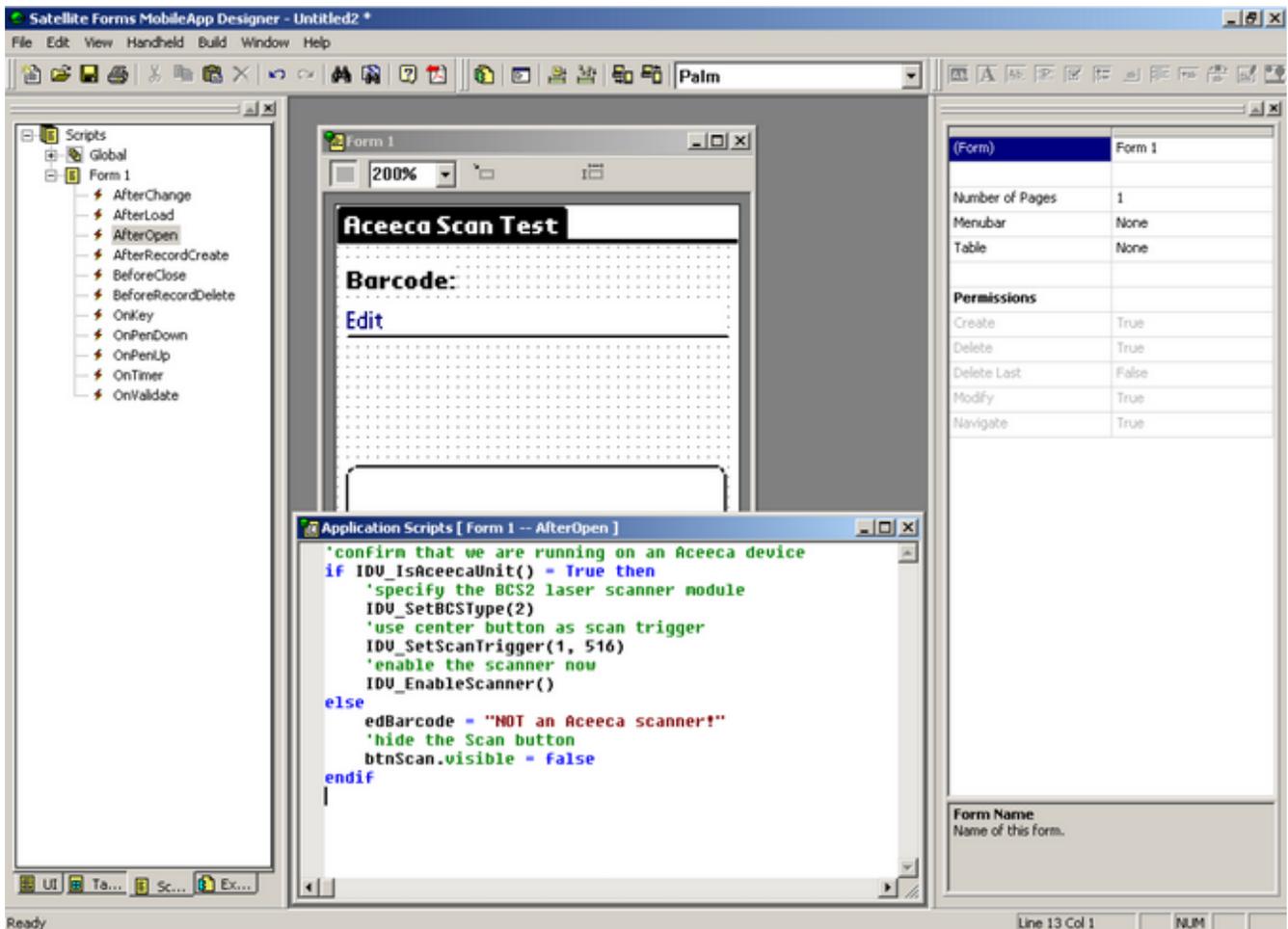


Step 3. The Aceeca scanner extension is a plugin script extension, and is not a custom SFX control. It adds functions to the Satellite Forms script language, but does not appear as a visual control on the form. We'll use these script functions to enable/disable the scanner, make certain hardware buttons trigger the scan, and obtain the data from the scanner when it reads a barcode.

Now we're getting into the thick of things: here is where we decide how to trigger the barcode scan, and what to do when a barcode is scanned. Let's begin by enabling the scanner in the AfterOpen event of the form. We also want to configure the extension so that the barcode scanner is triggered when the center button on the Aceeca Meazura is pressed. Click on the Scripts tab of the Workspace Palette, at the bottom left of the MobileApp Designer window. Click on the + symbol by Form 1 in the list to expand it to show all of the event and control scripts on the form. Select the AfterOpen event script, and the script editor window will appear. Type in this AfterOpen script:

```
'confirm that we are running on an Aceeca device
if IDV_IsAceecaUnit() = True then
  'specify the BCS2 laser scanner module
  IDV_SetBCSType(2)
  'use center button as scan trigger
  IDV_SetScanTrigger(1, 516)
  'enable the scanner now
  IDV_EnableScanner()
else
  edBarcode = "NOT an Aceeca scanner!"
  'hide the Scan button
  btnScan.visible = false
endif
```

Your screen should look like this:



Now, let's make sure that we disable the scanner when we close the form. Select the BeforeClose script from the list, and type in this script:

```

'confirm that we are running on an Aceeca device
if IDV_IsAceecaUnit() = True then
  'disable the scanner now
  IDV_DisableScanner()
endif

```

Step 4. Okay, now we need to add some script code to do something with the barcode data when it is scanned. The Aceeca Meazura scanner signals that a barcode has been scanned by posting a special MzBarcodeReceived virtual keypress (ASCII code decimal 7424) into the device's key queue. We need to monitor the key queue looking for that special virtual key, and grab the barcode data string when we see it. We'll play a high pitched confirmation beep to indicate the successful scan, or a low frequency buzz if the scan was not successful. Here's the OnKey event script code to type in (feel free to copy & paste right from this QuickStart Guide article):

```

Dim AscKey, Virtkey, Modkey
GetLastKey(AscKey, VirtKey, ModKey)

```

```

'watch for the MzBarcodeReceived virtual key (7424)
if AscKey = 7424 then
  'get the barcode data

```

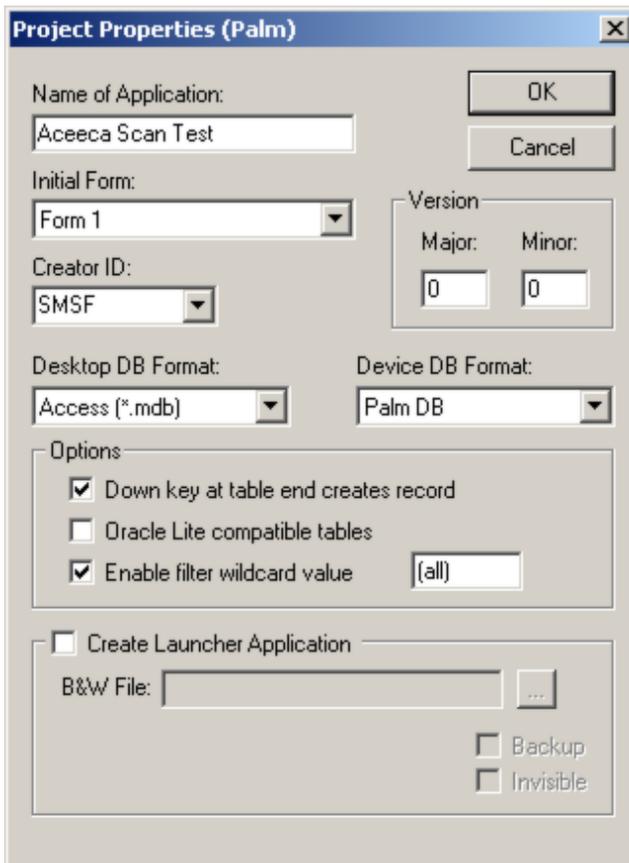
```
edBarcode = IDV_GetScan(5)
if edBarcode = "" then
  'barcode read failed
  edBarcode = "NO READ"
  'play failure tone
  Tone(400, 600, 64)
else
  'play good read tone
  Tone(3800, 100, 64)
endif
endif
```

Step 5. Okay, that is the bulk of it, but we still need to make the onscreen SCAN button trigger the scan, in addition to the center hardware button. Select the SCAN button on the form, and click on Edit Action in the control properties. Select Run Script from the list of actions, then click on Edit Script. Type in this script to trigger the scan when the button is tapped:

```
'trigger the barcode scan
IDV_Trigger(True)
```

That's about it for the barcode scanning stuff. Let's give the application a name, compile it, and test it out on the handheld.

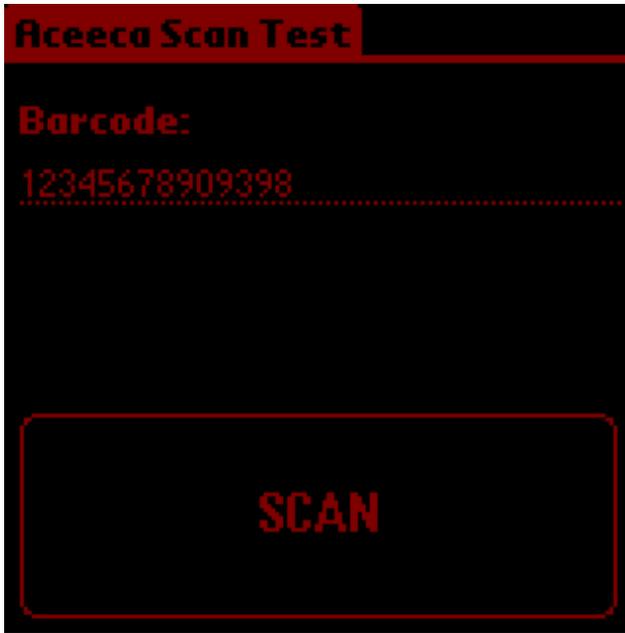
Step 6. Click on the Edit > Project Properties menu, and give our application the name Aceeca Scan Test, like this:



Click OK, and then save your project. Name it Aceeca Scan Test.sfa.

Now, compile the application by pressing the F7 hotkey (Rebuild All). If the compiler finds any typos, fix them, and then Rebuild All again. Next, use the Handheld > Download App & Tables menu option (or press the F5 hotkey) to send the application to your Aceeca Meazura device. Connect the Meazura to the cradle or sync cable, and start the hotsync from the handheld. Assuming you've already installed the Satellite Forms SDK runtime engine on the Meazura, your application will be sent to the device during the hotsync. Launch the SatForms SDK app on the handheld, select Aceeca Scan Test from the list, and you are ready to scan.

Point the scanner at a nearby barcode, press center hardware button or the onscreen SCAN button on the Meazura, and the scanner should read the barcode and play a confirmation beep. Here's a sample screenshot from the Meazura after reading a barcode:



Congratulations! You've just learned how to scan barcodes on the Aceeca Meazura scanner with Satellite Forms!

There are plenty of additional functions in the scanner control to give you even more detailed control over the scanning process, but you've got the basic scanning function covered just by applying what you learned in the QuickStart Guide above.

Additional Sample Projects:

 \Satellite Forms 7\Samples\Projects\IDVERIFI Barcode\IDVScanTest.sfa

Keywords: Aceeca, Meazura, MEZ1000, IDVERIFI, barcode, scanner, quickstart, PalmOS

KB ID: 10080

Updated: 2007-11-30

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Barcode Scanning on Intermec Windows Mobile/PocketPC Scanners

QuickStart Guide to: Barcode Scanning on Intermec Windows Mobile/PocketPC Scanners

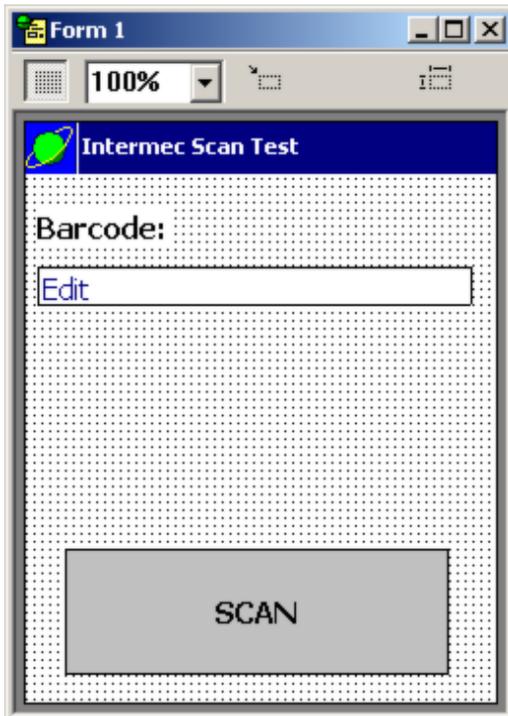
[Intermec](#) manufactures several models of Windows Mobile/PocketPC rugged handhelds with integrated barcode scanning capability. This QuickStart Guide shows you how to add Intermec Windows Mobile/PocketPC barcode scanning support to your Satellite Forms application, quickly and easily.

This QuickStart Guide contains plenty of screenshots to guide you through the process step by step, but don't let the length of this article scare you: the entire process of building the barcode scanning test application for the Intermec PocketPC scanners should only take about 15 minutes.

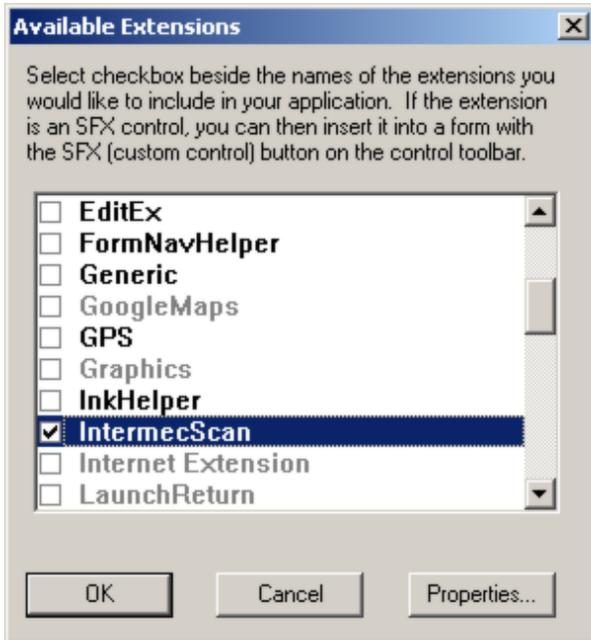
In order to control the Intermec barcode scanner, we'll utilize the IntermecScan plugin extension that is included with Satellite Forms. This extension is written to work with the Intermec scanner library that is preinstalled on most Intermec Windows Mobile/PocketPC scanners.

Okay, let's build an Intermec Scan Test sample application step by step:

Step 1. Start a new project in Satellite Forms MobileApp designer, and select the PocketPC platform target. A default form named Form 1 will be created, ready for you to add controls to. Click in the middle of the form, so that the Control Palette Toolbar becomes active. Let's add a form Title control, with the title text set to Intermec Scan Test. Next add a text control with the label Barcode: and below that add an edit control named edBarcode that stretches across the width of the form. Let's add a button control below that named btnScan, labeled SCAN, using the Tahoma Bold 10 font, and enlarge it to make it easy to tap on with a finger. Leave the button action set to No Action for now. Your form should look like this:

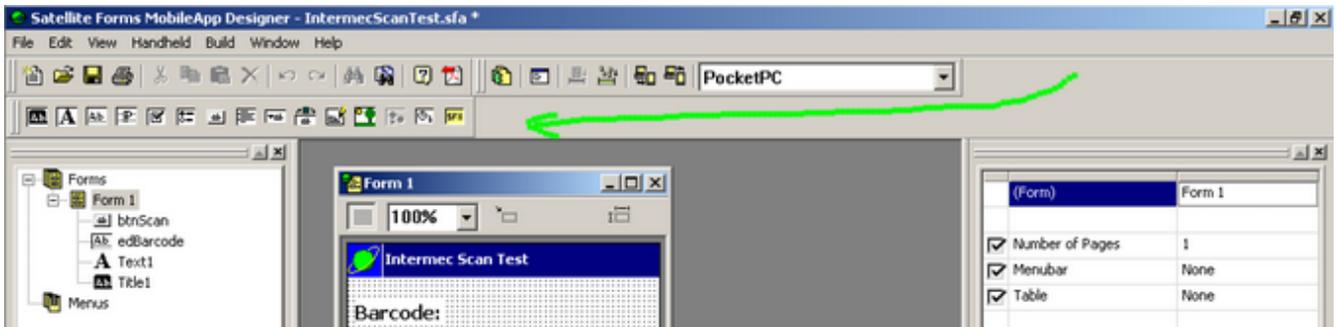


Step 2. The next step is to add the IntermecScan control extension to our project. Click on the Manage Extensions toolbar icon which looks like this:  A list of available extensions is displayed. Do not select the one at the top named Bar Code Reader. Scroll down the list, and select the IntermecScan extension, then click OK:



Step 3. Step 3. The IntermecScan extension is a custom SFX control, and not just a plugin script extension. Although it is a custom control, the control is not actually visible to the end user: it is only visible in form design view of MobileApp Designer. Because it is a custom control, an icon is added to the Control Palette toolbar so that you can click on the icon to add the control to your

forms. The SFX icon looks like this  and is added to the right end of the Control Palette toolbar. If you do not see it, there's a good chance that part of the Control Palette toolbar is simply being cut off because the MobileApp Designer window is not wide enough. That's easy to solve, we'll just grab the Control palette toolbar "handle" and drag it down to the next line in the MobileApp Designer desktop. Now we can see all the Control palette toolbar icons, like this:



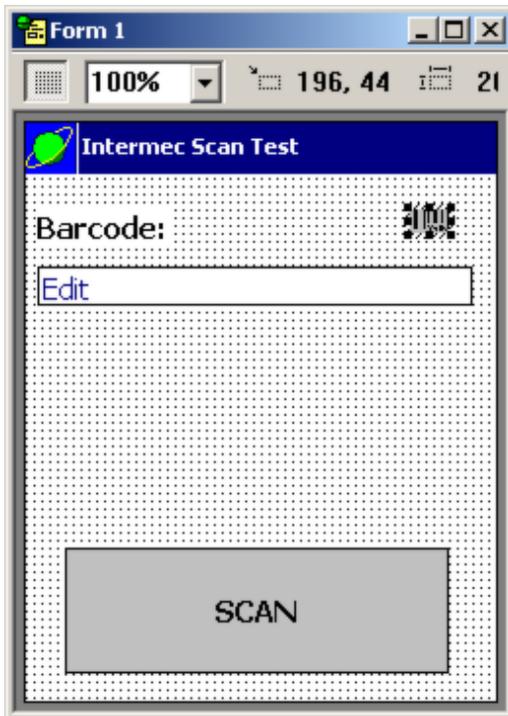
Okay good, we can see all the toolbar icons now.

So, click on the SFX custom control toolbar icon so that we can add it to our form. It's possible

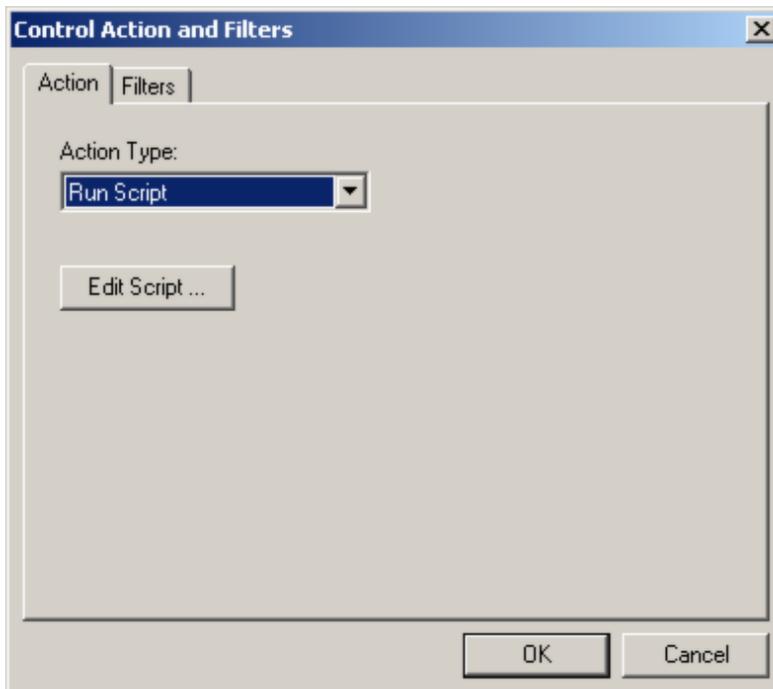
you might have more than one custom control on a form and they all share the same toolbar icon, so a selection box appears to let you choose the custom control you want, like this:



Click OK to select it, and the control then appears on our form as a little barcode symbol. Remember, this custom control is not visible to the end user. Let's move it out of the way so it does not overlap other controls, so your form should now look like this:



The default name that appears for the scanner control is IntermecScan1, and we'll leave it at that default. With the scanner control selected, click the Edit Action button in the control properties, and select Run Script from the Action Type droplist. This is the OnClick script for the scanner control, and this event will be fired when a barcode is scanned.



Step 4. Alright, now we're getting into the thick of things: here is where we decide what to do when a barcode is scanned. Click on Edit Script and the blank script editor window will open up. In this script, we are going to use a script method of the scanner control to obtain the barcode data that was scanned and place it into the edBarcode edit control on the form. Here's the script code to type in (feel free to copy & paste right from this QuickStart Guide article):

```
'this OnClick event is fired when a barcode is scanned
edBarcode = IntermecScan1.GetScanData
```

Now, let's make it so the onscreen SCAN button can trigger the barcode scan, in addition to the hardware scan buttons on the Intermec device. Select the SCAN button on the form, and click on Edit Action in the control properties. Select Run Script from the list of actions, then click on Edit Script. Type in this script to trigger the scan when the button is tapped:

```
'start scanner with timeout of 3000 ms
IntermecScan1.DoScan(3000)
```

Step 5. Okay, we need to do a couple more things to make barcode scanning work. What we need to do is to make sure the device we're running on is really an Intermec barcode scanner, and if so, make sure the scanner is enabled. If it is not an Intermec device, then we'll hide the btnScan control on the form. So, in the form AfterOpen event script that gets fired when the form is opened, confirm the scanner is enabled like this:

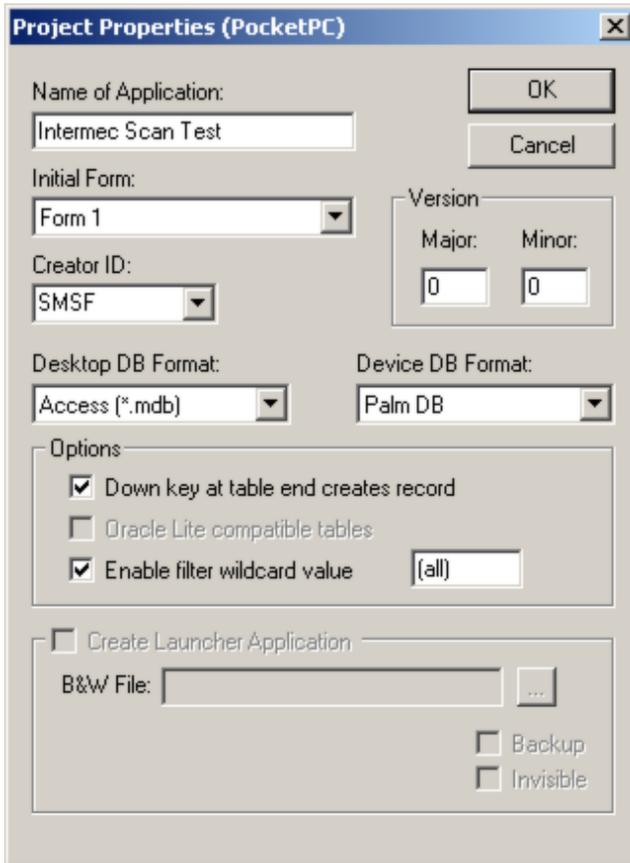
```
'if device is Intermec barcode scanner then
'confirm scanner is enabled

if IntermecScan1.IsIntermecScanner = True then
'confirm that scanning is enabled
if IntermecScan1.IsScannerEnabled = False then
IntermecScan1.ScannerEnable
endif
else
'warn user about no scanner
edBarcode = "Not Intermec scanner"
```

```
'hide the SCAN button  
btnScan.visible = false  
endif
```

That's about it for the barcode scanning stuff. Let's give the application a name, compile it, and test it out on the handheld.

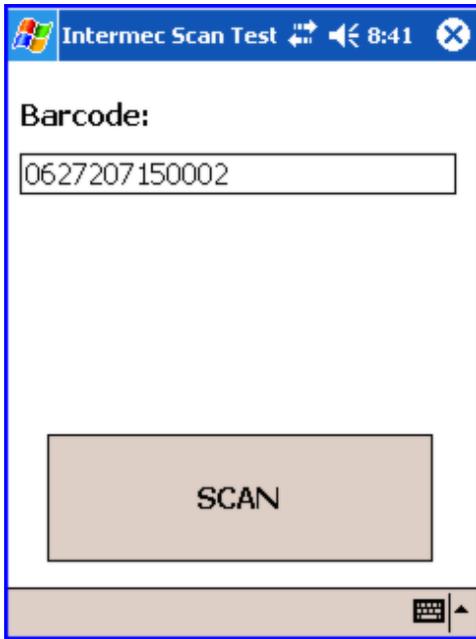
Step 6. Click on the Edit > Project Properties menu, and give our application the name Intermec Scan Test, like this:



Click OK, and then save your project. Name it Intermec Scan Test.sfa.

Now, compile the application by pressing the F7 hotkey (Rebuild All). If the compiler finds any typos, fix them, and then Rebuild All again. Next, connect the Intermec scanner to the cradle or sync cable, and use the Handheld > Download App & Tables menu option (or press the F5 hotkey) to download the application to your Intermec Windows Mobile/PocketPC scanner device. Launch the SatForms SDK app on the handheld, select Intermec Scan Test from the list, and you are ready to scan.

Point the scanner at a nearby barcode, press one of the scan buttons on the Intermec unit, and you the scanner should read the barcode and play a confirmation beep. Here's a sample screenshot from a Intermec CN2B screen:



Congratulations! You've just learned how to scan barcodes on Intermecc Windows Mobile/PocketPC scanners with Satellite Forms!

There are plenty of additional functions in the scanner control to give you even more detailed control over the scanning process, but you've got the basic scanning function covered just by applying what you learned in the QuickStart Guide above.

Additional Sample Projects:

 \Satellite Forms 7\Samples\Projects\IntermecScan\IntermecScan.sfa

Keywords: Intermecc, barcode, scanner, quickstart, PocketPC, Windows Mobile, CN2B

KB ID: 10081

Updated: 2007-12-03

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Barcode Scanning on Unitech Windows Mobile/PocketPC Scanners

QuickStart Guide to: Barcode Scanning on Unitech Windows Mobile/PocketPC Scanners

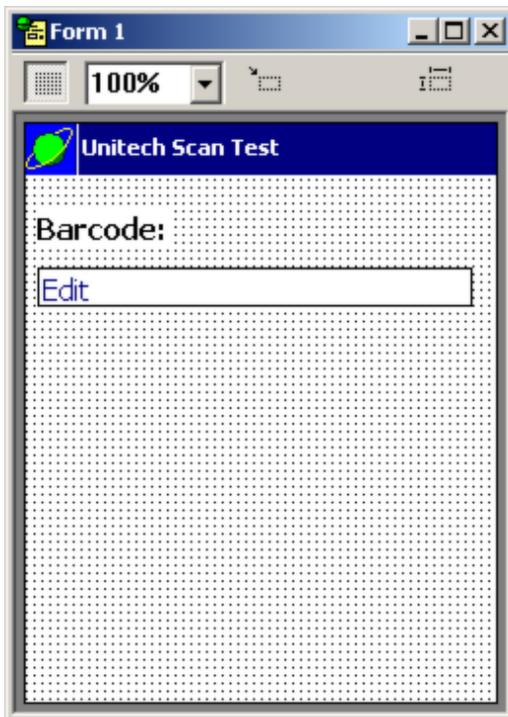
[Unitech](#) manufactures several models of Windows Mobile/PocketPC rugged handhelds with integrated barcode scanning capability. This QuickStart Guide shows you how to add Unitech Windows Mobile/PocketPC barcode scanning support to your Satellite Forms application, quickly and easily.

This QuickStart Guide contains plenty of screenshots to guide you through the process step by step, but don't let the length of this article scare you: the entire process of building the barcode scanning test application for the Unitech PocketPC scanners should only take about 15 minutes.

In order to control the Unitech barcode scanner, we'll utilize the UnitechScan plugin extension that is included with Satellite Forms. This extension is written to work with the Unitech scanner library that is preinstalled on most Unitech Windows Mobile/PocketPC scanners.

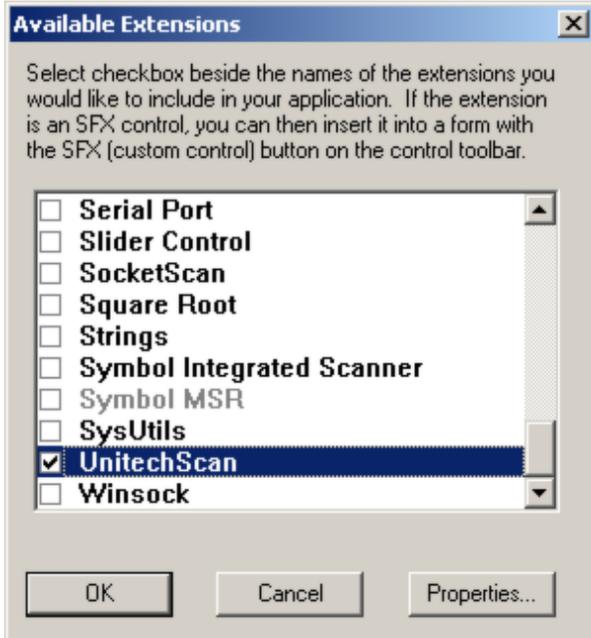
Okay, let's build an Unitech Scan Test sample application step by step:

Step 1. Start a new project in Satellite Forms MobileApp designer, and select the PocketPC platform target. A default form named Form 1 will be created, ready for you to add controls to. Click in the middle of the form, so that the Control Palette Toolbar becomes active. Let's add a form Title control, with the title text set to Unitech Scan Test. Next add a text control with the label Barcode: and below that add an edit control named edBarcode that stretches across the width of the form. Your form should look like this:



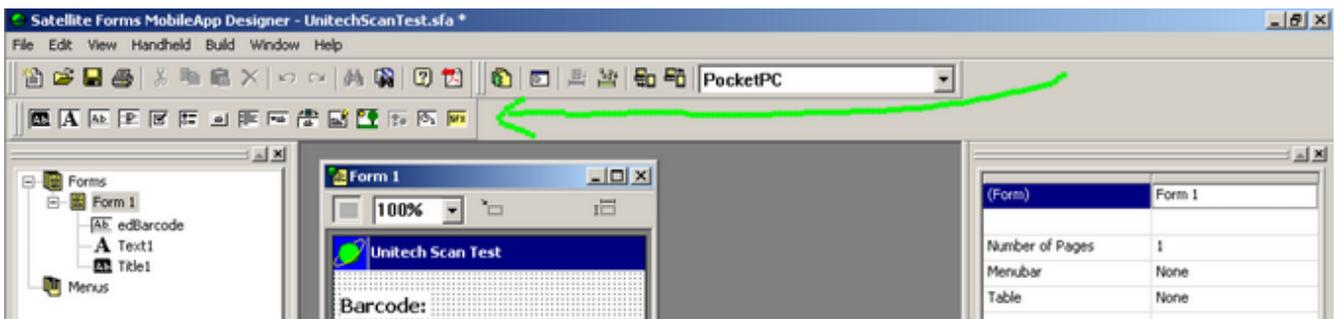
Step 2. The next step is to add the UnitechScan control extension to our project. Click on the Manage Extensions toolbar icon which looks like this:  A list of available extensions is

displayed. Do not select the one at the top named Bar Code Reader. Scroll down to the bottom of the list, and select the UnitechScan extension, then click OK:



Step 3. Step 3. The UnitechScan extension is a custom SFX control, and not just a plugin script extension. Although it is a custom control, the control is not actually visible to the end user: it is only visible in form design view of MobileApp Designer. Because it is a custom control, an icon is added to the Control Palette toolbar so that you can click on the icon to add the control to your

forms. The SFX icon looks like this  and is added to the right end of the Control Palette toolbar. If you do not see it, there's a good chance that part of the Control Palette toolbar is simply being cut off because the MobileApp Designer window is not wide enough. That's easy to solve, we'll just grab the Control palette toolbar "handle" and drag it down to the next line in the MobileApp Designer desktop. Now we can see all the Control palette toolbar icons, like this:

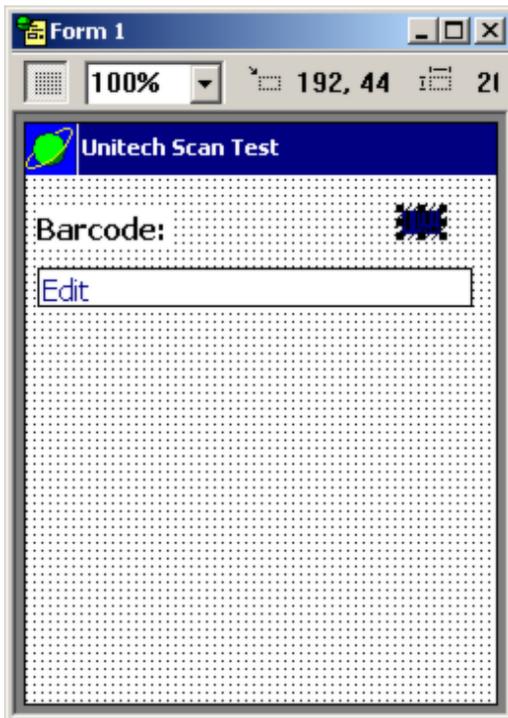


Okay good, we can see all the toolbar icons now.

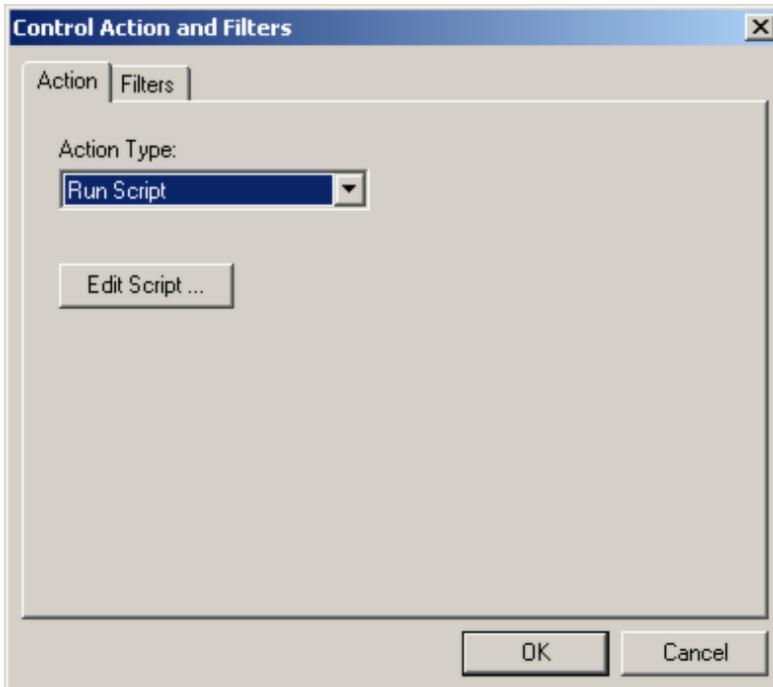
So, click on the SFX custom control toolbar icon so that we can add it to our form. It's possible you might have more than one custom control on a form and they all share the same toolbar icon, so a selection box appears to let you choose the custom control you want, like this:



Click OK to select it, and the control then appears on our form as a little barcode symbol. Remember, this custom control is not visible to the end user. Let's move it out of the way so it does not overlap other controls, so your form should now look like this:



The default name that appears for the scanner control is UnitechScan1, and we'll leave it at that default. With the scanner control selected, click the Edit Action button in the control properties, and select Run Script from the Action Type droplist. This is the OnClick script for the scanner control, and this event will be fired when a barcode is scanned.



Step 4. Alright, now we're getting into the thick of things: here is where we decide what to do when a barcode is scanned. Click on Edit Script and the blank script editor window will open up. In this script, we are going to use a script method of the scanner control to obtain the barcode data that was scanned and place it into the edBarcode edit control on the form. Here's the script code to type in (feel free to copy & paste right from this QuickStart Guide article):

```
'this OnClick event is fired when a barcode is scanned
edBarcode = UnitechScan1.GetScanData
```

Step 5. Okay, we need to do a couple more things to make barcode scanning work. What we need to do is to make sure the device we're running on is really an Unitech barcode scanner, and if so, make sure the scanner is enabled. We also need to disable the scanner when we leave the form. So, in the form AfterOpen event script that gets fired when the form is opened, confirm the scanner is enabled like this:

```
'if device is Unitech barcode scanner then
'enable scanner in AfterOpen event and
'disable in BeforeClose event

if UnitechScan1.IsUnitechScanner = True then
'enable scanning
UnitechScan1.ScannerEnable
else
'warn user about no scanner
edBarcode = "Not Unitech scanner"
endif
```

And in the form BeforeClose event, disable the barcode scanner like this:

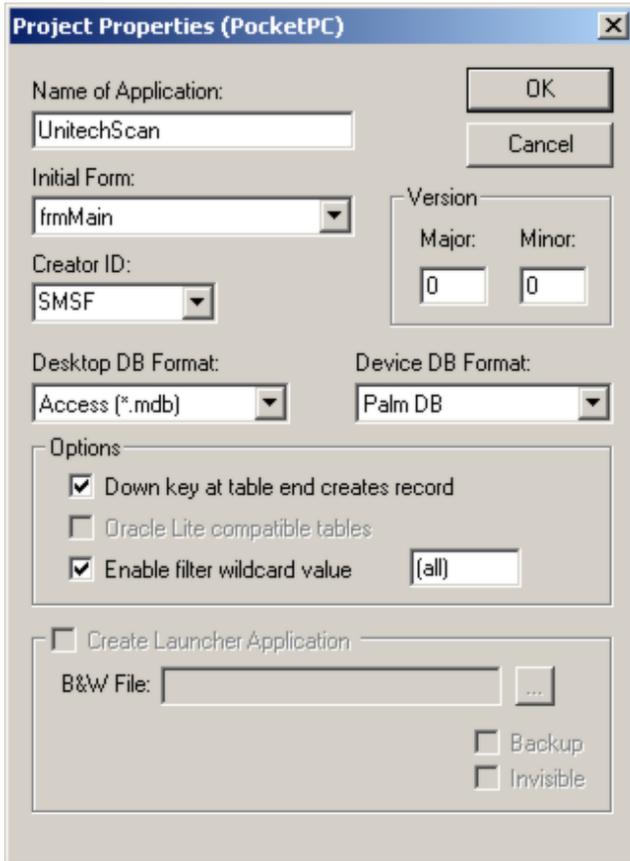
```
'if device is Unitech barcode scanner then
'disable in BeforeClose event

if UnitechScan1.IsUnitechScanner = True then
'disable scanning
```

```
UnitechScan1.ScannerDisable  
endif
```

That's about it for the barcode scanning stuff. Let's give the application a name, compile it, and test it out on the handheld.

Step 6. Click on the Edit > Project Properties menu, and give our application the name Unitech Scan Test, like this:



Click OK, and then save your project. Name it Unitech Scan Test.sfa.

Now, compile the application by pressing the F7 hotkey (Rebuild All). If the compiler finds any typos, fix them, and then Rebuild All again. Next, connect the Unitech scanner to the cradle or sync cable, and use the Handheld > Download App & Tables menu option (or press the F5 hotkey) to download the application to your Unitech Windows Mobile/PocketPC scanner device. Launch the SatForms SDK app on the handheld, select Unitech Scan Test from the list, and you are ready to scan.

Point the scanner at a nearby barcode, press one of the scan buttons on the Unitech unit, and you the scanner should read the barcode and play a confirmation beep. Here's a sample screenshot from a Unitech PA950 screen:



Congratulations! You've just learned how to scan barcodes on Unitech Windows Mobile/PocketPC scanners with Satellite Forms!

There are plenty of additional functions in the scanner control to give you even more detailed control over the scanning process, but you've got the basic scanning function covered just by applying what you learned in the QuickStart Guide above.

Additional Sample Projects:

\Satellite Forms 7\Samples\Projects\Unitech Scan\UnitechScan.sfa

Keywords: Unitech, barcode, scanner, quickstart, PocketPC, Windows Mobile, PA950, PA500, PA600

KB ID: 10082

Updated: 2007-12-03

[Satellite Forms KnowledgeBase Online](#)
[Satellite Forms Website Home](#)

-0-

Index

- 1 -

10 digit 42
1050 12
127 34

- 2 -

2002 50
2003 50

- 5 -

5V Power Out 174

- A -

Aceeca 203
ActiveSync 62, 54, 73, 13, 79, 120
activex 63, 47
administrator 47
alternate shape 22
An invisible 'hotspot' button is still clickable on PocketPC 31
AppDsn.tlb 47
authenticcode 47

- B -

Bar Code Reader 174
barcode 21, 25, 174, 189, 196, 41, 203, 209, 215
Barcode Scanning on Aceeca Meazura PalmOS Scanners 203
Barcode Scanning on Intermec Windows Mobile/PocketPC Scanners 209
Barcode Scanning on Janam XP PalmOS Scanners 189
Barcode Scanning on Symbol Windows Mobile/PocketPC Scanners 196
Barcode Scanning on Unitech Windows Mobile/PocketPC Scanners 215
binarysearch 152, 26, 175, 44, 43
Binarysearch function is not case sensitive on PocketPC PDB 43
Binarysearch function returns incorrect row number when no match on PocketPC PDB 44
binsearch 152, 43
bitmap 58, 63, 9, 103, 32, 28
Bitmap buttons behave differently on different PalmOS versions 9
bug 26
Bug in Binarysearch function with PocketPC PDB 26
bugs 19
build 100

bundle 133
button 9, 31

- C -

CAB 54, 83, 135
CABWiz 83, 135
cache 162
call 160
Cannot set droplist caption in code on PocketPC 36
capitals 158
caption 36
card 174
case 43
CDB 120, 100, 33
CERDKInst 149
char 156
checkbox 22
click 154, 31
close 30, 166
CN2B 209
color 58, 96, 103, 105
commit 162, 166
commitdata 166
commitdatabase 162
compile 28
compiler 52
control 17, 118, 96
conversion 120
convert 120
coordinate 118
create launcher icon 28
CreateFlag 120, 166
currentpage 17

- D -

data 56
database 62, 56, 49, 120, 46, 34, 166
DBF 62, 49, 120, 46
delete 39
Deleting a record causes 'Error 30: Table won't open or invalid' on PocketPC PDB 39
Delphi 79
deployment 135, 133, 149
desktop 46
DIA 107
dial 160
dialog 27
dimensions 37
display 63, 37
DLL 12
DLL error when installing application at Hotsync 12
dmsyncdatabase 162
double-tap 35
droplist 45, 36
dynamic 107

- E -

Edit 18
email 161
error 12, 16, 17, 41, 39
Error C016: Method 'Controls.SetPosition' takes 4 param(s) : 1 specified 23
Error compiling PocketPC target on PC with ActiveSync 4.0 13
Error locating third party PocketPC extensions when loading project 16
Error: Referenced control not on current page 17
expandable 107
extension 12, 16, 52

- F -

field 34, 33
filter 45, 42
Find 18, 152, 175
Find In Project does not always work 18
flag 120, 166
Form needs to be tapped by pen before accepting keyboard input on PocketPC 38
Form Scrollbars do not Appear Automatically in PocketPC 37
freeze 40
FTP 120
function 52

- G -

GetLastKey 158, 156, 38
GetPenStatus 154, 35
GetPosition 154
global 52, 154
graphic 58

- H -

HD 103, 105
height 118
high density 103, 105
hotspot 31
Hotsync 12, 49, 120, 47
How to Beam Files via IR or Bluetooth on PocketPC 169
How To Bundle the SatForms PocketPC runtime engine with your app 133
How To Change a control color using SFControlMagic 96
How To Change Control Fonts at Runtime 129
How To Change The CreatorID of a Palm Extension PRC To Match Your App 185
How To Commit Table Data to Storage Immediately 162
How To Create a shortcut to your PocketPC application 149
How To Create an Installer for your SatForms 6.x PocketPC Application 83
How To Create an Installer for your SatForms 7 PocketPC Application 135
How To Dial a Phone Number Using the LaunchURL Extension 160
How To Enable a User to Interrupt a Closed Loop 131
How To Enable Newer 2D Barcode Types on Janam Scanners 182
How To Force Input To ALL CAPS 158
How To Implement a Quick Find feature 152

How To Insert New Records Into a Sorted Table 175
How To Install SatForms PocketPC Runtime to multiple handhelds 73
How To Install the SatForms Runtime for Palm silently 77
How To Install the SatForms Runtime for PocketPC Programmatically 54
How To Limit Edit Control Input to Numeric Only 156
How To Make PocketPC PDB Apps Close Faster 166
How To make Satellite Forms 6.1 PocketPC applications launch faster 56
How To Move and Resize Controls at Runtime 118
How To Send an Email Message Using the LaunchURL Extension 161
How To support Expandable Screens in PalmOS applications 107
How To support multiple languages using build targets 94
How To Sync Satellite Forms Data to a Linux Server With jSyncManager 171
How To Use a Specific Connection using ConnectionMgr 180
How To use color bitmaps in your application 58
How To use different platform targets for PocketPC applications 50
How To use Global Functions & Subs to replace extension functions not available on the current target platforms
How To Use Google Maps for Windows Mobile from your Satellite Forms application 177
How To use High Density Bitmaps in PalmOS applications 103
How To use High Density Icons for your PalmOS applications 105
How To Use OnPenDown/OnPenUp Scripts to Detect Pen Taps on Controls 154
How To use PalmDB (PDB) tables in a PocketPC application 100
How To use SatSync to send data to the Palm device 49
How To use SatSyncPPC to send data to the PocketPC device 62
How To use SatSyncPPC to sync PocketPC data 127
How To use the Ink View OCX to display uploaded signatures on the PC 63
How To Use the MSR Attachment with Janam XP Scanners in Satellite Forms 174
How To use the PocketPC Emulator with Satellite Forms 97
How To use the SatForms ActiveSync control with Delphi 79
How To use the SFConvertPDB utility 120

- I -

icon 9, 105, 30, 28
IDVERIFI 203
image 58, 63, 103, 32
imager 41
incremental 152
index 56
INF 16
ink 63, 32
Ink control does not allow bitmap overlay on PocketPC 32
inkview 63
input 38
input area 107
install 54, 73, 77, 83, 135, 133, 149
installation 135
installer 83, 135, 149
Intermec 209
internationalization 94
Introduction 6
invalid 39
inventory 175
IsNumericInput 156
IsUppercaseInput 158

- J -

Janam 174, 196, 189, 40
Janam XP20/XP30 scanner powers off but needs to be reset to turn back on 40
Java 171
jSyncManager 171

- K -

keyboard 38
known issues 19
Known Issues for Satellite Forms 6.1 19

- L -

landscape 37
language 94
laser 25
launch 56, 30, 160
LaunchURL 161, 160
library 41
LifeDrive 107
Linux 171
listbox 34
localization 94
locate 175
location 118, 44
lockup 40
logo 58, 103
lookup 152

- M -

Magnetic Stripe Reader 174
mail 161
MC50 196, 41
MC70 196
MDB 62, 49, 120, 46
Meazura 203
memory 41
message 161, 160
messagebox 27
MEZ1000 203
Michael Schwarz 171
move 118
MoveRecord 175
msgbox 27
msi 77
MSR 174
multiple 73

- N -

NoAutoCommit 120, 166
NSIS 83
numeric 156, 42
NVFS 162

- O -

ocx 63, 54
onclick 9
OnKey 158, 156, 38
OnPenDown 154, 31, 35
OnPenUp 154, 31, 35
open source 171
overlay 32

- P -

PA500 215
PA600 215
PA950 215
page 17
palm 77, 100
PalmDataPro 96
PalmOS 49, 107, 189, 203
Patch 70002 42
PDB 120, 100, 26, 166, 44, 43, 42, 34
pen 154, 38, 35
pen tap 31
Pen tap on Text or Lookup control generates 2 OnPenDown events on PocketPC 35
PenTapInControl 154, 35
performance 56, 175
phone 160
picture 103
pixel 103
platform 52, 50, 22, 100, 36
PocketPC 21, 62, 56, 16, 54, 73, 13, 50, 79, 135, 22, 100, 26, 133, 149, 33, 27, 196, 46, 45, 44, 43, 42, 41, 39, 36, 34, 209, 215
PocketPC app is relaunched after it is closed 30
PocketPC CDB application cannot handle more than 127 fields per table 33
PocketPC device crashes when Symbol control used in app on non-Symbol PPC device 21
PocketPC PDB listbox problems with more than 127 table fields 34
PocketPC PDB problems filtering a record using a 10 digit numeric field 42
PocketPC project settings automatically change to MDB desktop DB format when project is loaded 46
PocketPC RemoveFilter function does not remove filter on droplist table 45
PocketPC target requires icon bitmap to compile even though it is not used 28
pointer 39
portrait 37
position 44
power 40
print 29
print all 29
Problem with alternate-shape checkboxes in PocketPC targets created from Palm target 22
Problems printing scripts in App Designer 29

project 46, 28
prompt 27
promptcustom 27
PromptCustom may cut off some text on PocketPC 27
properties 46, 28
protection 162

- Q -

quick 152
QuickSort 175
quickstart 189, 196, 203, 209, 215
QVGA 37

- R -

radio button 22
RDKInst 47
Read-Only 120, 166
record 33, 39
red circle 105
relaunch 30
removefilter 45
Replace 18
reset 162
resize 118
restart 30
ROM 40
runtime 54, 73, 77, 133

- S -

sandwich 9
SatSync 62, 49
SatSyncPPC 62
save 162
scanner 21, 25, 189, 196, 41, 40, 203, 209, 215
screen 37
scroll 37
scrollbar 37
SDIO 25
SDSC 3M 25
Search 18, 152, 175
sensitive 43
serial 174
settings 46
setup 77, 83, 135, 133, 149
SF-00328 26
SFControlMagic 96
SFConvertPDB 120, 166
SFrmUt 13
SFTempTable.cdb 13
shortcut 149
signature 63, 32
silent 77

size 118, 37
Socket 25
SocketScan 25
SocketScan PalmOS control does not work with laser SDIO scanner 25
soft Graffiti 107
sort 175, 44, 43
speed 56, 175, 166
square 37
StartApp 54
storage 162
sub 52
Symbol 21, 189, 196, 41
Symbol MC50 scanner shows Error Enabling Scanner Library after scanning for a while 41
sync 120
synchronization 120
SYS_POSIDX 56
SYS_POSITION 56

- T -

T3 107
T5 107
table 62, 49, 100, 33, 166
tap 154, 35
target 52, 50, 22, 100
targets 94
TCPIP 120
Technodane 171
telephone 160
template 32
transfer 120
TX 107
typeahead 152

- U -

uninstall 135
Unitech 215
Unix 171
upper case 158
URL 161, 160

- V -

validate 156
virtual Graffiti 107
visible 31

- W -

wceload 54
width 118
WinCE.NET 50
Windows Mobile 135, 26, 133, 149, 196, 209, 215

Windows Mobile 5 13
Windows Vista 47
Windows Vista Compatibility Issues 47
WinMobile 5 13, 50
WM5 26
write 162

- X -

XM60 196
XP 174, 189
XP20 189, 40
XP30 189, 40

