Dlaczego potok działa – info

Jak PowerShell łączy ze sobą cmd-lety w potoku

Jak działa przekazywanie parametrów by name

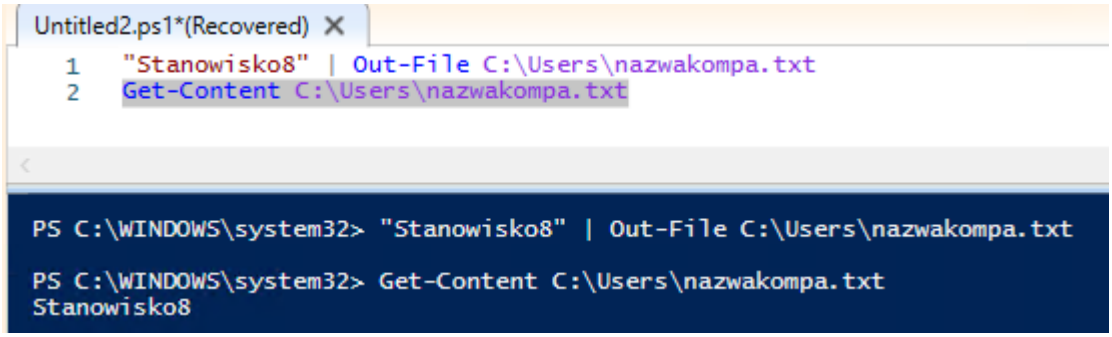
Jak działa przekazywanie parametrów by value

Rozwiazywanie problemów z potokami

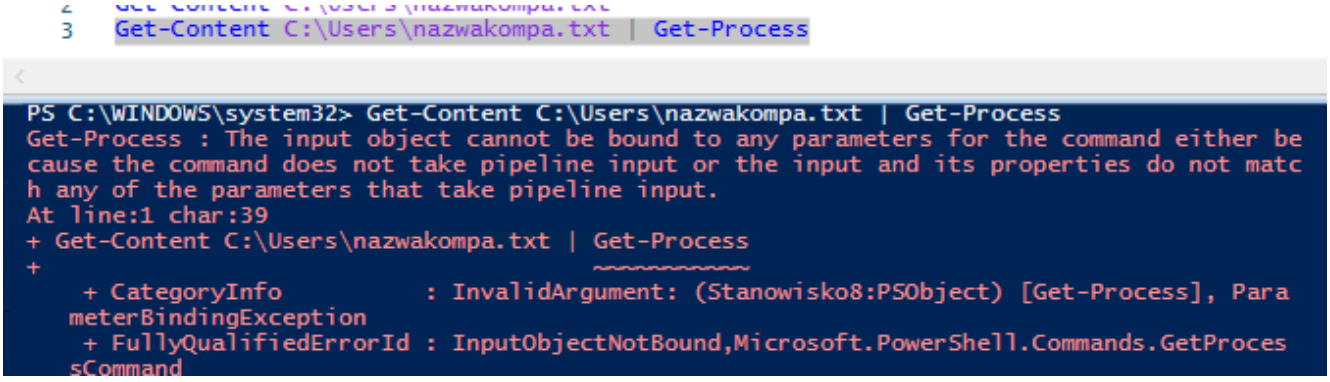
Jak działa potok

Wysyłam tekst Stanowisko8 do pliku C:\Users\nazwakompa.txt

Wyświetlam zawartość pliku C:\Users\nazwakompa.txt



Chcemy wyświetlić listę procesów na komputerze z pliku

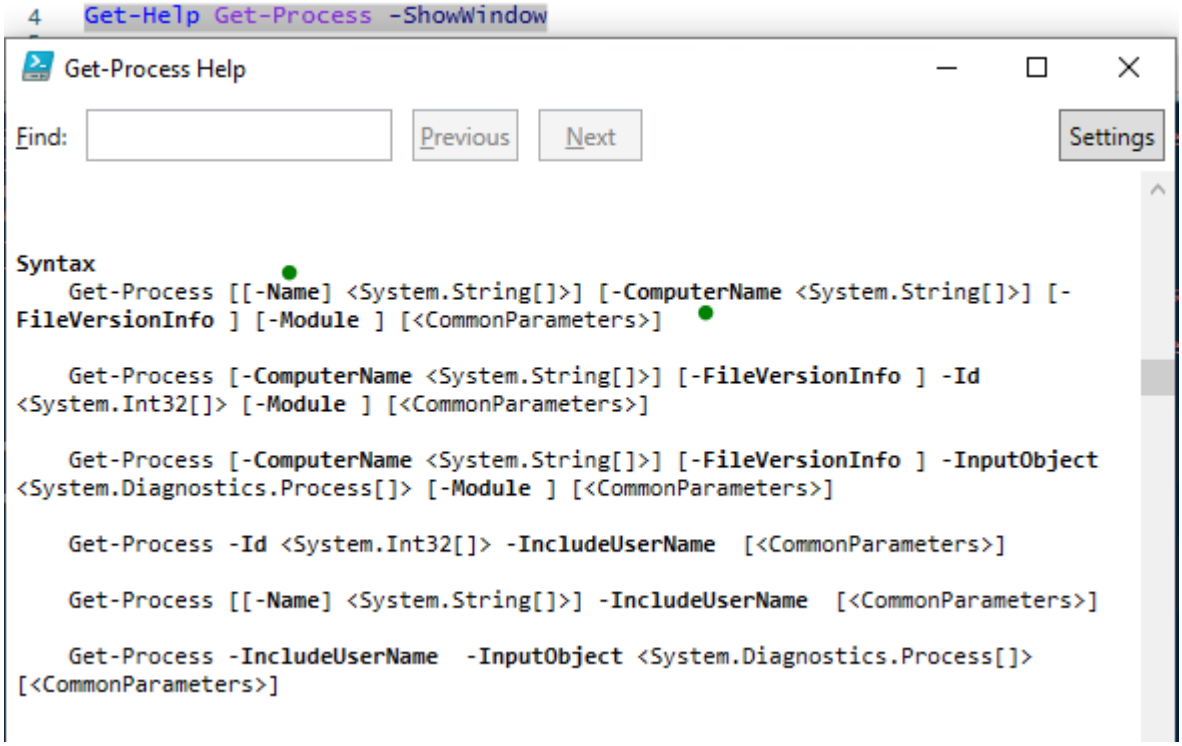


Nie działa

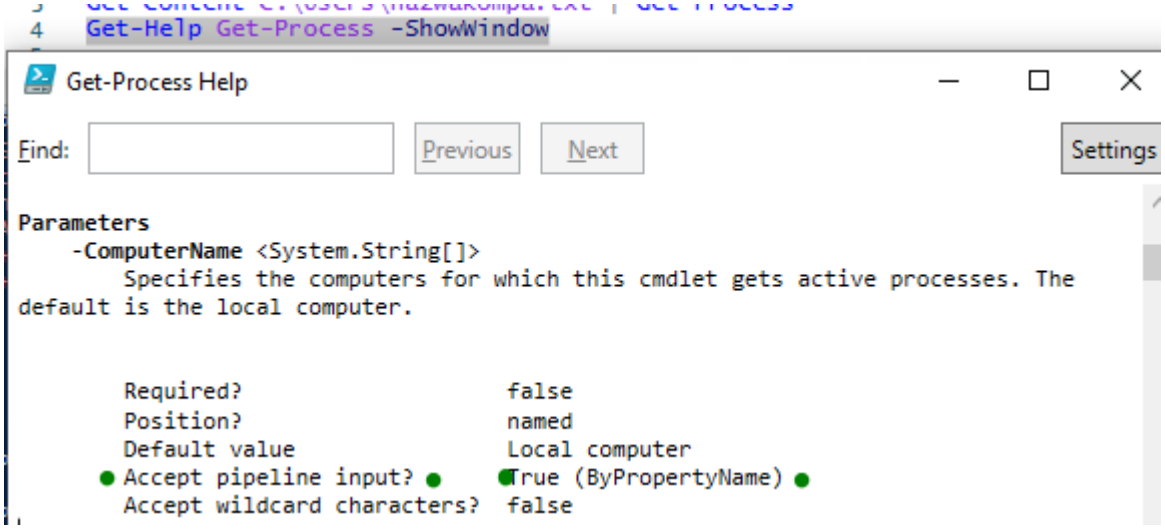
Parametry wej nie mogą być powiązane z parametrami komendy

Dlaczego Get-Process ma wiedzie co jest w pliku przecież zamiast nazwy komputera może znajdować się tam cokolwiek, nazwy użytkowników lub procesów.

Zacznijmy od helpa



Get-Process potrafi przyjąć parametr z potoku, ale przez nazwę właściwości przechodzącego potokiem obiektu



Należy zadbać o to, aby obiekt wczytany z pliku przez Get-Content miał nazwę ComputerName wtedy Get-Process domyśli się, że chodzi o szukanie procesów na komputerze o takiej nazwie

```
5   Get-Content C:\Users\nazwakompa.txt | Select-Object @{n="ComputerName";e={$_}} | Get-Process
```

```
PS C:\WINDOWS\system32> Get-Help Get-Process -ShowWindow

PS C:\WINDOWS\system32> Get-Content C:\Users\nazwakompa.txt | Select-Object @{n="ComputerName";e={$_}} | G
Get-Process : Nie można połączyć się z komputerem zdalnym.
At line:1 char:82
+ ... zwakompa.txt | Select-Object @{n="ComputerName";e={$_}} | Get-Process
+                                                              ~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [Get-Process], InvalidOperationException
    + FullyQualifiedErrorId : System.InvalidOperationException,Microsoft.PowerShell.Commands.GetProces
   sCommand
```

Nie działa, ale tak właściwie to nie zgadza się nazwa komputera.

Należy sprawdzać nazwę komputera i wysłać do pliku

Pobranie nazwy komputera i innych zmiennych systemowych

Sposób na-wprost:

$(Get-WmiObject Win32_Computersystem).name

Jeśli jednak chcemy ograniczyć ilość znaków, z powodzeniem możemy skorzystać ze zmiennej systemowej:

$env:computername

Mamy proste polecenie

$env:computername | Out-File C:\Users\nazwakompa.txt

```
6   $env:computername |  Out-File C:\Users\nazwakompa.txt
```

```
PS C:\WINDOWS\system32> $env:computername |  Out-File C:\Users\nazwakompa.txt

PS C:\WINDOWS\system32>
```

Ponownie wykonujemy właściwą linie polecenie

```
5   Get-Content C:\Users\nazwakompa.txt | Select-Object @{n="ComputerName";e={$_}}| Get-Process
6   $env:computername |  Out-File C:\Users\nazwakompa.txt
```

```
PS C:\Windows\system32> $env:computername |  Out-File C:\Users\nazwakompa.txt

PS C:\Windows\system32> Get-Content C:\Users\nazwakompa.txt | Select-Object @{n="ComputerName";e={$_}}| Get-Process

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
    141       9     1836       8580       0,13    4512   0 AggregatorHost
    394      19     5476       2080       0,16    6864   2 backgroundTaskHost
    510      21     1904       6036       0,56     652   0 csrss
    156      11     1664       5512       0,09     736   1 csrss
    441      19     1952       6556       2,73    3644   2 csrss
    489      18     4932      24532       6,38    5156   2 ctfmon
```

Można zrobić inaczej, używając Get-Process -ComputerName i wczytujemy nazwę z pliku (Get-Content C:\Users\nazwakompa.txt)

```
7   Get-Process -ComputerName (Get-Content C:\Users\nazwakompa.txt)
```

```
   359      20     4944      20196       0,19   4320   0 svchost
   164       9     1720       6560       0,05   4412   0 svchost
```

Inna metoda konkretne procesy przekazujemy do pliku

```
8   "winlogon","taskhost" | Out-File C:\Users\procesy.txt
```

```
PS C:\WINDOWS\system32> "winlogon","taskhost" | Out-File C:\Users\procesy.txt

PS C:\WINDOWS\system32>
```

Wyświetlenie pliku

```
9   Get-Content C:\Users\procesy.txt
```

```
PS C:\WINDOWS\system32> Get-Content C:\Users\procesy.txt
winlogon
taskhost
```

Znaki z pliku przesyłamy na Get-Process i błąd mamy

```
10  Get-Content C:\Users\procesy.txt | Get-Process
```

```
PS C:\WINDOWS\system32> Get-Content C:\Users\procesy.txt | Get-Process
Get-Process : The input object cannot be bound to any parameters for the c
mand does not take pipeline input or the input and its properties do not m
```

Zaglądam do helpa

```
11  Get-Help Get-Process -ShowWindow
```

Get-Process Help

Find: Previous Next

```
    -Name <System.String[]>
        Specifies one or more processes by process name. You can ty
process names (separated by commas) and use wildcard characters. Th
("Name") is optional.

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

Get-Content C:\Users\procesy.txt | Select-Object @{n="Name";e={$_}} | Get-Process

```
12  Get-Content C:\Users\procesy.txt | Select-Object @{n="Name";e={$_}} | Get-Process
```

```
Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
    221      12     2396      10400       0,23     816   1 winlogon
    262      11     2364      10140       0,22    4840   2 winlogon
Get-Process : Cannot find a process with the name "taskhost". Verify the process name and call the cmdlet again.
At line:1 char:71
+ ... C:\Users\procesy.txt | Select-Object @{n="Name";e={$_}} | Get-Process
+                                                              ~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (taskhost:String) [Get-Process], ProcessCommandException
    + FullyQualifiedErrorId : NoProcessFoundForGivenName,Microsoft.PowerShell.Commands.GetProcessCommand
```

Błąd występuje, ponieważ Get-Process nie może znaleźć procesu o nazwie taskhost. Może to być spowodowane kilkoma przyczynami, np. proces o tej nazwie nie jest uruchomiony w systemie w momencie wykonywania polecenia.

Inna metoda

Get-Process -Name (Get-Content C:\Users\procesy.txt)

```
13  Get-Process -Name (Get-Content C:\Users\procesy.txt)
```

```
PS C:\Windows\system32> Get-Process -Name (Get-Content C:\Users\procesy.txt)
Get-Process : Cannot find a process with the name "taskhost". Verify the process name and call the cmdlet again.
At line:1 char:1
+ Get-Process -Name (Get-Content C:\Users\procesy.txt)
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (taskhost:String) [Get-Process], ProcessCommandException
    + FullyQualifiedErrorId : NoProcessFoundForGivenName,Microsoft.PowerShell.Commands.GetProcessCommand


Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
    221      12     2320      10388       0,23     816   1 winlogon
    262      11     2288      10124       0,22    4840   2 winlogon
```

Aby nie było błędu należy użyć skrypt z filtrowaniem (ale to inny temat)

Skrypt, najpierw filtruje procesy, a następnie pobiera tylko te, które są uruchomione:

```
$processNames = Get-Content C:\Users\procesy.txt

$runningProcesses = Get-Process | Select-Object -ExpandProperty Name


foreach ($name in $processNames) {
    if ($runningProcesses -contains $name) {
        Get-Process -Name $name
    }
    else {
        Write-Host "Process $name not found."
    }
}
```

```
PS C:\Windows\system32> $processNames = Get-Content C:\Users\procesy.txt
$runningProcesses = Get-Process | Select-Object -ExpandProperty Name

foreach ($name in $processNames) {
    if ($runningProcesses -contains $name) {
        Get-Process -Name $name
    }
    else {
        Write-Host "Process $name not found."
    }
}


Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
    221      12     2320      10388       0,23     816   1 winlogon
    262      11     2288      10124       0,22    4840   2 winlogon
Process taskhost not found.
```

W ten sposób upewnisz się, że skrypt nie przerywa działania w przypadku, gdy jakiś proces nie jest uruchomiony, i wyświetli informację o braku tego procesu.

Kolejne przykłady będą dotyczyły próby zatrzymania procesu

Proces ten to notepad

```
15  Get-Process -Name notepad
```

```
PS C:\WINDOWS\system32> Get-Process -Name notepad

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
    248      13     2840      13948       0,08   2088   2 notepad
    272      15     3020      17224       0,34   4740   2 notepad
```

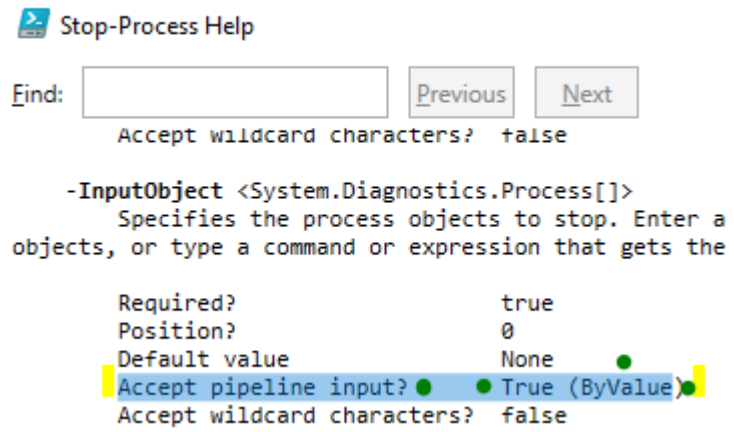
Lista właściwości i metod

```
16  Get-Process -Name notepad | Get-Member
```

```
PS C:\WINDOWS\system32> Get-Process -Name notepad | Get-Member


   TypeName: System.Diagnostics.Process
```

```
17   Get-Help Stop-Process -ShowWindow
```



Może przyjmować z potoku, ale teraz przez wartość (ByValue)

```
18   Get-Process -Name notepad | Stop-Process
```

```
PS C:\WINDOWS\system32> Get-Process -Name notepad | Stop-Process
```

Zmieniamy przykład

Nazwa procesu do zatrzymania wczytywana z pliku tekstowego

```
19   "notepad" | Out-File  C:\Users\zatrzymajprocesy.txt
20   Get-Content C:\Users\zatrzymajprocesy.txt
```

```
PS C:\WINDOWS\system32> "notepad" | Out-File  C:\Users\zatrzymajprocesy.txt

PS C:\WINDOWS\system32> Get-Content C:\Users\zatrzymajprocesy.txt
notepad
```

Uruchamiam notatnik

Wczytamy nazwę z pliku tekstowego, Select-Object tworzy obiek, stop zatrzyma procesy które przyszły w potoku

```
21   notepad
22   Get-Content C:\Users\zatrzymajprocesy.txt | Select-Object @{n="Name";e={$_}} | Stop-Process
```

```
PS C:\WINDOWS\system32> notepad

PS C:\WINDOWS\system32> Get-Content C:\Users\zatrzymajprocesy.txt | Select-Object @{n="Name";e={$_}} | Sto
```

Kolejna metoda

```
24   Stop-Process -Name (Get-Content C:\Users\zatrzymajprocesy.txt)
```

```
PS C:\WINDOWS\system32> Stop-Process -Name (Get-Content C:\Users\zatrzymajprocesy.txt)
```

A co by było, gdyby elementy potoku dawały się połączyć przez wartość i nazwę w takim przypadku mapowanie najpierw odbędzie się przez wartość, gdy to się nie uda to przez nazwę.

Zwrocie uwagę, że jeżeli nie udaje się połączyć komend to sięgamy do pomocy, to daje możliwość samodzielnego rozwiązywania problemów.

Jak PowerShell łaczy ze sobą cmd-let-y w potoku

Jak działa przekazywanie parametrów by name

Jak działa przekazywanie parametrów by value

Rozwiazywanie problemów z potokami