PowerShell tradycyjnie

http://aplitt.zsl.gda.pl/Powershell/Ściąga%20Paula-Januszkiewicz-Powershell-reference-PL.pdf

PowerShell Obiekty a pipe

Obiekty są fajne ułatwiają życie. Dla przypomnienia że są fajne:

```
Get-process |out-gridview
```

```
Get-History | ogv -PassThru  | Invoke-Expression
```

```
Add-Type -AssemblyName System.speech
$speak = New-Object System.Speech.Synthesis.SpeechSynthesizer
$speak.Speak('Hello...')
```

O „|” i o obiektach

Get-Service|more {Jakie są nagłówki kolumn}

Get-Service|Select-Object ServicesDependedOn, service {tabulatorem uzupełnijcie Propoeryty}

Get-Service|gm {wyświetli wam informacje o obiekcie}

Obiekt może (i składa się) z podobiektów, więc tak naprawdę często jest to struktura zagnieżdżona.

No dobra skoro wszystko co jest w PS jest obiektem, i jest to struktura, to co tak naprawdę podróżuje przez „|”?.

Teraz jednak interesuje nas dla Get-Service|gm:

TypeName: System.ServiceProcess.**ServiceController**

O „|” i o obiektach

No to robimy tak:

get-help stop-service -full, i szukamy:

-InputObject **<ServiceController[]>**

Specifies ServiceController objects that represent the services to stop. Enter a variable that contains the objects, or type a command or expression that gets the objects.

Required? true
Position? 0
Default value None
Accept pipeline input? True (ByValue)
Accept wildcard characters? false

O „|” i o obiektach

To jest jeden z możliwych sposobów przesyłania za pośrednictwem „|”, - ByValue.

To co podróżuje za pomocą „|” musi do siebie pasować, dlatego:

get-service|stop-service -**whatif**

Zadziała (jak nie to nie pracujecie jako Admini).

Druga opcja ByPropertyName:

Get-Process Calculator|dir

Wskaże lokalizację pliku. Opcja ByValue jest wydajniejsza.

O „|” i o obiektach

To co podróżuje za pomocą „|” musi do siebie pasować, ale można to modyfikować:

```
Get-ADComputer -Filter *| select -Property name, @{name = 'ComputerName'; expression={$_.name}}

name        ComputerName
----        ------------
test        test
```

Get-ADComputer -Filter *| select –Property **@{name = 'ComputerName'; expression={$_.name}}**|get-service –name bits

Cmdlet Get-ADComputer jest częścią modułu ActiveDirectory. By móc go uruchomić:

- Komputer musi być w domenie
- Jeżeli nie jest to Kontroler domeny to odinstalowujemy pakiet:
  WindowsTH-RSAT_WS_1709-{wersja os’a}.msu.
- Następnie robimy w powershellu Import-Module ActiveDirectory

O „|” i o obiektach

Ok pipe jest fajny, ale co wtedy kiedy nie działa?:

Get-WmiObject -class win32_bios

Get-help Get-WmiObject –showwindow ***Uwaga: brak input'a, i [-ComputerName <String[]>]***

Get-WmiObject -class win32_bios –ComputerName test

Get-ADComputer -Filter *| select -Property name |gm

Get-ADComputer -Filter *| select -ExpandProperty name {|gm -> string}

Czyli:

Get-WmiObject -class win32_bios –ComputerName (Get-ADComputer -Filter *| select -ExpandProperty name)

O „|” i o obiektach

Tu:

Get-WmiObject -class win32_bios –ComputerName (Get-ADComputer -Filter *| select -ExpandProperty name)

Najpierw zostanie wykonane polecenie:

(Get-ADComputer -Filter *| select -ExpandProperty name)

Co spowoduje przygotowanie listy nazw komputerów w formacie akceptowalnym dla „wyższego” polecenia:

Get-WmiObject -class win32_bios –ComputerName **[string]**

Które oczekuje string(ów).

O „|” i o obiektach