Quick Start

Get-Process # displays a list of running processes

Get-Process | Select-Object Name, Company # selects several columns

Get-Process | Select-Object Name, Company | Format-Table -AutoSize # uses minimal column width

Get-Process | Select-Object Name, Company | Format-List # displays a list instead of a table

Get-Process | Sort-Object ID -Descending # sorts on process id instead of name

Get-Process | Where-Object {$_ -vm -gt 150MB} # selects processes where virtual memory is greater than 150MB

Get-Process | Select-Object Name, @Name "Virtual Memory", Expression {$_ -vm -gt $ws1} # introduces a calculated column

Get-Process | Select-Object Name,VM,WS, @Label "TotalMemory",Expression {{int{($-vm -gt $ws1)1MB}}}. # calculated column and rounded to integer

Built-in Help functionality

Get-Help

Get-Help Get-Process -full

Get-Help Get-Process -examples # view the example commands

Get-Help about # lists all the about-articles, use the full article name to view its contents, e.g. about_scripts

Get-Command # display all commands

Get-Command *process* # display all commands containing the word "process"

Get-Command -CommandType Cmdlet # display all native PowerShell cmdlets

Get-Process | Get-Member # display the properties and methods of the output

# The following two examples help you to protect you from .. you!

Get-Process PowerShell | Stop-Process -whatif # displays the command without actually executing it

Get-Process PowerShell | Stop-Process -confirm # asks for confirmation

Aliases (and functions denoted by "*"),

All default cmdlets with "-Object" as a noun are aliased to their verb without "-Object", e.g. Sort is an alias of Sort-Object, Select is an alias of Select-Object

Get- alias | Group-Object Definition # display all possible aliases per command

<table>
<thead>
<tr>
<th>Command</th>
<th>Alias</th>
<th>Command</th>
<th>Alias</th>
<th>Alias</th>
<th>Alias</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear-Host</td>
<td>cls, clear, Get-Help</td>
<td>help *, man</td>
<td>Rename-Item</td>
<td>rni, ren</td>
<td></td>
</tr>
</tbody>
</table>
| Copy-Item | copy, cp, cp | Get-Member | gm | Select-String | sls (Ps3v+)
| ForEach-Object | foreach, % | Get-Process | gpd, ps | Set-Location | sl, cd, chair |
| Format-List | FL | Get-WmiObject | gwm | Start-Sleep | sleep |
| Format-Table | FT | Move-Item | mi, move, mv | Stop-Process | sps, kill |
| Get-ChildItem | gi, dir, ls | Get-Host | oh | Where-Object | where, ? |
| Get-Command | gcm | PowerShell_ise.exe | ise (Ps3v+) | Write-Output | echo, write |
| Get-Content | gc, type, cat | Remove-Item | ri, del, erase, rmdir, rd, rm |

New alias expected in Windows 2015: Set-Alias Ping Test-Connection. Select-String has an alias in Ps3v: sls. Shouldn't this have been 'grep'? ;)

Operators (most of them), Get-Help about Operators

Operator Meaning

+, *, /, %= add, subtract, multiply, divide, remainder assign/change/append one or more values to variables

- eq, ne Equal, not equal: 5 = 5 regular expression match: "Rick" match "[DMNR]ick"

gt, ge greater than, greater than or equals: 6 –gt 5 Array contains specific value: "red", "blue" –contains "blue"

lt, le less than, less than or equals: 5 –lt 6 Logical operators

like, notlike, clikewildcard comparison: "Samantha" -like "sam*" -f Formatting: $a=2987654; "free space: (0:N0) bytes" -f $a

Punctuation Marks

(expression) [ code block ] [ item in array ] "string with automatic variable expansion"

backtick is the escape character, mostly found on the key combined with tilde-sign ~

'without automatic variable expansion'

Keyboard shortcuts

Tab: command completion F7: display history popup, Alt-F7: clears command buffer Ctrl + $, Ctrl +*: jump one word left or right

Esc: clear the command line F8: lookup last command that starts with current input. Try this: Get-Process; <enter>; Get-F8> | More: <Ctrl-C> quit, <q> quit, <space> scroll page, <enter> scroll one line

Use arrow up and down to browse previous commands Home, End: jump to start or end of current command line

Within ISE: F5 = Run, F8 = Run Selection

Security

The .ps1 extension Execution Policy (Set- and Get-ExecutionPolicy) To prevent command hijacking

Associated with Notepad. When a user receives a PowerShell script through e-mail and double-clicks it then the script just opens in notepad instead of executing (like the i-love-you virus did).

Restricted (default), AllSigned, RemoteSigned, Unrestricted (not recommended)
Remote scripts: not on local fixed disks, like CD'S/DVD's, drive mappings to network shares, attachments in e-mail and chat-programs.

Variables

$_ # Current object in the pipeline
$host # Full path to the user's home directory
$PSHome # Full path to the installation directory

$host # Displays the PowerShell version
$host | $i = 1 # storing value 1 in variable $i
$host | ++$ i # incrementing $i with 1, resulting in 2

Working with files


Import-Csv p.csv # displays using Format-List because there are more than four columns and object type is not recognized
Import-ComXml.pxml # displays using Format-Table because object type is recognized (try to add: | Get-Member)

Compare-Object (Import-ComXml.pxml) | (Get-Process) -Property name # compare processes stored in XML with current situation

Dir -Recurse | Where {$_ -length -gt 100MB} | Group Length | Where {$_ count -gt 1} # displays large files with exact same size, might be duplicate

This document is licensed under a Creative Commons Attribution-NonCommercial-NoDerivsCC BY-NC-ND license.
What’s New in PowerShell 3

Requirements: W2008 R2 SP1, W7 SP1, .NET 4.0, download and install KB2506146 or KB2506143.

Note: install ISE first before upgrading to PowerShell 3 on Windows 2008 (R2) or Windows 7!

Automatic module loading: no need to use e.g. "Import-Module ActiveDirectory" anymore.

ISE changes: IntelliSense, brace matching, error indication, start snippets, rich copy, block select (Alt+Mouse), context sensitive help (F1), many more!

New modules: BranchCache, DirectAccess, Dism, DHCP, DNS, iSCSI, NetAdapter, NetTCP/IP, NetworkSecurity, PKI, Printing, Scheduled Tasks, SMB, Storage, many more!

SPItem # same function as $.: current item in pipeline

Show-Command # shows a GUI for a specific Cmdlet

Get-Process | Out-GridView -OutputMode Multiple | Stop-Process # Out-GridView has an OutputMode parameter: select several items and press OK


Install-PowerShellWebAccess -UseTestCertificate # creates the virtual directory and application pool using a test certificate

Add-PwaAuthorizationRule * * # Creates an authorization rule. Now browse with any supported browser (IE, Chrome, Firefox, Safari!) to http://<server>/pswa

Loopying

<table>
<thead>
<tr>
<th>While loop only executes when condition is true</th>
<th>Do ... While loop, always executes, at least once</th>
<th>Do ... Until loop, always executes, at least once</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i = 1</td>
<td>$a = 1</td>
<td>$a = 1</td>
</tr>
<tr>
<td>While ($i le 10) {$i++}</td>
<td>Do ($a; $a++) While ($a le 10)</td>
<td>Do ($a; $a++) Until ($a le 10)</td>
</tr>
</tbody>
</table>

Typical example of a Do ... Until loop

$RequiredLength = 12

Do {
    $password = read-host -prompt "Password, please"
    if ($password length $t (RequiredLength) ("password is too short!") )
    Until (Password length $t $RequiredLength)
}

Functions


WMI

Get-WmiObject -list # lists all WMI classes

Get-WmiObject Win32_Share $Share = Get-WmiObject Win32_Share | Where { $_.Name -eq "CS" } # inspects shares through WMI

$Share = Get-WmiObject Win32_Share $id = Get-WmiObject Win32_Share | Where { $_.Name eq "\CS" } # displays all WMI classes

$Share.create("\CS", "mynewshare", 0) # creating a new share

Get-WmiObject Win32_OperatingSystem -computername (Get-Content servers.txt) | Format-Table __SERVER Version,ServicePackMajorVersion,ServicePackMinorVersion # displays the default action when an error occurs

Get-WmiObject Win32_LogicalDisk -filter DriveType=3 -computername (Get-Content computertos.txt) | Format-Table __SERVER,DeviceID,FreeSpace @([Label = 'PercentFree']Expression = $_.FreeSpace / $_.Size,FormatString = '{0:#0.00%}')

Active Directory

Requirements: PowerShell v2, Active Directory Module for Windows PowerShell (on a Domain Controller, also part of RSAT). Open port TCP/9389.


Get-Module ActiveDirectory # imports the Active Directory module for PowerShell

Get-Command -module ActiveDirectory # displays all 76 commands in PowerShell

Get-ADOrganizationalUnit "Employees" -Path "DC=Contoso,DC=com" # creates a new OU

Get-ADOrganizationalUnit -Filter "*" | FT Name, DistinguishedName,AutoSize # or 'Continue' or 'Inquire' or 'Stop'

New-ADUser Testuser # creates a disabled user in the Users container

# The next script takes a plain text password as input and creates an enabled user account in the Employees OU

SUserpwd = ConvertTo-SecureString -AsPlainText "PSw@Word" -Force # converts plaintext to secure string

New-ADUser Testuser -AccountPassword SUserpwd -Enabled $true -Path OU=Employees,DC=Contoso,DC=com'

For ($i = 1; $i le 10; $i++) { New-ADUser --name Testuser$i } # creates ten new testers

Background Jobs

Start-Job -Get-Process PowerShell

Get-Job

Get-Job -id 1 Receive-Job # use the -keep parameter to keep the data in memory

Start-Job -Sleep 60 # it starts a new job which just waits for 60 seconds

Wait-Job -id 3 # wait for a job to complete, or use: Stop-Job -id 3 # stops job

Remove-Job -id 3 # remove a completed job

Error handling and Debugging

SerrorActionPreference # displays the default action when an error occurs

Dir c. x. c. # should result in a file listing of the c-drive, followed by an error, followed by a listing of the c-drive

SerrorActionPreference = "SilentlyContinue" # or 'Continue' or 'Inquire' or 'Stop'

Dir c. x. c. # should result in two file listings of the c-drive, no more error

# Use the –ErrorAction parameter to use a other error action for the current command only

SerrorAction = Get-Member # displays information about the last error

More information on the Internet (and the previous page...)

http://blogs.msdn.com/b/powershell/
http://poshollic.com/
http://thepowershellguy.com/
http://www.powershellmagazine.com/

http://www.computerperformance.co.uk/powershell/
http://powerwf.com/products.aspx