**Working with software: Step-by-step exercises**

In the first exercise, you will explore the use of *Win32\_Product* and classes provided by the WMI Microsoft Installer (MSI) provider. In the second exercise, you will work with the environment provider.

**Note**

The first exercise, which takes advantage of the *Win32\_Product* class, is for illustrative purposes, to demonstrate the power of WMI and Windows PowerShell. The *Win32\_Product* class, when queried, will initiate an MSI consistency check, which can have undesirable effects. When working with this class, use caution.

**Using WMI to find installed software**

**1.** Open the Windows PowerShell ISE or your favorite script editor.

**2.** On the first line, you will use the variable *$Query* to hold your WMI query. This query will select everything from the *Win32\_Product* WMI class.

**3.** Because this query can take a rather long time to complete (depending on the speed of your machine, CPU load, and number of installed applications), use the *Write-Host* cmdlet to inform the user that the script could take a while to run. As long as you’re using *Write-Host*, let’s have a little fun and specify the *-ForegroundColor* parameter of the *Write-Host* cmdlet, which will change the color of your font. I chose blue, but you can choose any color you want. Use the *`n* escape sequence to specify a new line at the end of your command. I used the grave accent character to break the line of code for readability, but this certainly is not necessary for you.

**4.** On the next line, use the *Get-CimInstance* cmdlet. Supply the *-Query* parameter with the value contained in the *$Query* variable.

**5.** Save and run your script. Call it *<yourname>MSI\_InstalledApplications.ps1*. You should get output similar to that shown here. If you do not, compare it with *MSI\_InstalledApplications.ps1*.

Counting Installed Products. This may take a little while.  
  
  
Name    Caption    Vendor                  Version     IdentifyingNumber  
----    -------    ------                  -------     -----------------  
NWT 11  NWT 11     NWTraders Corporation   11.4.0     {5EAF3FAA-C4B6-5741-81B4-64CD...

**6.** Now you’ll add a timer to your script to find out how long it takes to execute. On the first line of your script, above the *$Query* line, declare a variable called *$dteStart* and assign the date object that is returned by the *Get-Date*cmdlet to it.

**7.** At the end of your script, under the last *Get-CimInstance* command, declare a variable called *$dteEnd* and assign the date object that is returned by the *Get-Date* cmdlet to it.

**8.** On the next line, declare a variable called *$dteDiff* and assign the date object that is returned by the *New-TimeSpan* cmdlet to it. Use the *New-TimeSpan* cmdlet to subtract the two date objects contained in the *$dteStart* and *$dteEnd* variables. The *$dteStart* variable will go first.

**9.** Use the *Write-Host* cmdlet to print the total number of seconds it took for the script to run. This value is contained in the *totalSeconds* property of the date object held in the *$dteDiff*

**Upload script and screenshot of results**