July, 1998

Advisor Answers

Visual FoxPro 3.0, 5.0 and 6.0

Visual FoxPro 5.0 and 6.0

Q: I have an application that uses a SDI main form (ShowWindow = 2, Desktop = .T., MDIForm = .F.) and hides the FoxPro screen. Nothing happens when I try to do a print preview in my compiled applications. When I do a print preview in the development environment, the preview window is contained within the FoxPro screen even if I use the IN WINDOW clause of the REPORT command in a window that has Desktop = .T.. Is there a way to do print preview with a SDI-type form?

–Henry Hicks (via Advisor.Com)

A: This is another thing that's much easier in VFP 6 than in previous versions. In VFP 5, I can't find any way of getting a report preview to appear in the desktop window or a child window of it, so you have to use a clever work-around.

Let's start with the easy version. In VFP 6, if you're operating in a desktop window, report previews automatically use that window as home base. Just issue REPORT FORM YourReport PREVIEW and you're in business. If, for some reason (and I can't think of any), you want the report to preview in the main VFP window, there's a new IN SCREEN clause that forces things to work that way.

In VFP 6, REPORT FORM PREVIEW also supports both the WINDOW and IN WINDOW clauses, so you can play all the same tricks you do with other windows. The WINDOW clause indicates that the preview window should draw its characteristics (like size and position) from the named window. IN WINDOW says that the preview window should be placed in the named window. On the whole, the improvements in report previews by themselves seem like a good reason to upgrade from VFP 5 to VFP 6.

Back to VFP 5. Unfortunately, although REPORT FORM PREVIEW accepts the WINDOW clause here, it doesn't use the IN WINDOW clause. IN WINDOW is the one we'd need to make the preview appear in a desktop window.

Also, in VFP 5, as you've discovered, report previews automatically use the main VFP window. In your compiled application, you've no doubt hidden the main VFP window (_SCREEN), so you don't see the preview when it appears. (See Ask Advisor in the September '97 issue for a discussion of hiding _SCREEN.)

So what are your options? Essentially, you need to unhide the main VFP window long enough to show the preview, then hide it again. If you've hidden it by positioning it in negative space, move it back temporarily, do your preview, then move it out again.

To make the change as invisible as possible, try code this that makes the main VFP window mimic the form you're in:

```
LOCAL nTop, nLeft, nHeight, nWidth, cCaption
```

```
nTop = _SCREEN.Top
nLeft = _SCREEN.Left
nHeight = _SCREEN.Height
nWidth = _SCREEN.Width
cCaption = _SCREEN.Caption

_SCREEN.Top = THISFORM.Top
_SCREEN.Left = THISFORM.Left
_SCREEN.Height = THISFORM.Height – SYSMETRIC(9) ;
                 – SYSMETRIC(20)
_SCREEN.Width = THISFORM.Width
_SCREEN.Caption = THISFORM.Caption

REPORT FORM Test PREVIEW

_SCREEN.Top = nTop
_SCREEN.Left = nLeft
_SCREEN.Height = nHeight
_SCREEN.Width = nWidth
_SCREEN.Caption = cCaption
```

You also need to make sure that the window doesn't bring any extra toolbars with it. In a runtime environment, where you're keeping _SCREEN hidden, that shouldn't be a problem since, by default, _SCREEN has no toolbars in that setting.

The subtraction in the assignment to _SCREEN.Height is because it appears that assigning a value to _SCREEN.Height simply doesn't account for the title bar and menu bar. With the two subtractions, the main VFP window ends up almost exactly the same height as the top-level form. I still see a small discrepancy, but only the most discerning users are likely to notice.

Do be aware that users can actually move _SCREEN in this situation and expose the form beneath. (In fact, that's how I discovered that the two weren't exactly the same size.) However, the user can't actually do anything with your form until she closes the report preview, because previews are modal.

–Tamar