### Using Windows PowerShell remoting and WMI: Step-by-step exercises

In this exercise, you will practice using Windows PowerShell remoting to run remote commands. For the purpose of this exercise, you can use your local computer, but commands designed to fail (in the exercise) will more than likely succeed instead of creating the errors that appear here.

**Using Windows PowerShell remoting to retrieve remote information**

**1.** Log on to your computer with a user account that does not have administrator rights.

**2.** Open the Windows PowerShell console.

**3.** Use the Get-CimInstance cmdlet to retrieve process information from a remote system that has WMI remoting enabled on it. Do not supply alternate credentials.

**Insert command and screen shot of results**

**4.** The command fails due to an Access Denied error. Now create a new CIM session to the remote system and connect with alternate credentials. Store the CIM session in a variable named $session. (Use a remote system accessible to you and credentials appropriate to that system. You will use your local computer as your remote system.)

**Insert command and screen shot of results**

**5.** Use the stored CIM session from the $session variable to retrieve process information from the remote system.

**Insert command and screen shot of results**

**6.** Use the stored CIM session from the $session variable to retrieve the name and the status of all services on the remote system. Sort the output by state, and format a table with the name and the state.

**Insert command and screen shot of results**

**7.** Use the Get-WmiObject cmdlet to run a WMI command on a remote system. Use the Win32\_BIOS WMI class and target the same remote system you used earlier. Specify appropriate credentials for the connection.

**Insert command and screen shot of results**

**8.** Use Windows PowerShell remoting by using the Invoke-Command cmdlet to run a WMI command against a remote system. Use the credentials you stored earlier. Use the Get-CimInstance cmdlet to retrieve BIOS information from WMI.

**Insert command and screen shot of results**

**Creating and receiving WMI jobs**

**9.** Open the Windows PowerShell console (if it is not already open) as a non-elevated user.

**10.** Use the Get-WmiObject cmdlet to retrieve BIOS information from a remote system. Use the -AsJobswitch parameter to run the command as a job. Use the credentials you stored in the $credential variable in the previous exercise.

**Insert command and screen shot of results**

**11.** Check on the success or failure of the job by using the Get-Job cmdlet. Make sure you use the job ID from the previous command.

**Insert command and screen shot of results**

**12.** If the job was successful, receive the results of the job by using the Receive-Job cmdlet. Do not bother with storing the results in a variable or keeping the results, because you will not need them.

**13.** Create a new Windows PowerShell session object by using the New-PSSession cmdlet. Store the results in a variable named $psSession. (Use appropriate computer names and credentials for your network/local.)

**Insert command and screen shot of results**

**14.** Use the Invoke-Command cmdlet to make the Get-WmiObject cmdlet retrieve BIOS information from the remote system. Use the session information stored in the $psSession variable. Make sure you use the -AsJob switch parameter with the command.

**Insert command and screen shot of results**

**15.** Use the Get-Job cmdlet with the job ID returned by the previous command to check on the status of the job.

**Insert command and screen shot of results**

**16.** Use the Receive-Job cmdlet to retrieve the results of the WMI command. Store the returned information in a variable named $bios. (make sure that you use the job ID number from your system).

**17.** Now query the BIOS version by accessing the version property from the $bios variable.

**Insert command and screen shot of results**

This concludes the exercise.