PowerShell One-Liners – Take my script, please!

“Doctor, my leg hurts. What can I do?” The doctor says, “Limp!”

Hail Henny Youngman, the king of one-liners! One of the great things about PowerShell is the ability to create simple, powerful one-line scripts. This is not to say that we need to crunch everything down to single line just to be cute, but we have the ability to build effective processes by making use of the pipeline.

With shell scripting, it’s often regarded as a number of processes that we pile together with inline code, but quite often we only have some small tasks that we need to run, but they require more than one action in the script.

The long and the short of it (emphasis on short) is that we can do a lot of things with a single line of code by leveraging the pipeline.

Here is an article about piping and the pipeline from Microsoft TechNet: http://technet.microsoft.com/en-us/library/ee176927.aspx

The pipeline allows you to take some input, whether it is a file, or a query and then to perform a cumulative set of actions.

Let’s say I want to read the contents of a text file, and create an output file with only lines that contain the phrase “yourdomain.com”. To do this we use this simple one-line script:

> **Get-Content yourinputfile.txt | Where-Object {$_ -match ‘yourdomain.com’} | Out-File youroutputfile.txt**

As you can see we are reading the file, applying a where clause to the input and then writing the output to a destination. This is a classic case of a pipeline by taking one object, processing it, and then outputting the processed content.

Let's look at an example where we want to scan Active Directory users with the Quest ActiveRoles Get-QADUser CmdLet and output the Password status to screen but we want to exclude non-expiring passwords:

> **Get-QADUser -SizeLimit 0 | Select-Object samAccountName,mail,PasswordStatus | Where-Object {$_.PasswordStatus -ne "Password never expires" | Write-Host "$_.samAccountName status with email $_.mail has a password status of $_.PasswordStatus"**

This script reads the directory, selects the fields we want and then outputs to screen for us using the Write-Host CmdLet. Simple and effective.

As you can see we can do many tasks where it is **Input => Process => Output** and the Process step can contain many steps within it.

So what's the downside? One-liners aren't always appropriate for a few reasons. Primarily, if we need to loop through input multiple times, or parse the content, run complex calculations. Another reason is that we may be processing very large input which must reside in memory during the process which may require a more carefully designed script to include multiple loops and garbage collection during runtime.

This is an organic and dynamic article where I will collect my favorite one-liners by category over time. This is the starting point and I welcome you to add your own in the comments and I will be sure to add them to the collection.

Last updated on June 24, 2013

Exchange 2010

Query your Exchange servers for databases and file locations

> **Get-MailboxDatabase -Status | select ServerName,Name,DatabaseSize,EdbFilePath, | Export-Csv X:MailboxDatabase.csv**

Report Exchange 2010 Recipient Mailbox Policies (from @JeffHicks)

> **get-mailbox | sort RetentionPolicy | format-table -group RetentionPolicy**

Query your Exchange server queues and show the message count as a point in time snapshot (from @HardcoreITGuy)

> **Get-exchangeserver | where {$_.isHubTransportServer –eq $true} | get-queue | where {$_.MessageCount –gt 0}**

Query your Exchange environment for a list of ActiveSync devices sorted by Device Type and output to CSV

> **Get-ActiveSyncDevice | select-object DeviceModel,FriendlyName,DeviceOS,UserDisplayname | sort-object DeviceModel | Export-CSV "ActiveSyncDevices.csv"**

Windows

Find the installation date of Windows through WMI

([WMI]”).ConvertToDateTime((Get-WmiObject Win32_OperatingSystem).InstallDate)

Active Directory

List computer object counts categorized by Operating System (uses Quest ActiveRoles snap in)

Get-QADComputer -sizeLimit 0 | Group-Object -Property OSName,OSServicePack -noelement | Sort-Object -Property Count -Descending | Format-Table -Property Count,Name

Export all XP Computers, with all properties into a CSV format sorted by IP Address

Get-ADComputer -Filter {OperatingSystem -like “Windows XP*”} -Properties * | sort-Object IPv4Address | Export-CSV XPComputers.csv

Export a list of all non-server OS computers into a CSV format where the computer account hasn’t contacted AD for over 30 days

get-adcomputer -Filter { OperatingSystem -notlike “Windows Server*” } -Properties PasswordLastSet | Where-Object { (((Get-Date) – $_.PasswordLastSet).Days) -gt 30} | select-object Name, @{Name=”Age”;Expression={ (((Get-Date) – $_.PasswordLastSet).Days) }} | Export-CSV ComputerListByAge.csv

Much more to come so check back often and I’d love to have your contributions to add and share.