Launch Your App

by Chris Barlow

Make sure you use the latest version of your app when making changes to the code.

You complete your first Visual Basic system and install it on the user's computer. It consists of a database and three different applications. You put shortcuts to all three applications on the user's desktop. The user double-clicks on the shortcuts and sees the database open and each application start up. The user is satisfied. You're all done, right?

If you've been through a system startup, you know the answer to that question. You're not even close to "all done," and in some ways, this is the beginning. I guarantee the user will call and say one of the applications is not working right, or that he or she wants a "minor" change in one of the screens. When taking a fresh look at the applications, you'll find a little code you want to tweak or a feature you want to improve.

Welcome to the world of system maintenance. Fortunately, Visual Basic makes it easy to implement the kind of changes users invariably want as they begin to live with a new application. You can change the menus, redraw the screens, and even add new fields to the database almost faster than users can ask you to. The nice thing is that most of these changes require only a simple compile of a new EXE—not an entire new setup kit.

But installing the new version of your application so the user can launch it is not as easy as it sounds. Often, you don't visit the user's site to install the new version; instead, you connect to his or her computer through a modem, LAN, or the Internet. Can you connect directly to the user's disk to overwrite your old EXE, or can you see a shared server on the user's network? You can't have your applications reside on the server and point each user's desktop shortcuts to the server—suppose the user is running the application when you want to replace it? You'll get the error message saying the file is open and cannot be overwritten.

At SunOpTech, we solved this problem by creating an application launcher. Whenever a user wants to run one of the applications in the system, his or her request is funneled through the launcher first. The SunOpTech Launcher is not as rigid as a typical menu application. In most cases, the user doesn't realize it's being executed. Its job is to make sure the latest version of the application is running.

All the desktop shortcuts point to the launcher instead of the application. When the user double-clicks on the shortcut to launch an application, Launcher fires up. Launcher looks at the version information from the application's EXE file in the program folder. Then Launcher looks for the same EXE file in a predefined install folder on that computer (or server)

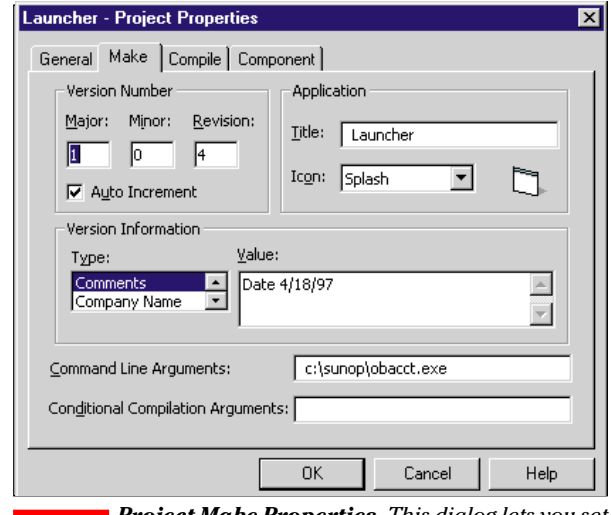

FIGURE 1 ***Project Make Properties.*** *This dialog lets you set the Command Line Arguments you need to test the launcher application in the Visual Basic design environment. Notice Launcher's version information and the Auto Increment check box to increment the App.Revision property every time the project is compiled.*

Chris Barlow is president and CEO of SunOpTech, a developer of manufacturing decision-support applications, including the ObjectBank, Object Order, and ObjectJob Systems. He is a frequent speaker at VBITS, Tech•Ed, and DevDays, and is featured in two Microsoft videos. Reach Chris at ChrisB@SunOpTech.com or through SunOp Tech's Web server at www.SunOpTech.com.

and checks the version information of that file. If the EXE file in the install folder is newer, Launcher copies the newer version into the program folder. Finally, Launcher gives the command to launch the application, and Launcher terminates.

The launcher makes program upgrades easy. When you compile a new version of the application, copy it to the install folder. The next time the user runs the application, Launcher copies your new version to the program folder and launches it.

Because requests to run your applications are funneled through the launcher, you can centralize functionality in Launcher. For example, you can use Launcher to log the user onto your system by requesting a user ID and password and verifying this information against a database. You can check a central message database for messages of interest to this user. You can "lock up" your applications so they won't start unless they receive a certain code on the command line from Launcher. That way, instead of letting the user run the application directly, you'd force the user to go through Launcher.

CREATE A LAUNCHER

To use the launcher, you must control the version of your application. Visual Basic makes it easy to identify your program by adding a version resource to the EXE file. You can modify the information about your application using the Options button on the Make Project window. I suggest setting the option to automatically increment the revision each time you compile the application. You can override the Minor and Major properties as you make substantial changes to your application.

In last month's column ["Identify an App's Version" *VBPJ* August 1997], I wrote an ActiveX DLL that reads this version information, along with the file's date and time. Launcher uses this FileVer DLL to read the version resource information from the file in both the program and the install folders to determine which is more recent. You can download this DLL from the Premier Level of The Development Exchange (see the Code Online box at the end of this column for details).

Start a new Standard EXE project in Visual Basic and right-click on the Project window to add a module. Click on the Properties menu item on the Project menu to name the project "Launcher." Set the Startup Object to Sub Main. Then click on the References menu item on the Project menu to add a reference to the FileVer DLL from last month's column.

Visual Basic's Standard EXE project has a single form you can use as a splash screen to let the user know what's happening while you copy the upgraded EXE file. Rename the form "Splash.frm." I suggest designing a simple form with Label controls for Launcher's version and copyright, as well as a larger Label control to display the action that Launcher is performing.

The Main subroutine executes when Launcher starts. First, show the Splash form to give the user some visible feedback. Get the setting for the install folder from the registry using the GetSetting statement, and store the setting in a public variable called InstallFolder. If this variable is null, as it will be the first time Launcher runs on a computer, use the InputBox function to request the install folder. If the user doesn't enter a folder or presses the Cancel button, Launcher will terminate; otherwise, the install folder setting is saved to the registry:

VB MAKES IT EASY TO IMPLEMENT THE KIND OF CHANGES USERS INVARIABLY WANT AS THEY BEGIN TO LIVE WITH A NEW APP.

```
Public Sub Main()
Dim txt$
Splash.Show
Splash.Refresh
InstallFolder = _
   GetSetting("Launcher", _
   "Folders", "InstallFolder", "")
If Len(InstallFolder) = 0 Then
   InstallFolder = InputBox_
      ("Please enter path to " & _
      "Install Folder", _
      "No Install Folder", App.Path)
   If Len(InstallFolder) = 0 Then _
      End
   SaveSetting "Launcher", _
      "Folders", "InstallFolder", _
      InstallFolder
End If
```

The application to be launched is passed on the command line with the full path to the application. You can test this in debug mode by setting the Command Line Arguments on the Make tab of the Project properties dialog (see Figure 1). Visual Basic makes command-line arguments available through the Command property of the App object. If this property is null, the user hasn't specified a program to launch, and an appropriate error message is displayed:

```
If Len(Command) = 0 Then
   txt = "This program will " & _
      "launch the program " & _
      "specified on the " & _
      "command line." & vbCrLf
   txt = txt & "It will " & _
      "compare this program's " & _
      "date and version " & _
      "against the same " & _
      "program in " & vbCrLf
   txt = txt & InstallFolder & _
      vbCrLf
   txt = txt & "and give you " & _
      "the option to replace " & _
      "your program with this " & _
      "version." & vbCrLf
   txt = txt & "Please enter " & _
      "path and EXE name on " & _
      "command line and re-run " & _
      "this program."
   MsgBox txt
```

If there is any text on the command line, then the Command property is parsed at the first space, in case additional command-line arguments need to be passed to the launched program (download the ParseString procedure, included in the complete listing for the Launcher module, from the free, Registered Level of The Development Exchange. See the Code Online box at the end of this column for details). The file path is passed to a ProgCheck procedure to check the version information. If this procedure returns True, the application launches using the Shell statement. Finally, the Splash form is unloaded, and Launcher terminates:

```
Else
   If ProgCheck(ParseString_
      (Command, " ", 1)) Then
      Shell Command, 1
   End If
   Unload Splash
End If
End
End Sub
```

CHECKING THE VERSION

The ProgCheck procedure needs to check whether the file exists in both the program and install folders. The ProgCheck procedure also checks the versions and copies the EXE from the install folder if it is a newer version. First, check for the file in the program folder and get its version

with the GetProgVersion procedure:

```
Private Function ProgCheck(File$) _
   As Boolean
Dim verLocalF$, res%
Dim verServerF$
Dim prgServerF$, msg$
res = FileExist(File)
If res = True Then
   verLocalF = GetProgVersion(File)
Else
   msg = File & _
      " is not in Program folder." _
      & vbCrLf
End If
```

THE LAUNCHER'S JOB IS TO MAKE SURE THE LATEST VERSION OF THE APPLICATION IS RUNNING.

Check for the file in the install folder, get its version, and compare the two versions. If the program folder version is lower, put a message in the Comment label on the splash form and use the FileCopy statement to copy the file:

```
prgServerF = InstallFolder & "\" & _
   StripPathName(File)
res = FileExist(prgServerF)
If res = True Then
   verServerF = _
      GetProgVersion(prgServerF)
   On Error GoTo ErrorHdler
   If verLocalF < verServerF Then
      Splash.lbComment = _
         "Copying from Server"
      Splash.Refresh
      FileCopy prgServerF, File
      msg = ""
   End If
   ProgCheck = True
```

The procedure ends with appropriate error handling.

The GetProgVersion procedure is simple because it uses the FileVer DLL created in last month's column. Dimension an instance of the cFileVer class and set the sFileName property. Then call the GetFileVersionData method and return the sFileVersion property:

```
Private Function _
   GetProgVersion(File$) As String
Dim cF As New cFileVer

On Error GoTo Errhdlr
cF.sFileName = File
cF.GetFileVersionData
GetProgVersion = cF.sFileVersion

Exit Function

Errhdlr:
   MsgBox "Error= " & Error$ & _
      " Err=" & Err.Number & _
      Chr(13) & "File name=" & _
      "(" & File$ & _
      ")"

End Function
```

You can download the source code files for Launcher from the Premier Level of The Development Exchange (see the Code Online box for details). I'm curious to see how you enhance Launcher. E-mail me your improvements, and I'll share them with other readers.

Code Online

You can find all the code published in this issue of **VBPJ** *on The Development Exchange (DevX) at http://www.windx.com. All the listings and associated files essential to the articles are available for free to Registered members of DevX, in one ZIP file. This ZIP file is also posted in the Magazine Library of the* **VBPJ** *Forum on CompuServe. DevX Premier Club members ($20 for six months) can get each article's listings in a separate file, as well as additional code and utilities for selected articles, plus archives of all code ever published in* **VBPJ** *and* **Microsoft Interactive Developer** *magazines.*

Launch Your App
Locator+ Codes

Listings ZIP file and GS0997.ZIP, which includes the complete listing for the Launcher module discussed in this column (free Registered Level): VBPJ0997

Listings for this article plus the source code for the launcher application and the FileVer DLL discussed in last month's column (subscriber Premier Level): GS0997P