CHAPTER 20

Putting It All Together: Your First WMI/ADSI Script

IN THIS CHAPTER

It's time to leverage what you've learned about ADSI and WMI scripting. In this chapter, I'll walk you through the entire design and creation process for a new script. In addition to demonstrating a useful new purpose for WMI and ADSI, this chapter will help strengthen your script design skills.

By now, you should have a good idea of what WMI and ADSI can do for you. In this chapter, I'll walk you through the complete design process for an entirely new script. This time, I'll use both WMI and ADSI in the same script. The script's job will be to check in on every computer in an Active Directory or NT domain and query some information about its operating systems. I want the script to output this information to a text file on a file server. The information I want to collect includes operating system version, service pack level, number of processors in the machine, maximum physical memory in the machine, and so forth. This is a useful way to quickly inventory a network and see what machines might need to be upgraded before deploying a new application, or to see what machines don't have the latest service pack applied.

Designing the Script

My script is a reasonably complex undertaking, so it helps to break it down into manageable tasks. I need the script to do three things:

1. Query a list of computers from the domain.
2. Query information from each computer.
3. Write information out to a text file.

The last bit is probably the easiest. I can use the FileSystemObject to open a text file, write information to it, and then close the text file. Something like the following would work.

```
Dim oFSO, oFile
Set oFSO = CreateObject("Scripting.FileSystemObject")
Set oFile = oFSO.CreateTextFile("output.txt")
oFile.Write "Information"
oFile.Close
```

For more information on using the FileSystemObject, refer to Chapter 12.

Querying a list of computers from the domain shouldn't be too hard, either. If I want the script to work with both NT and Active Directory domains, I need to use the WinNT ADSI provider, because only that provider works with both domains. I can query all of the objects in the domain, and then use an `If…Then` construct to work with only the computer objects. Code such as the following should do the trick.

```
Dim oDomain
Set oDomain = GetObject("WinNT://" & sDomain)
Dim oObject, sComputerName, sDetails
For Each oObject In oDomain

 'is this object a computer?
 If oObject.Class = "Computer" Then

  'yes – do something with it

 End If
Next
```

For more information on querying domains by using ADSI, see Chapter 14, and see "Querying Domain Information" in Chapter 15.

Pulling the operating system (OS) information is tougher. WMI seems like the way to go, but WMI has about three gazillion classes. Which one do I need? Fortunately, I have a way to cheat. My primary script editor is Sapien Technology's PrimalScript 3.0, and it includes a WMI Script Wizard.

NOTE A trial version of PrimalScript 3.0 is included on the CD that accompanies this book.

Running the wizard displays the dialog box shown in Figure 20.1. The left side of the dialog box shows a list of every WMI class that my computer knows about. Scrolling through the list, I find that there's a class named `Win32_OperatingSystem`. That seems like a good place to start.

Clicking the `Win32_OperatingSystem` class changes the dialog box to look like the one shown in Figure 20.2. Here, the wizard has filled in a sample script capable of querying information from the selected class. I see things like service pack level and operating system version, so this is probably the class I want. The wizard offers an **Insert** button to immediately insert this code into my script, and a **Copy** button to copy the code to the clipboard. Listing 20.1 shows the complete wizard code.

NOTE I've added line breaks and line continuation characters (_) to Listing 20.1 so that it will fit in this book.

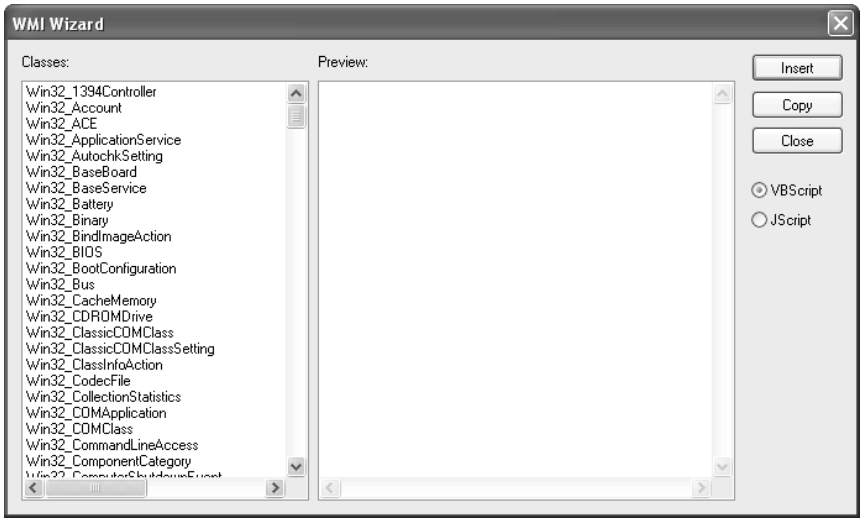

Figure 20.1 The WMI Wizard starts with a list of all available WMI classes.

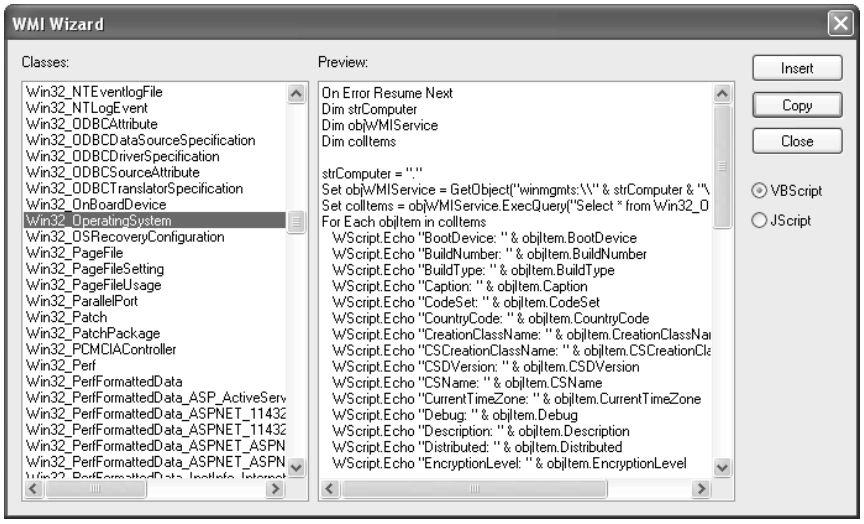

Figure 20.2 The wizard generates sample code to query the selected class.

Listing 20.1 *WizardCode.vbs*. This code queries the Win32_OperatingSystem class and outputs all of the class' attributes and their values.

```
On Error Resume Next
Dim strComputer
Dim objWMIService
Dim colItems

strComputer = "."
Set objWMIService = GetObject("winmgmts:\\" & _
 strComputer & "\root\cimv2")
Set colItems = objWMIService.ExecQuery( _
 "Select * from Win32_OperatingSystem",,48)
For Each objItem in colItems
      WScript.Echo "BootDevice: " & objItem.BootDevice
      WScript.Echo "BuildNumber: " & objItem.BuildNumber
      WScript.Echo "BuildType: " & objItem.BuildType
      WScript.Echo "Caption: " & objItem.Caption
      WScript.Echo "CodeSet: " & objItem.CodeSet
      WScript.Echo "CountryCode: " & objItem.CountryCode
      WScript.Echo "CreationClassName: " & objItem.CreationClassName
      WScript.Echo "CSCreationClassName: " & _
  objItem.CSCreationClassName
      WScript.Echo "CSDVersion: " & objItem.CSDVersion
```

```
    WScript.Echo "CSName: " & objItem.CSName
    WScript.Echo "CurrentTimeZone: " & objItem.CurrentTimeZone
    WScript.Echo "Debug: " & objItem.Debug
    WScript.Echo "Description: " & objItem.Description
    WScript.Echo "Distributed: " & objItem.Distributed
    WScript.Echo "EncryptionLevel: " & objItem.EncryptionLevel
    WScript.Echo "ForegroundApplicationBoost: " & _
objItem.ForegroundApplicationBoost
    WScript.Echo "FreePhysicalMemory: " & _
objItem.FreePhysicalMemory
    WScript.Echo "FreeSpaceInPagingFiles: " & _
objItem.FreeSpaceInPagingFiles
    WScript.Echo "FreeVirtualMemory: " & objItem.FreeVirtualMemory
    WScript.Echo "InstallDate: " & objItem.InstallDate
    WScript.Echo "LargeSystemCache: " & objItem.LargeSystemCache
    WScript.Echo "LastBootUpTime: " & objItem.LastBootUpTime
    WScript.Echo "LocalDateTime: " & objItem.LocalDateTime
    WScript.Echo "Locale: " & objItem.Locale
    WScript.Echo "Manufacturer: " & objItem.Manufacturer
    WScript.Echo "MaxNumberOfProcesses: " &
objItem.MaxNumberOfProcesses
    WScript.Echo "MaxProcessMemorySize: " &
objItem.MaxProcessMemorySize
    WScript.Echo "Name: " & objItem.Name
    WScript.Echo "NumberOfLicensedUsers: " &
objItem.NumberOfLicensedUsers
    WScript.Echo "NumberOfProcesses: " & objItem.NumberOfProcesses
    WScript.Echo "NumberOfUsers: " & objItem.NumberOfUsers
    WScript.Echo "Organization: " & objItem.Organization
    WScript.Echo "OSLanguage: " & objItem.OSLanguage
    WScript.Echo "OSProductSuite: " & objItem.OSProductSuite
    WScript.Echo "OSType: " & objItem.OSType
    WScript.Echo "OtherTypeDescription: " &
objItem.OtherTypeDescription
    WScript.Echo "PlusProductID: " & objItem.PlusProductID
    WScript.Echo "PlusVersionNumber: " & objItem.PlusVersionNumber
    WScript.Echo "Primary: " & objItem.Primary
    WScript.Echo "ProductType: " & objItem.ProductType
    WScript.Echo "QuantumLength: " & objItem.QuantumLength
    WScript.Echo "QuantumType: " & objItem.QuantumType
    WScript.Echo "RegisteredUser: " & objItem.RegisteredUser
    WScript.Echo "SerialNumber: " & objItem.SerialNumber
    WScript.Echo "ServicePackMajorVersion: " & _
```

continues

```
objItem.ServicePackMajorVersion
      WScript.Echo "ServicePackMinorVersion: " & _
  objItem.ServicePackMinorVersion
      WScript.Echo "SizeStoredInPagingFiles: " & _
  objItem.SizeStoredInPagingFiles
      WScript.Echo "Status: " & objItem.Status
      WScript.Echo "SuiteMask: " & objItem.SuiteMask
      WScript.Echo "SystemDevice: " & objItem.SystemDevice
      WScript.Echo "SystemDirectory: " & objItem.SystemDirectory
      WScript.Echo "SystemDrive: " & objItem.SystemDrive
      WScript.Echo "TotalSwapSpaceSize: " & _
  objItem.TotalSwapSpaceSize
      WScript.Echo "TotalVirtualMemorySize: " & _
  objItem.TotalVirtualMemorySize
      WScript.Echo "TotalVisibleMemorySize: " & _
  objItem.TotalVisibleMemorySize
      WScript.Echo "Version: " & objItem.Version
      WScript.Echo "WindowsDirectory: " & objItem.WindowsDirectory
Next
```

The wizard's code pulls more information than I want, and it's displaying the information in message boxes, rather than writing them to a file, but the code makes a great place to start. I can easily modify it to meet my needs.

The script is designed! I identified the three major tasks that the script needs to be able to complete, and I've created some prototype code that can be adapted to the script's exact requirements. In short, I now know how to do everything I need; I just need to rearrange it and customize it.

What, No Wizard?

If you're not using PrimalScript, there are some other tools you can use to make WMI scripting easier. In Chapter 18, for example, I introduced Microsoft's Scriptomatic tool, which performs a similar function to the PrimalScript WMI Wizard. You can also dive into the WMI documentation in the MSDN Library (http://msdn.microsoft.com/library), which documents each WMI class and includes some scripting examples.

Writing Functions and Subroutines

The one bit of functionality that seems to be standalone is the code generated by the wizard, which will do my WMI querying for me. I may need to use that code in another script someday, and I'll definitely be using it over and over in the script I'm writing now, so it makes sense to write it as a function.

I want the function to accept a computer name, query that computer for specific operating system information, and then compile all that information into a neatly formatted string. The function should return the string to the main script, which can then write it to a file or whatever.

Adapting the wizard's code isn't too difficult. Listing 20.2 shows my new `GetOSIno()` function. Note that this isn't intended to be run as a standalone script; as a function, it must be called by another script, which must provide the name of the computer to connect to as the function's input parameter.

Listing 20.2 *GetOSInfo.vbs.* This function queries a computer's operating system information and returns the results in a string.

```
Function GetOSInfo(sComputer)

      'declare variables
      Dim objWMIService
      Dim colItems
      Dim strOutput

      'get WMI service
      Set objWMIService = GetObject("winmgmts:\\" & _
       strComputer & "\root\cimv2")

      'get item collection
      Set colItems = objWMIService.ExecQuery( _
       "Select * from Win32_OperatingSystem",,48)

      'init output string
      sOutput = String(70,"-")
      sOutput = sOutput & sComputer

      'append info to output string
      For Each objItem in colItems
```

continues

```
        strOutput = strOutput & "BuildNumber: " & _
         objItem.BuildNumber & vbCrLf
        strOutput = strOutput & "BuildType: " & _
         objItem.BuildType & vbCrLf
        strOutput = strOutput & "Caption: " & _
         objItem.Caption & vbCrLf
        strOutput = strOutput & "EncryptionLevel: " & _
         objItem.EncryptionLevel & vbCrLf
        strOutput = strOutput & "InstallDate: " & _
         objItem.InstallDate & vbCrLf
        strOutput = strOutput & "Manufacturer: " & _
         objItem.Manufacturer & vbCrLf
        strOutput = strOutput & "MaxNumberOfProcesses: " & _
         objItem.MaxNumberOfProcesses & vbCrLf
strOutput = strOutput & "MaxProcessMemorySize: " & _
        objItem.MaxProcessMemorySize & vbCrLf
 strOutput = strOutput & "Name: " & _
        objItem.Name & vbCrLf
        strOutput = strOutput & _
         "NumberOfLicensedUsers: " & _
         objItem.NumberOfLicensedUsers & vbCrLf
        strOutput = strOutput & "NumberOfProcesses: " & _
         objItem.NumberOfProcesses & vbCrLf
        strOutput = strOutput & "NumberOfUsers: " & _
         objItem.NumberOfUsers & vbCrLf
        strOutput = strOutput & "OSProductSuite: " & _
         objItem.OSProductSuite & vbCrLf
        strOutput = strOutput & "OSType: " & _
         objItem.OSType & vbCrLf
        strOutput = strOutput & "OtherTypeDescription: " & _
         objItem.OtherTypeDescription & vbCrLf
        strOutput = strOutput & "Primary: " & _
         objItem.Primary & vbCrLf
        strOutput = strOutput & "ProductType: " & _
         objItem.ProductType & vbCrLf
        strOutput = strOutput & "RegisteredUser: " & _
         objItem.RegisteredUser & vbCrLf
        strOutput = strOutput & "SerialNumber: " & _
         objItem.SerialNumber & vbCrLf
        strOutput = strOutput & _
         "ServicePackMajorVersion: " & _
         objItem.ServicePackMajorVersion & vbCrLf
        strOutput = strOutput & _
         "ServicePackMinorVersion: " & _
```

```
            objItem.ServicePackMinorVersion & vbCrLf
           strOutput = strOutput & "Version: " & _
            objItem.Version & vbCrLf
           strOutput = strOutput & "WindowsDirectory: " & _
            objItem.WindowsDirectory & vbCrLf
      Next

 'return results
      GetOSInfo = sOutput

End Function
```

I didn't have to do much to adapt the script. First, I deleted all the lines that I didn't want in my script. I changed all the `WScript.Echo` commands to `strOutput = strOutput &`, which appends the information into a string rather than displays it in a message box. I also added `& vbCrLf` to the end of each line, which adds a carriage return and linefeed character. Those help keep the final output file looking nice.

I also dressed up the code at the beginning of the function.

```
'declare variables
Dim objWMIService
Dim colItems
Dim strOutput

'get WMI service
Set objWMIService = GetObject("winmgmts:\\" & _
 strComputer & "\root\cimv2")

'get item collection
Set colItems = objWMIService.ExecQuery( _
 "Select * from Win32_OperatingSystem",,48)

'init output string
sOutput = String(70,"-")
sOutput = sOutput & sComputer
```

I added some comments to document the code—PrimalScript isn't so good about that—and I initialized my `sOutput` variable. I also started `sOutput` off to contain a line of 70 hyphens, and the name of the computer I'm

querying. These extra touches help make the final output file easier to read and more useful.

Writing the Main Script

The function was probably the toughest part to write; with that out of the way, I can adapt my prototype code to create the main script, shown in Listing 20.3.

Listing 20.3 *MainScript.vbs.* Queries the domain, creates the output file, and calls the custom function I already wrote.

```
Dim sDomain
sDomain = InputBox("Enter domain to inventory")

'connect to domain and retrieve
'a list of member objects
Dim oDomain
Set oDomain = GetObject("WinNT://" & sDomain)

'get the filesystemobject
Dim oFSO
Set oFSO = CreateObject("Scripting.FileSystemObject")

'open an output file
Dim oOutput
Set oOutput = oFSO.CreateTextFile("\\server1\public\output.txt")

'run through the objects
Dim oObject, sComputerName, sDetails
For Each oObject In oDomain

 'is this object a computer?
 If oObject.Class = "Computer" Then

  'yes - get computer name
  sComputerName = oObject.Name

  'get OS info
  sDetails = GetOSInfo(sComputerName)
```

```
  'write info to the file
  oOutput.Write sDetails

 End If
Next

'close the output file
oOutput.Close

'release objects
Set oOutput = Nothing
Set oFSO = Nothing
Set oObject = nothing
Set oDomain = Nothing

'display completion message
WScript.Echo "Output saved to \\server1\public\output.txt"
```

I'll provide my usual walk-through of this script in a bit; for now, try to pick out the adapted pieces of prototype code. Notice where I'm querying the domain, opening and writing to the text file, closing the text file, and calling the `GetOSInfo()` function.

➤➤ Inventorying the Domain

Listing 20.4 shows the complete, ready-to-run script. Get this ready to run, but don't execute it just yet. In the next section, I'll cover testing and troubleshooting this script.

Listing 20.4 *InventoryDomain.vbs.* The complete domain inventory script.

```
'get domain name
Dim sDomain
sDomain = InputBox("Enter domain to inventory")

'connect to domain and retrieve
'a list of member objects
Dim oDomain
Set oDomain = GetObject("WinNT://" & sDomain

'get the filesystemobject
```

continues

```
Dim oFSO
Set oFSO = CreateObject("Scripting.FileSystemObject")

'open an output file
Dim oOutput
oOutput = oFSO.CreateTextFile("\\server1\public\output.txt")

'run through the objects
Dim oObject, sComputerName, sDetails
For Each oObject In oDomain

 'is this object a computer?
 If oObject.Class = "Computer" Then

  'yes - get computer name
  sComputerName = oObject.Name

  'get OS info
  sDetails = GetOSInfo(sComputerName)

  'write info to the file
  oOutput.Write sDetails

 End If
Next

'close the output file
oOutput.Close

'release objects
Set oOutput = Nothing
Set oFSO = Nothing
Set oObject = nothing
Set oDomain = Nothing

'display completion message
WScript.Echo "Output saved to \\server1\public\output.txt"

Function GetOSInfo(sComputer)

      'declare variables
      Dim objWMIService
      Dim colItems
      Dim strOutput
```

```
'get WMI service
Set objWMIService = GetObject("winmgmts:\\" & _
 strComputer & "\root\cimv2")

'get item collection
Set colItems = objWMIService.ExecQuery( _
 "Select * from Win32_OperatingSystem",,48)

'init output string
sOutput = String(70,"-")
sOutput = sOutput & sComputer

'append info to output string
For Each objItem in colItems
      strOutput = strOutput & "BuildNumber: " & _
       objItem.BuildNumber & vbCrLf
      strOutput = strOutput & "BuildType: " & _
       objItem.BuildType & vbCrLf
      strOutput = strOutput & "Caption: " & _
       objItem.Caption & vbCrLf
      strOutput = strOutput & "EncryptionLevel: " & _
       objItem.EncryptionLevel & vbCrLf
      strOutput = strOutput & "InstallDate: " & _
       objItem.InstallDate & vbCrLf
      strOutput = strOutput & "Manufacturer: " & _
       objItem.Manufacturer & vbCrLf
      strOutput = strOutput & "MaxNumberOfProcesses: " & _
       objItem.MaxNumberOfProcesses & vbCrLf
strOutput = strOutput & "MaxProcessMemorySize: " & _
       objItem.MaxProcessMemorySize & vbCrLf
 strOutput = strOutput & "Name: " & _
       objItem.Name & vbCrLf
      strOutput = strOutput & _
       "NumberOfLicensedUsers: " & _
       objItem.NumberOfLicensedUsers & vbCrLf
      strOutput = strOutput & "NumberOfProcesses: " & _
       objItem.NumberOfProcesses & vbCrLf
      strOutput = strOutput & "NumberOfUsers: " & _
       objItem.NumberOfUsers & vbCrLf
      strOutput = strOutput & "OSProductSuite: " & _
       objItem.OSProductSuite & vbCrLf
      strOutput = strOutput & "OSType: " & _
       objItem.OSType & vbCrLf
```

continues

```
            strOutput = strOutput & "OtherTypeDescription: " & _
             objItem.OtherTypeDescription & vbCrLf
            strOutput = strOutput & "Primary: " & _
             objItem.Primary & vbCrLf
            strOutput = strOutput & "ProductType: " & _
             objItem.ProductType & vbCrLf
            strOutput = strOutput & "RegisteredUser: " & _
             objItem.RegisteredUser & vbCrLf
            strOutput = strOutput & "SerialNumber: " & _
             objItem.SerialNumber & vbCrLf
            strOutput = strOutput & _
             "ServicePackMajorVersion: " & _
             objItem.ServicePackMajorVersion & vbCrLf
            strOutput = strOutput & _
             "ServicePackMinorVersion: " & _
             objItem.ServicePackMinorVersion & vbCrLf
            strOutput = strOutput & "Version: " & _
             objItem.Version & vbCrLf
            strOutput = strOutput & "WindowsDirectory: " & _
     Next             objItem.WindowsDirectory & vbCrLf

 'return results
      GetOSInfo = sOutput

End Function
```

You need to change where this script puts its output file before using it in your environment. The script prompts for the domain name, so you won't have to make any changes there.

➤➤ Inventorying the Domain—Explained

The script starts by prompting for the domain name. This allows the script to be used in a multidomain environment. The domain name is stored in a string variable.

```
'get domain name
Dim sDomain
sDomain = InputBox("Enter domain to inventory")
```

Next, the script uses ADSI to connect to the domain and retrieve a list of all domain objects. This may be a lengthy operation in a large domain, because computer, user, and all other objects are included in the results.

```
'connect to domain and retrieve
'a list of member objects
Dim oDomain
Set oDomain = GetObject("WinNT://" & sDomain
```

The script creates a new FileSystemObject and assigns it to a variable.

```
'get the filesystemobject
Dim oFSO
Set oFSO = CreateObject("Scripting.FileSystemObject")
```

The script now creates a new text file by using the FileSystemObject's `CreateTextFile` method. The method returns a TextStream object, which is assigned to the variable `oOutput`.

```
'open an output file
Dim oOutput
oOutput = oFSO.CreateTextFile("\\server1\public\output.txt")
```

`oDomain` now represents all of the objects in the domain; I'll use a `For Each…Next` loop to iterate through each object in turn. Within the loop, `oObject` will represent the current object.

```
'run through the objects
Dim oObject, sComputerName, sDetails
For Each oObject In oDomain
```

Because `oDomain` contains more than just computers, I need to check each object to see if its `Class` property equals "Computer." That way, I can just work with the computer objects and skip the rest.

```
 'is this object a computer?
 If oObject.Class = "Computer" Then
```

For objects that are a computer, I pull the computer name into a variable. Then, I assign the results of `GetOSInfo()` to variable `sDetails`. Finally, I write `sDetails` to the output text file using the TextStream object's `Write`

method. Closing up the loop with `Next` moves on to the next object in the domain.

```
  'yes - get computer name
  sComputerName = oObject.Name

  'get OS info
  sDetails = GetOSInfo(sComputerName)

  'write info to the file
  oOutput.Write sDetails

 End If
Next
```

When I'm done with all the objects, I close the output file, release all the objects I created by setting them equal to `Nothing`, and then display a simple completion message.

```
'close the output file
oOutput.Close

'release objects
Set oOutput = Nothing
Set oFSO = Nothing
Set oObject = nothing
Set oDomain = Nothing

'display completion message
WScript.Echo "Output saved to \\server1\public\output.txt"
```

Here's that function I wrote earlier. It starts with basic variable declaration.

```
Function GetOSInfo(sComputer)

      'declare variables
      Dim objWMIService
      Dim colItems
      Dim strOutput
```

Next is pure wizard code, which uses `GetObject` to connect to the specified computer's WMI service.

```
'get WMI service
Set objWMIService = GetObject("winmgmts:\\" & _
 strComputer & "\root\cimv2")
```

After I am connected, I execute a query to retrieve the `Win32_OperatingSystem` class.

```
'get item collection
Set colItems = objWMIService.ExecQuery( _
 "Select * from Win32_OperatingSystem",,48)
```

I set up my output string to include a line of hyphens and the current computer name.

```
'init output string
sOutput = String(70,"-")
sOutput = sOutput & sComputer
```

Finally, I append the WMI information to the output string.

```
'append info to output string
For Each objItem in colItems
      strOutput = strOutput & "BuildNumber: " & _
       objItem.BuildNumber & vbCrLf
      strOutput = strOutput & "BuildType: " & _
       objItem.BuildType & vbCrLf
      strOutput = strOutput & "Caption: " & _
       objItem.Caption & vbCrLf
      strOutput = strOutput & "EncryptionLevel: " & _
       objItem.EncryptionLevel & vbCrLf
      strOutput = strOutput & "InstallDate: " & _
       objItem.InstallDate & vbCrLf
      strOutput = strOutput & "Manufacturer: " & _
       objItem.Manufacturer & vbCrLf
      strOutput = strOutput & "MaxNumberOfProcesses: " & _
       objItem.MaxNumberOfProcesses & vbCrLf
strOutput = strOutput & "MaxProcessMemorySize: " & _
       objItem.MaxProcessMemorySize & vbCrLf
```

continues

```
        strOutput = strOutput & "Name: " & _
              objItem.Name & vbCrLf
             strOutput = strOutput & _
              "NumberOfLicensedUsers: " & _
              objItem.NumberOfLicensedUsers & vbCrLf
             strOutput = strOutput & "NumberOfProcesses: " & _
              objItem.NumberOfProcesses & vbCrLf
             strOutput = strOutput & "NumberOfUsers: " & _
              objItem.NumberOfUsers & vbCrLf
             strOutput = strOutput & "OSProductSuite: " & _
              objItem.OSProductSuite & vbCrLf
             strOutput = strOutput & "OSType: " & _
              objItem.OSType & vbCrLf
             strOutput = strOutput & "OtherTypeDescription: " & _
              objItem.OtherTypeDescription & vbCrLf
             strOutput = strOutput & "Primary: " & _
              objItem.Primary & vbCrLf
             strOutput = strOutput & "ProductType: " & _
              objItem.ProductType & vbCrLf
             strOutput = strOutput & "RegisteredUser: " & _
              objItem.RegisteredUser & vbCrLf
             strOutput = strOutput & "SerialNumber: " & _
              objItem.SerialNumber & vbCrLf
             strOutput = strOutput & _
              "ServicePackMajorVersion: " & _
              objItem.ServicePackMajorVersion & vbCrLf
             strOutput = strOutput & _
              "ServicePackMinorVersion: " & _
              objItem.ServicePackMinorVersion & vbCrLf
             strOutput = strOutput & "Version: " & _
              objItem.Version & vbCrLf
             strOutput = strOutput & "WindowsDirectory: " & _
              objItem.WindowsDirectory & vbCrLf
    Next
```

With the main script finished, I return the output string as the function's result.

```
 'return results
      GetOSInfo = sOutput

End Function
```

There you have it—a nice, easy-to-use administrative script that uses both WMI and ADSI to accomplish a useful task.

Testing the Script

If you jumped ahead and already tried to execute the final script, you realize that it's flawed. If you haven't, go ahead and give it a whirl now. Take a few minutes to see if you can track down the problem. There are actually three errors, and here are some hints.

- One is a simple typo.
- One is a sort of logic error, where something isn't being used properly for the situation.
- The last one is a typo, and could have been avoided if I had followed my own advice from earlier in the book.

Can you find them all? The first one is an easy mistake: I simply forgot a closing parentheses.

```
'connect to domain and retrieve
'a list of member objects
Dim oDomain
Set oDomain = GetObject("WinNT://" & sDomain
```

The correct code should be `Set oDomain = GetObject("WinNT://" & sDomain)`. The next one's a bit trickier.

```
'open an output file
Dim oOutput
oOutput = oFSO.CreateTextFile("\\server1\public\output.txt")
```

Can you see it? I'm using `oOutput` to represent an object, but I forgot to use the `Set` keyword when making the assignment. VBScript requires `Set` whenever you're assigning an object to a variable. The corrected code looks like this.

```
'open an output file
Dim oOutput
Set oOutput = oFSO.CreateTextFile("\\server1\public\
  output.txt")
```

The last error is tricky, too. It's in the `GetOSInfo()` function.

```
Function GetOSInfo(sComputer)

      'declare variables
      Dim objWMIService
      Dim colItems
      Dim strOutput

      'get WMI service
      Set objWMIService = GetObject("winmgmts:\\" & _
       strComputer & "\root\cimv2")
```

Did you find it? The problem is that I used the wizard-generated code, which uses "str" as a prefix for string variables. I'm in the habit of using the shorter prefix "s" for string variables, and that's where my problem lies. In the function definition, I declared `sComputer`, but in the line of code that connects to the WMI service, I used `strComputer`. I continued using `sComputer` elsewhere, so `strComputer` is wrong. Here's the corrected code snippet.

```
Function GetOSInfo(sComputer)

      'declare variables
      Dim objWMIService
      Dim colItems
      Dim strOutput

      'get WMI service
      Set objWMIService = GetObject("winmgmts:\\" & _
       sComputer & "\root\cimv2")
```

The problem with this error is that it doesn't cause a problem for the script; the script will execute just fine. You just won't get any results, because the script would try to connect to a computer named "". I mentioned that I could have avoided this problem by following my own advice. Had I included `Option Explicit`, VBScript would have produced an error on the offending line of code, because `strComputer` wasn't declared. `sComputer`, on the other hand, is implicitly declared because it's part of a function declaration. You'll notice that I did the same thing with `strOutput` and `sOutput`, meaning they'll have to be corrected, too.

Just to make sure you've got it all, Listing 20.5 includes the complete, corrected script. Remember that this script is also available on the CD that accompanies this book.

Listing 20.5 *InventoryDomain2.vbs.* This corrected script produces the expected results.

```
'get domain name
Dim sDomain
sDomain = InputBox("Enter domain to inventory")

'connect to domain and retrieve
'a list of member objects
Dim oDomain
Set oDomain = GetObject("WinNT://" & sDomain)

'get the filesystemobject
Dim oFSO
Set oFSO = CreateObject("Scripting.FileSystemObject")

'open an output file
Dim oOutput
Set oOutput = oFSO.CreateTextFile("\\server1\public\output.txt")

'run through the objects
Dim oObject, sComputerName, sDetails
For Each oObject In oDomain

 'is this object a computer?
 If oObject.Class = "Computer" Then

  'yes - get computer name
  sComputerName = oObject.Name

  'get OS info
  sDetails = GetOSInfo(sComputerName)

  'write info to the file
  oOutput.Write sDetails

 End If
Next
```

continues

```
'close the output file
oOutput.Close

'release objects
Set oOutput = Nothing
Set oFSO = Nothing
Set oObject = nothing
Set oDomain = Nothing

'display completion message
WScript.Echo "Output saved to \\server1\public\output.txt"

Function GetOSInfo(sComputer)

      'declare variables
      Dim objWMIService
      Dim colItems
      Dim strOutput

      'get WMI service
      Set objWMIService = GetObject("winmgmts:\\" & _
       sComputer & "\root\cimv2")

      'get item collection
      Set colItems = objWMIService.ExecQuery( _
       "Select * from Win32_OperatingSystem",,48)

      'init output string
      strOutput = String(70,"-")
      strOutput = strOutput & sComputer

      'append info to output string
      For Each objItem in colItems
            strOutput = strOutput & "BuildNumber: " & _
             objItem.BuildNumber & vbCrLf
            strOutput = strOutput & "BuildType: " & _
             objItem.BuildType & vbCrLf
            strOutput = strOutput & "Caption: " & _
             objItem.Caption & vbCrLf
            strOutput = strOutput & "EncryptionLevel: " & _
             objItem.EncryptionLevel & vbCrLf
            strOutput = strOutput & "InstallDate: " & _
             objItem.InstallDate & vbCrLf
            strOutput = strOutput & "Manufacturer: " & _
```

```
            objItem.Manufacturer & vbCrLf
           strOutput = strOutput & "MaxNumberOfProcesses: " & _
            objItem.MaxNumberOfProcesses & vbCrLf
      strOutput = strOutput & "MaxProcessMemorySize: " & _
            objItem.MaxProcessMemorySize & vbCrLf
       strOutput = strOutput & "Name: " & _
            objItem.Name & vbCrLf
           strOutput = strOutput & _
            "NumberOfLicensedUsers: " & _
            objItem.NumberOfLicensedUsers & vbCrLf
           strOutput = strOutput & "NumberOfProcesses: " & _
            objItem.NumberOfProcesses & vbCrLf
           strOutput = strOutput & "NumberOfUsers: " & _
            objItem.NumberOfUsers & vbCrLf
           strOutput = strOutput & "OSProductSuite: " & _
            objItem.OSProductSuite & vbCrLf
           strOutput = strOutput & "OSType: " & _
            objItem.OSType & vbCrLf
           strOutput = strOutput & "OtherTypeDescription: " & _
            objItem.OtherTypeDescription & vbCrLf
           strOutput = strOutput & "Primary: " & _
            objItem.Primary & vbCrLf
           strOutput = strOutput & "ProductType: " & _
            objItem.ProductType & vbCrLf
           strOutput = strOutput & "RegisteredUser: " & _
            objItem.RegisteredUser & vbCrLf
           strOutput = strOutput & "SerialNumber: " & _
            objItem.SerialNumber & vbCrLf
           strOutput = strOutput & _
            "ServicePackMajorVersion: " & _
            objItem.ServicePackMajorVersion & vbCrLf
           strOutput = strOutput & _
            "ServicePackMinorVersion: " & _
            objItem.ServicePackMinorVersion & vbCrLf
           strOutput = strOutput & "Version: " & _
            objItem.Version & vbCrLf
           strOutput = strOutput & "WindowsDirectory: " & _
            objItem.WindowsDirectory & vbCrLf
      Next

 'return results
      GetOSInfo = sOutput

End Function
```

Testing a large script like this is much easier with the Script Debugger. You can spot lines that are causing trouble just by following the execution path.

For more information on the Script Debugger, see "Testing the Script" in Chapter 13. You can also read up on the Script Debugger in the VBScript documentation at http://msdn.microsoft.com/scripting.

Review

Pulling together ADSI and WMI into a single script offers some powerful functionality. More importantly, though, the example in this chapter should make you feel more comfortable with the sometimes-daunting task of creating a script from scratch. Just break down the tasks that need to be completed, and then develop some prototype code for each task. Use wizards, examples from the Web, or samples from this book to help create prototype code. After all, there's no sense reinventing the wheel when there's a large library of samples on the Web and in this book to work with!

With your task list and prototype out of the way, you can start assembling the script. Write functions and subs to perform repetitive tasks, or tasks that you may want to reuse in future scripts. Write the main script, and then start testing. With this methodology in mind, most scripts can be whipped together quickly!

COMING UP

Web pages offer an exciting way to create your own centrally located, easily accessible administrative tools. In the next chapter, I'll introduce you to Active Server Pages, and in the following chapters, I'll show you how to easily and quickly apply your scripting skills to create great administrative Web pages.