Lab A: Converting, Exporting, and Importing Objects

Exercise 1: Converting Objects  
▶ Task 1: Produce an HTML report listing the processes running on a computer  
**Note:** In this document, long commands are typically displayed on several lines. Doing so  
helps prevent an unintended line break in the middle of a command. However, when you type  
these commands, you should type them as a single line. That line may wrap on your screen into  
multiple lines, but the command will still work. You should press Enter only after typing the entire  
command.  
1. Log on to the DC1 virtual machine logged in as **Administrator**.  
2. To display a list of running processes, run:  
Get-Process  
  
3. To display a list of running processes, sorted in reverse alphabetic order by process name, that shows  
only the process name, ID, virtual memory, and physical memory consumption, run:  
Get-Process | Sort Name -Descending | Select Name,ID,VM,PM  
4. To view the Help for **ConvertTo-HTML**, run:  
help ConvertTo-HTML –ShowWindow  
5. To convert the process list to an HTML page, run:  
Get-Process | Sort Name -Descending | ConvertTo-HTML –Property Name,ID,VM,PM  
6. To save the HTML page in a file, run:  
Get-Process |  
Sort Name -Descending | ConvertTo-HTML –Property Name,ID,VM,PM |  
Out-File ProcReport.html  
7. To view the HTML file, run:  
Invoke-Expression .\ProcReport.html  
8. To create the modified HTML file, run:  
Get-Process |  
Sort Name -Descending | ConvertTo-HTML –Property Name,ID,VM,PM –PreContent  
"Processes" –PostContent (Get-Date) |  
Out-File ProcReport.html  
9. To display the HTML file, run:  
Invoke-Expression .\ProcReport.html  
**Results**: After completing this exercise, you will have converted objects to different forms of data.  
  
  
▶ Task 1: Create a comma-separated values (CSV) file listing the most recent 10 entries  
from the System event log.  
**Note:** When typing these commands, you should type them as a single line, and press  
Enter only once, after typing the entire line. However, in the Console application, you can also  
type these commands exactly as they are shown. Typically, that means pressing Enter after each  
vertical pipe (|) character. If you use this technique, you will have to press Enter on a blank line,  
after typing all of the lines, to execute the command.  
1. To display the System event log, run:  
Get-EventLog –Newest 10 –LogName System  
2. To convert the log to CSV, run:  
Get-EventLog –Newest 10 –LogName System |  
ConvertTo-CSV  
3. To export the log as a CSV file, run:  
Get-EventLog –Newest 10 –LogName System |  
Export-Csv SysEvents.csv  
4. To view the CSV file, run:  
Notepad SysEvents.csv  
5. To export the log and remove the comment line containing type information, run:  
Get-EventLog –Newest 10 –LogName System |  
Export-Csv SysEvents.csv –NoTypeInformation  
6. To view the revised CSV file, run:  
Notepad SysEvents.csv

▶ Task 2: Create an XML file listing services  
1. To display a list of services that shows stopped services last, run:  
Get-Service |  
Sort Status –Descending  
2. To export the service list to an XML file, run:  
Get-Service |  
Sort Status –Descending |  
Export-CliXML Services.xml  
3. To display the XML file, run:  
Notepad Services.xml  
4. To choose specified columns to be included in the file, run:  
  
Get-Service |  
Sort Status –Descending |  
Select Name,DisplayName,Status | Export-CliXML Services.xml  
5. To display the revised file, run:  
Notepad Services.xml

▶ Task 3: Produce a pipe-delimited list of the most recent 20 Security event log entries  
1. To view the Help file for **ConvertTo-HTML**, run:  
Help ConvertTo-CSV -ShowWindow  
2. To display the log entries, run:  
Get-EventLog –newest 20 –LogName Security  
3. To display only specified properties of the log entries, run:  
Get-EventLog –newest 20 –LogName Security |  
Select EventID,TimeWritten,Message  
4. To export the log entry list as a pipe-delimited file, run:  
Get-EventLog –newest 20 –LogName Security |  
Select EventID,TimeWritten,Message |  
Export-CSV Security.pdd –Delimiter '|'  
5. To display the file, run:  
Notepad Security.pdd

▶ Task 4: Import data from a pipe-delimited file  
• To import data from the pipe-delimited file, run:  
Import-Csv Security.pdd –Delimiter '|' |  
Select –First 10  
**Results**: After completing this lab, you will have imported data from and exported data to external  
storage.

Lab B: Filtering Objects

Exercise 1: Filtering Objects  
▶ Task 1: Display a list of all users in the Users container of Active Directory  
1. Log on to the 10961B-LON-CL1 virtual machine as **Adatum\Administrator**.  
2. To find a command that can list Active Directory® users, run:  
help \*user\*  
Notice the **Get-ADUser** command.  
  
3. To view the Help for the command, run:  
help Get-ADUser –ShowWindow  
Notice that the **–Filter** parameter is mandatory. Review the examples for the command.  
4. To display a list of all users, run:  
Get-ADUser –Filter \*  
5. To display a list of all users in a specified container, run:  
Get-ADUser –Filter \* -SearchBase "cn=Users,dc=Adatum,dc=com"  
▶ Task 2: Create a report that shows Security event log entries having the event ID  
4624  
1. To display a list of Security event log entries that have the event ID 4624, run:  
Get-EventLog -LogName Security |  
Where EventID -eq 4624  
2. To display the same list, showing only specified properties, run:  
Get-EventLog -LogName Security |  
Where EventID -eq 4624 |  
Select TimeWritten,EventID,Message  
3. To convert the list to an HTML file, run:  
Get-EventLog -LogName Security |  
Where EventID -eq 4624 |  
Select TimeWritten,EventID,Message |  
ConvertTo-HTML |  
Out-File EventReport.html  
4. To view the HTML file, run:  
Invoke-Expression .\EventReport.html

▶ Task 3: Display a list of encryption certificates installed on the computer  
1. To display a directory listing of all items in the **CERT:** drive, run:  
Get-ChildItem -Path CERT: -Recurse  
2. To display the members of the objects, run:  
Get-ChildItem -Path CERT: -Recurse |  
Get-Member  
3. To show only the certificates that do not have a private key, run either this:  
Get-ChildItem -Path CERT: -Recurse |  
Where HasPrivateKey -eq $False  
or this:  
Get-ChildItem -Path CERT: -Recurse |  
  
Where { $PSItem.HasPrivateKey -eq $False }  
**4.** To display the list again by using the specified filtering criteria, run:  
Get-ChildItem -Path CERT: -Recurse |  
Where { $PSItem.HasPrivateKey -eq $False -and $PSItem.NotAfter -gt (Get-Date) -and  
$PSItem.NotBefore -lt (Get-Date) }  
**5.** To display the list again by using the specified filtering criteria and showing only the specified  
properties, run:  
Get-ChildItem -Path CERT: -Recurse |  
Where { $PSItem.HasPrivateKey -eq $False -and $PSItem.NotAfter -gt (Get-Date) -and  
$PSItem.NotBefore -lt (Get-Date) } |  
Select Issuer,NotBefore,NotAfter

▶ Task 4: Create a report that shows disk volumes that are running low on space  
1. To display a list of disk volumes, run:  
Get-Volume  
If you did not know the command name, you could have run **Help \*volume\*** to discover the  
command name.  
2. To display a list of object members, run:  
Get-Volume | Get-Member  
Notice the **SizeRemaining** property.  
3. To display only volumes that have more than zero bytes of free space, run:  
Get-Volume |  
Where-Object { $PSItem.SizeRemaining -gt 0 }  
4. To display only volumes that have less than 99 percent free space, and more than zero bytes of free  
space, run:  
Get-Volume |  
Where-Object { $PSItem.SizeRemaining -gt 0 -and $PSItem.SizeRemaining / $PSItem.Size  
-lt .99 }  
5. To display only volumes that have less than 10 percent free space and more than zero bytes of free  
space, run:

Get-Volume |  
Where-Object { $PSItem.SizeRemaining -gt 0 -and $PSItem.SizeRemaining / $PSItem.Size  
-lt .1 }

This command may not produce any output on your lab computer if the computer has more than 10  
percent free space on all of its volumes.

▶ Task 5: Create a report that displays specified Control Panel items  
1. To find a command that can display Control Panel items, run:

help \*control\*  
  
Notice the **Get-ControlPanelItem** command.  
2. To display a list of Control Panel items, run:  
Get-ControlPanelItem  
3. To display items in the **System and Security** category, run:  
Get-ControlPanelItem –Category 'System and Security'  
Notice that you do not have to use **Where-Object**.  
**Results**: After completing this exercise, you will have used filtering to produce lists of management  
information that include only specified data and elements.

Lab C: Enumerating Objects

Exercise 1: Enumerating Objects

▶ Task 1: Display a list of key algorithms for all encryption certificates installed on your  
computer.  
1. To display a list of all items in the **CERT:** drive, run:  
Get-ChildItem -Path CERT: -Recurse  
2. To display the members of those objects, run:  
Get-ChildItem -Path CERT: -Recurse | Get-Member  
Notice the **GetKeyAlgorithm()** method in the list.  
3. To display a list of key algorithms for each installed certificate, run:  
Get-ChildItem -Path CERT: -Recurse |  
ForEach GetKeyAlgorithm  
4. To display the same list by using **Select-Object,** run:  
Get-ChildItem -Path CERT: -Recurse |  
Select Issuer,@{n='KeyAlgorithm';e={$PSItem.GetKeyAlgorithm()}}

▶ Task 2: Use enumeration to produce 100 random numbers  
1. To find a command that can produce random numbers, run:  
help \*random\*  
Notice the **Get-Random** command.  
2. To view the Help for the command, run:  
help Get-Random –ShowWindow  
Notice the **–SetSeed** parameter.  
3. To place 100 numeric objects into the pipeline, run:

1..100

4. To produce 100 random numbers, run:  
1..100 |  
ForEach { Get-Random –SetSeed $PSItem }

▶ Task 3: Execute a method of a Windows Management Instrumentation (WMI) object  
1. Close all applications other than the Windows PowerShell™ console.  
2. Run:  
Get-WmiObject –Class Win32\_OperatingSystem -EnableAllPrivileges  
3. To display the members of the object, run:  
Get-WmiObject –Class Win32\_OperatingSystem -EnableAllPrivileges |  
Get-Member  
4. Notice the **Reboot()** method.  
5. To restart the computer, run:  
Get-WmiObject –Class Win32\_OperatingSystem -EnableAllPrivileges |  
ForEach Reboot