August, 2003

Advisor Answers

Docking toolbars at runtime

VFP 8/7/6

Q: I'm using the new DOCK WINDOW command in VFP 8 for my application to dock my toolbar to the main VFP window, like this:

```
DOCK WINDOW dpatoolbar POSITION 0
```

It works great in the compiled APP. However, when I try to use it in an EXE, I get an error message "Feature Not Available." Can anyone tell me why?

–Walter Berg (via Advisor.COM)

A: Although it's not documented, it appears that the DOCK WINDOW command is on the list of restricted commands that are available only in the development environment and not at runtime. That's not surprising because the DOCK WINDOW command is documented as working only with IDE windows and toolbars.

I was surprised to find that you can use DOCK WINDOW with a custom toolbar, but in fact, code like this does work:

```
oToolbar = CREATEOBJECT("MyToolbarClass")
oToolbar.Show()
DOCK WINDOW (oToolbar.Name) POSITION 0
```

This code doesn't work in the runtime environment, however. Fortunately, there's another way to dock toolbars that works at development time and runtime, and even better, works in all versions of VFP.

Toolbars have a Dock method that docks and undocks them. The method has one required parameter, the docking location. The choices are shown in Table 1.

Table 1. Where to dock—The first parameter to the Dock method indicates whether to dock or undock and, if you're docking, where to dock.

Value	Meaning
-1	Undock the toolbar
0	Dock the toolbar at the top
1	Dock the toolbar at the left side
2	Dock the toolbar at the right side
3	Dock the toolbar at the bottom

So, to instantiate and dock a toolbar at the top as in the example above, you can use code like:

```
oToolbar = CREATEOBJECT("MyToolbarClass")
oToolbar.Dock(0)
oToolbar.Show()
```

By calling the Dock method before the Show method, the toolbar is docked on its initial appearance.

Be aware that you can't always control the exact location where a toolbar will dock. Christof wrote about this problem in the October, 2001 issue.

You can also check on the docking status of a toolbar, using its DockPosition property. This read-only property holds the values shown in Table 1 to indicate a toolbar's current status.

The object model for docking is pretty complete. In addition to the Dock method and DockPosition properties, there are events that fire as a toolbar is docking and undocking. Three events are involved: BeforeDock, AfterDock and Undock. BeforeDock receives as a parameter the docking location (the same values as in Table 1).

BeforeDock fires as a toolbar is docked and AfterDock fires once docking is complete. When you undock a toolbar, Undock fires first, followed by BeforeDock. If you move a toolbar from one docked location to another, you get this sequence: Undock, BeforeDock (with -1), BeforeDock (with the new location), AfterDock.

There are a couple of items worth noting. First, the firing sequence has changed a number of times in different versions of VFP. The paragraph above describes the firing sequence in VFP 7 and 8. If you're using an earlier version, experiment to be sure you know what's really happening.

Second, there's nothing you can do in BeforeDock to prevent the toolbar from docking. You might think that NODEFAULT or RETURN .F. would prevent docking, but they don't and calling Dock(-1) in BeforeDock raises an error.

Finally, the events fire whether the toolbar is docked interactively or programmatically.

One thing VFP doesn't offer in any version is the ability to dock user-defined forms. However, this has been demonstrated as a possible feature for the next version of VFP, code-named Europa.

–Tamar