Avoiding Enumeration Name Conflicts

In the preceding code example, both the Enum and its members were prefixed with four lowercase characters chosen to identify the component they belong to, and to reduce the chance that users of the component will encounter name conflicts. This is one of the general naming rules discussed in "What's in a Name?" earlier in this chapter.

For More Information Enumerations are discussed in detail in Chapter 8, "More About Programming," and Chapter 9, "Programming with Objects," in the *Microsoft Visual Basic 6.0 Programmer's Guide.*

Providing Non-Numeric and Non-Integer Constants

The members of an Enum can have any value that fits in a Long. That is, they can assume any integer value from –2,147,483,648 to 2,147,483,647. When you declare a variable using the name of an Enum as the data type, you're effectively declaring the variable As Long.

Occasionally you may need to provide a string constant, or a constant that isn't an integer value. Visual Basic doesn't provide a mechanism for adding such values to your type library as public constants, but you can get a similar effect using a global object with read-only properties.

If your component doesn't contain a global object, such as Application, add a public class module named GlobalConstants to your project. Set the Instancing property to GlobalMultiUse.

For each constant you want to provide, add to the GlobalConstants class module a Property Get procedure that returns the desired value. For example, the following code provides Avogadro's Number as a constant, and mimics the vbCrLf constant in Visual Basic.

```
Public Property Get Avogadro() As Double
   Avogadro = 6.02E+23
End Property
Public Property Get vbCrLf() As String
   vbCrLf = Chr$(13) & Chr$(10)
End Property
```

Because the Instancing property is GlobalMultiUse, a user of the component doesn't have to explicitly create an instance of the GlobalConstants class in order to use the constants. The constants can be used as if they were part of Visual Basic:

```
strNewText = "Line1" & vbCrLf & "Line2"
```

> **Note** A user of Visual Basic, Microsoft Excel, or any other application that hosts Visual Basic for Applications would never see this version of the vbCrLf constant, because the VBA type library is always higher in the References dialog than the type library of any component.