**Obtaining service information: Step-by-step exercises**

In this exercise, you will explore the use of the *Get-Service* cmdlet as you retrieve service information from your computer. You will sort and filter the output from the *Get-Service* cmdlet. In the second exercise, you will use WMI to retrieve similar information. You should compare the two techniques for ease of use and completeness of data.

**Obtaining Windows service information by using the *Get-Service* cmdlet**

**1.** Start the Windows PowerShell console.

**2.** From the Windows PowerShell prompt, use the *Get-Service* cmdlet to obtain a listing of all the services and their associated status. This is shown here.

**Insert command and screen shot of results**

**3.** Use the *Sort-Object* cmdlet to sort the listing of services. Specify the *status* property for *Sort-Object*. To sort the data based upon status, pipeline the results of the *Get-Service* cmdlet into the *Sort-Object* cmdlet. Use the *sort* alias for the *Sort-Object* cmdlet to reduce the amount of typing. The results are shown here.

**Insert command and screen shot of results**

**4.** Use the *Get-Service* cmdlet to produce a listing of services. Sort the resulting list of services alphabetically by name. To do this, use the *Sort-Object* cmdlet to sort the listing of services by the *name* property. Pipeline the object returned by the *Get-Service* cmdlet into the *Sort-Object* cmdlet. The command to do this, using the *sort* alias for *Sort-Object*, is shown here.

**Insert command and screen shot of results**

**5.** Use the *Get-Service* cmdlet to produce a listing of services. Sort the objects returned by both the name and the status of the service. The command to do this is shown here.

**Insert command and screen shot of results**

**6.** Use the *Get-Service* cmdlet to return objects containing service information, using the *DisplayName* parameter and specifying *\*server\** to retrieve all service objects that contain the word *server* in the display name.

**Insert command and screen shot of results**

**7.** Use the *Get-Service* cmdlet with the *Name* parameter to retrieve an object that represents the BITS service. The code that does this is shown here.

**Insert command and screen shot of results**

**8.** Press the Up Arrow key to retrieve the previous command that retrieves the BITS service. Store the resulting object in a variable called *$a*. This code is shown here.

**Insert command and screen shot of results**

**9.** Pipeline the object contained in the *$a* variable into the *Get-Member* cmdlet. You can use the *gm* alias to simplify typing. This code is shown here.

**Insert command and screen shot of results**

**10.** By using the object contained in the *$a* variable, obtain the status of the Bits service. The code that does this is shown here.

**Insert command and screen shot of results**

**11.** If the Bits service is running, stop it. To do so, use the *Stop-Service* cmdlet. Instead of pipelining the object in the *$a* variable, you use the *-InputObject* argument from the *Stop-Service* cmdlet. The code to do this is shown here.

**Insert command and screen shot of results**

**12.** If the Bits service stops, use the *Start-Service* cmdlet instead of the *Stop-Service* cmdlet. Use the *-inputobject* argument to supply the object contained in the *$a* variable to the cmdlet. This is shown here.

**Insert command and screen shot of results**

**13.** Query the *status* property of the object contained in the *$a* variable to confirm that the Bits service’s status has changed. This is shown here.

**Insert command and screen shot of results**

This concludes this step-by-step exercise.

**Note**

If you are working with a service that has its startup type set to Disabled, Windows PowerShell will not be able to start it and will return an error. If you do not have administrator rights, Windows PowerShell will be unable to stop the service.