Visual Studio Magazine Online
Classic VB Corner

VB Statement Against Vista

ONLINE ONLY

VB survived Vista nearly intact, with the glaring exception of one statement. Learn what your options with SendKeys are.

November 13, 2007 · by Karl E. Peterson

I gotta hand it to Microsoft. For all the turmoil over how Vista broke this and Vista broke that, and oh yeah, that too!, Vista didn't really break Classic VB all that badly.

But there is one glaring exception that's gotten some spotty press, lots of snide commentary, and not a few unscrupulous vendors looking to capitalize on others' misfortune.

The one element of the VB language that almost got totally hosed by Vista is the SendKeys statement. Yes, SendKeys has a well-deserved reputation as one of the most maligned statements in the VB language. The best recourse is to simply avoid it in nearly every circumstance. And yet, there are times when it's just too darn convenient to ignore. Besides, the promise of support is there, right?

To see this problem for yourself, just start a new project and plop a textbox and commandbutton on the default form. Add this code:

```
Private Sub Command1_Click()
   Text1.SetFocus
   SendKeys "{Home}+{End}Ka-boom!"
End Sub
```

Run it then press the button. Ka-boom!, right? No? Heh, then you'll never experience the true joy that (perhaps many) of your users do. (But hey, if you were looking for a user experience, you wouldn't be a developer, right?) Anyway, if UAC is enabled <scold>, attempting to call SendKeys within the VB IDE will cause an error 70 -- permission denied.

Now we get to the good news and the bad news. The good news is Microsoft was jumped on during the beta over this issue, and it actually patched the runtime (MSVBVM60.DLL) such that it now uses a different method to accomplish much the same thing. So "all you have to do" is compile, and you're back in shape. SendKeys doesn't generate the error 70 in a VB6-authored EXE, only in the VB6 IDE.

Ah, but you noticed that I said it behaves "much the same" as before, you say? Sharp eyes. If you've used SendKeys much, you'd recognize the "{Home}+{End}" sequence as one that highlights the entire content of a textbox. For some reason, the Shift (+) in there isn't recognized in Vista anymore. Now, you just get "{Home}{End}", thus losing much effect. (Again, if UAC is disabled, the Shift-End works just fine.)

You say you've also noticed that I'm only talking about VB6? Very sharp eyes, Sparky! That's right. VB5 is utterly unsupported anymore, so no fixes have come its way. Same goes for Office 2000 VBA. So you're very much out of luck if you've shipped solutions based on either of those platforms that make use of SendKeys.

What about Office 2003? Yep, it's still fully within its standard support lifecycle, but nope, Microsoft hasn't addressed this bug there at all. A spokesman told me, "Office 2007 is the first product release that was available at the time of or after Vista's release, and it makes sense that it has the best experience on Vista. Office 2003 is still supported. The fix may be considered for a future Office 2003 SP if and when that is available."

So shipped Office 2003-based solutions are SOL, too. There's a glimmer of good news in Office 2007 support -- not that anyone much is using it -- but SendKeys doesn't generate any outright errors there.

What workarounds do you have? Well, they're not pretty. If you're using a flavor of ClassicVB other than VB6, every one of them requires a recompile/redeploy. VB5 applications that can be recompiled in VB6 offer the easiest solution. Another option would be to compile a DLL in VB6 that exposes a class that exposes a method wrapper around SendKeys. That's probably the next-easiest solution and should work in any situation, including VBA. (There are a number of commercial offerings out there, which are essentially this.)

Taking full control away from VB and/or other outside libraries is possible by turning to the keybd_event or SendInput APIs. I found a nice class-based solution on Steve McMahon's vbAccelerator site, which you might want to check out. It suffers the same issues with highlighting, but all in all is a really comprehensive beginning to a full-blown solution.

One other really off-the-wall option exists, but may be appropriate if you're already using the Windows Scripting Host (WSH) library. WSH has also been fixed up for Vista, although it's far from a desirable dependency for this feature alone. Here's the general idea with that library:

```
Public Sub SendKeys(ByVal Keys As String, Optional Wait As Boolean = False)
   Dim wsh As Object
   ' Only mess with shell object under constraint.
   If (IsVista() = True) And (Compiled() = False) Then
      ' Create shell object, which works.
      Set wsh = CreateObject("WScript.Shell")
      wsh.SendKeys Keys, Wait
      Set wsh = Nothing
   Else
      ' Use standard function when compiled or
      ' not running on Vista!
      VBA.SendKeys Keys, Wait
   End If
End Sub
```

All in all, as I said, not a pretty picture. I think Microsoft could've done better than this for "supported" platforms, don't you?

Oh, almost forgot, the same issue exists in both the 1.0 and 2.0 versions of the .NET Framework. It's been partially addressed in the 3.0 framework and is supposed to finally be finished in the 3.5 version. There's support, huh?

One final option: Turn off UAC entirely. Of course, you'll have to advise your users to do so, as well, or handle the "permission denied" error appropriately.

About the Author

Karl E. Peterson wrote Q&A, Programming Techniques, and various other columns for VBPJ and VSM from 1995 onward, until Classic VB columns were dropped entirely in favor of other languages. Similarly, Karl was a Microsoft BASIC MVP from 1994 through 2005, until such community contributions were no longer deemed valuable. He is the author of VisualStudioMagazine.com's new Classic VB Corner column. You can contact him through his Web site if you'd like to suggest future topics for this column.