Building Regular Expression Commands

Jeff Hicks
AUTHOR/TEACHER/SENSEI

@jeffhicks | https://jdhitsolutions.com

Testing a regular expression pattern results in True or False

Operators

-Like

-NotLike

-Match

-NotMatch

```
PS C:\> "Pluralsight" -like "plu*"
True
```

-Like

Uses wildcards

Not case-sensitive by default

```
PS C:\> "Pluralsight" -like "plu*"
True
PS C:\> "PowerShell" -like "p*shell"
True
```

-Like

Uses wildcards

Not case-sensitive by default

```
PS C:\> "Pluralsight" -like "plu*"
True
PS C:\> "PowerShell" -like "p*shell"
True
PS C:\> "PowerShell" -like "bash*shell"
False
```

-Like

Uses wildcards

Not case-sensitive by default

```
PS C:\> "PowerShell" -notlike "bash"
True
```

-NotLike

Tests the opposite

Also not case-sensitive

```
PS C:\> "PowerShell" –notlike "bash"
True
PS C:\> "apple tree" -notlike "app*"
False
```

-NotLike

Tests the opposite

Also not case-sensitive

```
PS C:\> get-service -name win*

Status   Name               DisplayName
------   ----               -----------
Running  WinDefend          Windows Defender Antivirus Service
Running  WinHttpAutoProx... WinHTTP Web Proxy Auto-Discovery Se...
Running  Winmgmt            Windows Management Instrumentation
Running  WinRM              Windows Remote Management (WS-Manag...
```

Wild Cards are Everywhere in PowerShell

You don’t always have to use –Like

Many parameters accepts wildcards

Don’t always believe the help – Try it!

Use Filtering for Everything Else

```
PS C:\> get-content C:\scripts\servers.txt | where {$_ -like "*core*"}
chi-core01
chi-core02
```

Use Filtering for Everything Else

```
PS C:\> get-process | where-object {$_.company –AND $_.company -notlike
"Mic*"} | Group-Object -property Company -NoElement |
Sort-Object -Property Count -Descending | Select-Object -First 5

Count Name
----- ----
    8 Lenovo Group Ltd.
    8 Mozilla Corporation
    8 Intel Corporation
    5 Dropbox, Inc.
    4 TechSmith Corporation
```

```
PS C:\> "PowerShell" -match "\w+"
True
```

-Match

Compare to a regular expression pattern

Not case sensitive by default

```
PS C:\> $matches
Name                           Value
----                           -----
0                              PowerShell
```

-Match

Creates a $Matches object

Matching Regular Expression Patterns

```
PS C:\> "10.11.12" -match "\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}"
False
PS C:\> "10.11.121.130" -match "\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}"
True
PS C:\> $matches
Name                           Value
----                           -----
0                              10.11.121.130
PS C:\> $matches.0
10.11.121.130
```

```
PS C:\> "10.11.12" -notmatch "\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}"
True
```

-NotMatch

Test if a regular expression does not match

Often easier than creating a negative regular expression pattern

Always test for failures

Case Sensitivity

Force a case-sensitive comparison

$Matches is not populated

-cLike

-cNotLike

-cMatch

-cNotMatch

Case Matching

```
PS C:\> "jeff" -clike "Je*"
False
PS C:\> "jeff" -cnotlike "JE*"
True
PS C:\> "PowerShell 5.1" -cMatch "P*shell"
False
PS C:\> "PowerShell 5.1" -cnotMatch "P*shell"
True
```

Pattern Matching Drift

```
PS C:\> "jeff@company.com" -match "\w+@\w*\.com"
True
PS C:\> $matches.0
jeff@company.com
PS C:\> "jeff@company.comm" -match "\w+@\w*\.com"
True
PS C:\> $matches.0
jeff@company.com
PS C:\> "jeff-foo@company.comm" -match "\w+@\w*\.com"
True
PS C:\> $matches.0
foo@company.com
```

Anchor Your Match

^

Anchor at the beginning

“The string must start with...”

$

Anchor at the end

“The string must end with...”

Avoiding Pattern Matching Drift

```
PS C:\> "jeff-foo@company.comm" -match "^\w+@\w*\.com"
False
PS C:\> "jeff@company.comm" -match "\w+@\w*\.com$"
False
PS C:\> "jeff@company.com" -match "^\w+@\w*\.com$"
True
PS C:\> "jeff-foo@company.com" -match "^\w+@\w*\.com$"
False
```

You decide how much anchoring you need when matching regular expression patterns

ValidatePattern

Validate script and function parameters

Use a regular expression pattern to validate

[ValidatePattern("<pattern>")]

Match is implied

Validating Parameters

```
Function Test-CompanyIP {
[cmdletbinding()]

Param(
[Parameter(Mandatory)]
[ValidatePattern("^10\.\d{1,3}\.\d{1,3}\.\d{1,3}$")]
[string]$IPAddress
)

Write-Verbose "Testing $IPAddress"
#Your code here

}
```

Successful Validation

```
PS C:\> Test-CompanyIP -IPAddress 10.100.1.2 -Verbose
VERBOSE: Testing 10.100.1.2
...
```

Failed Validation

```
PS C:\> Test-CompanyIP -IPAddress 11.100.1.2 -Verbose

Test-CompanyIP : Cannot validate argument on parameter 'IPAddress'. The
argument "11.100.1.2" does not match the "^10\.\d{1,3}\.\d{1,3}\.\d{1,3}$"
pattern. Supply an argument that matches "^10\.\d{1,3}\.\d{1,3}\.\d{1,3}$"
and try the command again.

At line:1 char:27

+ Test-CompanyIP -IPAddress 11.100.1.2 -Verbose

+                           ~~~~~~~~~~

    + CategoryInfo          : InvalidData: (:) [Test-CompanyIP],
ParameterBindingValidationException

    + FullyQualifiedErrorId : ParameterArgumentValidationError,Test-
CompanyIP
```

Parameter Validation Tips

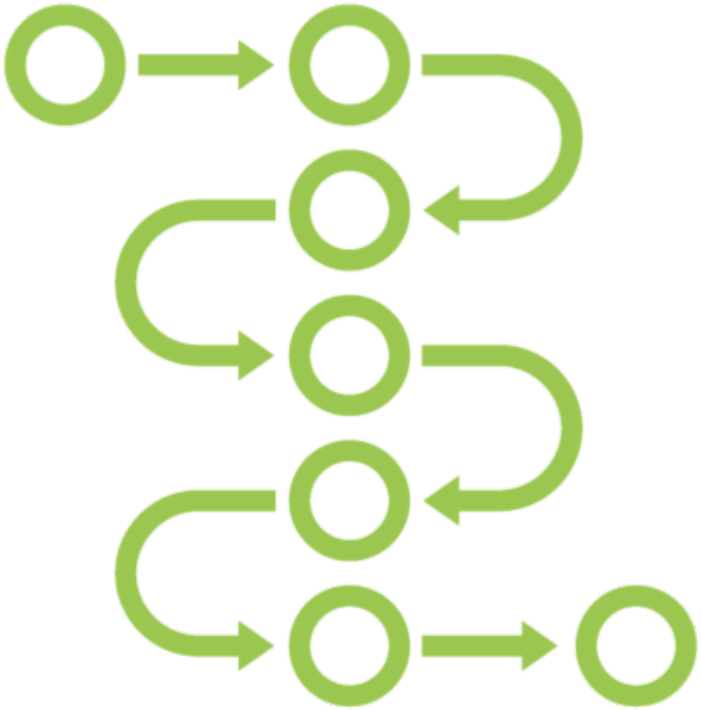
Use anchors to help trim out extra characters and white space

Test with known good values

Test with values that should fail

Document the parameter

Switch -Regex



Base code execution on a regular expression match

Not case-sensitive by default

Multiple matches are possible

Help about_switch

Switch -Regex

```
switch -Regex ($string) {

    "<pattern>" {
        #code
    }
    "<pattern>" {
        #code
    }
    Default {
        #code for no matches - optional
    }
}
```

Switch -Regex

```
Switch -Regex ($servername) {
    "^LON" {Write-Host "Setting up for London" -fore green}
    "^SFO" {Write-Host "Setting up for San Francisco" -fore green}
    "-MKT-" {Write-Host "Configuring Marketing" -fore cyan}
    "-MFG-" {Write-Host "Configuring Manufacturing" -fore cyan}
    "DC\d+$" {Write-Host "Installing as a domain controller" -fore Magenta}
    "FP\d+$" {Write-Host "Installing as a file&print server" -fore Magenta}
     Default { Write-Warning "Failed to process $servername"}
}
```

Switch -Regex

```
PS C:\scripts\> .\serverbuild.ps1
Processing LON-MKT-DC03
Setting up for London
Configuring Marketing
Installing as a domain controller
Processing MIA-MFG-FP23
Configuring Manufacturing
Installing as a file&print server
Processing SFO-SLS-WB11
Setting up for San Francisco
Processing SFO-MKT-FP24
Setting up for San Francisco
Configuring Marketing
Installing as a file&print server
Processing CHI-EXE-WB234
WARNING: Failed to process CHI-EXE-WB234
```

Select-String

Find matching lines of text

Similar to *grep*

Uses a pattern

- Could be a simple string match (* implied)
- Could be a regular expression pattern

Can be case-sensitive

Can find non-matches

Select-String

```
PS C:\> get-content .\process-list.txt | select-string -Pattern "Power*"

    412       22     5876       3512       6.42   9072   1 PowerMgr
   2072       82   157752     251920     295.72  18088   1 POWERPNT
   1286       73   365204     375304      36.73   8440   1 powershell
    907       51   152972     125876      12.44  13116   1 powershell
   1190       73   248440     293140      22.98  19272   1 powershell_ise
```

Select-String

Searching Files

```
PS C:\> dir trans* | select-string "Set-Service"

trans-2.txt:48:PS C:\work> get-service xb* | set-service -StartupType
Disabled -PassThru | select name,starttype
trans-2.txt:726:Cmdlet          Set-Service  3.1.0.0
Microsoft.PowerShell.Mana...
trans-3.txt:92:PS C:\work> help set-service
```

Select-String

Searching Files

```
PS C:\> dir trans* | select-string "Set-Service" –list

trans-2.txt:48:PS C:\work> get-service xb* | set-service -StartupType
Disabled -PassThru | select name,starttype
trans-3.txt:92:PS C:\work> help set-service

PS C:\> dir trans* | select-string "Set-Service" | select-object
Path,LineNumber

Path                 LineNumber
----                 ----------
C:\trans-2.txt               48
C:\trans-2.txt              726
C:\trans-3.txt               92
```

Select-String

Searching Files

```
PS C:\> dir trans* | select-string "[Ss]et-\w+\s+(-)?[a-zA-Z]+.*"

trans-2.txt:48:PS C:\work> get-service xb* | set-service -StartupType
Disabled -PassThru | select name,starttype
trans-2.txt:816:PS C:\work> Set-ExecutionPolicy RemoteSigned -force
trans-3.txt:58:PS C:\work> Set-VMSwitch Labnet -Notes "Test network
virtual switch using NAT"
trans-3.txt:137:PS C:\work> Set-PSReadLineOption -HistoryNoDuplicates
$True -MaximumHistoryCount 500
trans-3.txt:141:+ Set-PSReadLineOption -HistoryNoDuplicates $True -
MaximumHistoryCount  ...
trans-3.txt:147:+ Set-PSReadLineOption -HistoryNoDuplicates $True -
MaximumHistoryCount  ...
trans-3.txt:152:PS C:\work> Set-PSReadLineOption -HistoryNoDuplicates -
MaximumHistoryCount 500
```

Select-String

Searching Files

```
PS C:\> $lines = (dir trans* | select-string "[Ss]et-\w+\s+(-)?[a-zA-
Z]+.*").line
PS C:\> $lines | select-string "set-\w+" |
select-object -expandproperty matches |
select-object -property Value -Unique

Value
-----
set-service
Set-ExecutionPolicy
Set-VMSwitch
Set-PSReadLineOption

PS C:\> ($lines | select-string "set-\w+").matches |
select-object -property Value -Unique
```

Select-String

Context

```
 PS C:\> dir trans* | select-string set-service -Context 1,1

 C:\> trans-2.txt:48:
>C:\> trans-2.txt:49:PS C:\work> get-service xb* | set-service -
StartupType Disabled -PassThru | select name,starttype
 C:\> trans-2.txt:50:
 C:\> trans-2.txt:726:Cmdlet   Set-SecureBootUEFI    2.0.0.0    SecureBoot
>C:\> trans-2.txt:727:Cmdlet   Set-Service           3.1.0.0
Microsoft.Powe...
 C:\> trans-2.txt:728:Cmdlet   Set-SPOBrowserIdleSignOut 16.0.89...
Microsoft...
 C:\> trans-3.txt:92:
>C:\> trans-3.txt:93:PS C:\work> help set-service
 C:\> trans-3.txt:94:
```

Select-String

Context

```
PS C:\> dir trans*.txt | Select-String Set-Service -Context 0,4 |
Sort Filename |  Select-Object Filename,LineNumber,
@{Name="ContextResult";Expression = {
$(@"
$(($_.context.precontext | Out-String).trim())
$($_.line)
$(($_.context.postcontext | Out-String).trim())
"@).trim() }} | Format-List
```

```
Filename      : trans-2.txt
LineNumber    : 49
ContextResult : PS C:\work> get-service xb* | set-service -StartupType Disabled -PassThru | select name,starttype
                Name           StartType
                ----           ---------
                XblAuthManager  Disabled

Filename      : trans-2.txt
LineNumber    : 727
ContextResult : Cmdlet          Set-Service                                  3.1.0.0    Microsoft.PowerShell.Management
                Cmdlet          Set-SPOBrowserIdleSignOut                    16.0.89... Microsoft.Online.SharePoint.PowerShell
                Cmdlet          Set-SPOBuiltInDesignPackageVisibility        16.0.89... Microsoft.Online.SharePoint.PowerShell
                Cmdlet          Set-SPOGeoStorageQuota                       16.0.89... Microsoft.Online.SharePoint.PowerShell
                Cmdlet          Set-SPOHideDefaultThemes                     16.0.89... Microsoft.Online.SharePoint.PowerShell

Filename      : trans-3.txt
LineNumber    : 93
ContextResult : PS C:\work> help set-service
                PS C:\work> Get-PSReadLineOption
```

Show-Context.ps1

```
...
if ($ContextOnly) {
    $grouped = $Data | Group-Object -Property Filename
    foreach ($file in $grouped) {
        Write-Host "`n$($file.name)" -ForegroundColor cyan
        Write-Host $("-" * $($file.name.length)) -ForegroundColor Cyan
        foreach ($item in $file.group) {
          $l = $item.linenumber
            if ($item.context.precontext) {
                foreach ($pre in $item.context.precontext) {
                    $l--
                    $leader = $("[$l]").Padright(6, ' ')
                    Write-Host "$leader $($pre.trim())" -ForegroundColor yellow
                }
            }
            $leader = $("[$($item.linenumber)]").Padright(6, ' ')
            Write-Host "$leader $($item.line.trim())" -ForegroundColor green

            if ($item.context.postcontext) {
             foreach ($post in $item.context.postcontext) {
                    $l++
                    $leader = $("[$l]").Padright(6, ' ')
                    Write-Host "$leader $($post.trim())" -ForegroundColor yellow
                }
            }
        } #foreach item
    } #foreach file
} #if $contextonly
...
```

Select-String

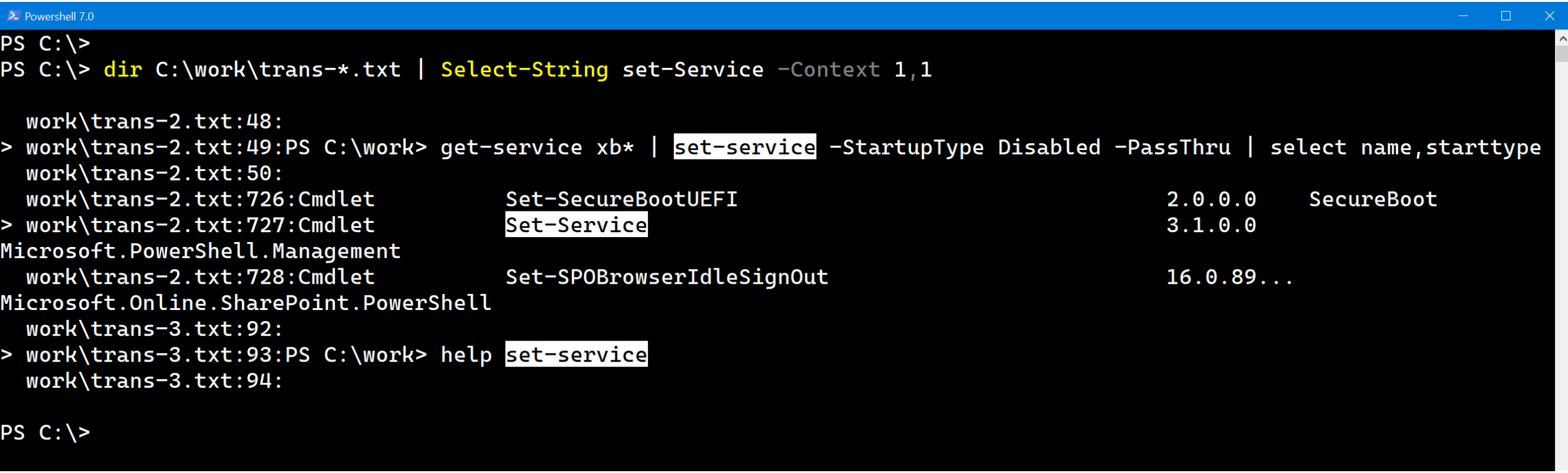
Context

```
PS C:\> Get-ChildItem trans*.txt | Select-String set-service -context 0,2
| Show-Context -ContextOnly
```

```
trans-2.txt
-----------
[49]   PS C:\work> get-service xb* | set-service -StartupType Disabled -PassThru | select name,starttype
[50]
[51]   Name           StartType
[727]  Cmdlet          Set-Service                                3.1.0.0    Microsoft.PowerShell.Management
[728]  Cmdlet          Set-SPOBrowserIdleSignOut                  16.0.89... Microsoft.Online.SharePoint.PowerShell
[729]  Cmdlet          Set-SPOBuiltInDesignPackageVisibility      16.0.89... Microsoft.Online.SharePoint.PowerShell

trans-3.txt
-----------
[93]   PS C:\work> help set-service
[94]
[95]   PS C:\work> Get-PSReadLineOption
```

PowerShell 7 Bonus!



Summary

PowerShell operators evaluate as $True or $False

Use –Like with wildcards

Use –Match with regular expression patterns

Regular Expressions makes a useful parameter validation technique

Switch –Regex can add flexibility to your code

Select-String is a handy tool for searching across text files