

# PowerShell Pipe

Patrick Gruenauer | Microsoft MVP

Co-Author of „The PowerShell Conference Book“

Author of sid-500.com | <https://sid-500.com>



# Topics

---

- Pipeline
- Formatierung und Ausgabe
  - Format-Table Format-List Format-Wide
  - Import-CSV | Export-CSV
  - ConvertTo-CSV | ConvertFrom-CSV
  - Out-GridView
  - Export-Excel
- \$\_ und \$PSItem
- Filtering
- Sort-Object und Select-Object
- Exkurs: Parameterbindung: ByValue, ByPropertyName

---

# Pipeline

# Piping

---

- Die Piping Technik ermöglicht die „**Verbindung**“ zweier oder mehrerer Cmdlets
- Die Verwendung der Pipe ist eine **Schlüsseltechnologie** in PowerShell
- | → **ALTGR+<>**



```
SID-500.COM | Patrick Gruenauer | MVP PowerShell
PS C:\> Get-Process notepad,mspaint | Stop-Process
PS C:\>
```



```
SID-500.COM | Patrick Gruenauer | MVP PowerShell
PS C:\> Get-SmbOpenFile | Close-SmbOpenFile
PS C:\>
```

# PowerShell Pipe

---

## The PowerShell Pipe



Get-Process notepad,mspaint | Stop-Process

**The pipe takes everything on the left of the pipe and forwards it to the command to the right of the pipe**

sid-500.com

<https://sid-500.com/2018/01/25/powershell-for-beginners-part-7-the-format-commands-and-the-pipe/>

# Piping – Beispiele

```
SID-500.COM | © Patrick Gruenauer | MVP PowerShell [2017-2019]
PS C:\> Get-Service spooler | Stop-Service
PS C:\>
```

```
SID-500.COM | Patrick Gruenauer | MVP PowerShell [2018-2020]
PS C:\> Get-ADUser s.stollane | Set-ADUser -Enabled $false
PS C:\> █
```

```
SID-500.COM | © Patrick Gruenauer | MVP PowerShell [2018-2020]
PS C:\> 'Patrick', 'Patrick', 'Herbert', 'Michael' | Get-Unique
Patrick
Herbert
Michael
PS C:\>
```

# Looking behind the scenes

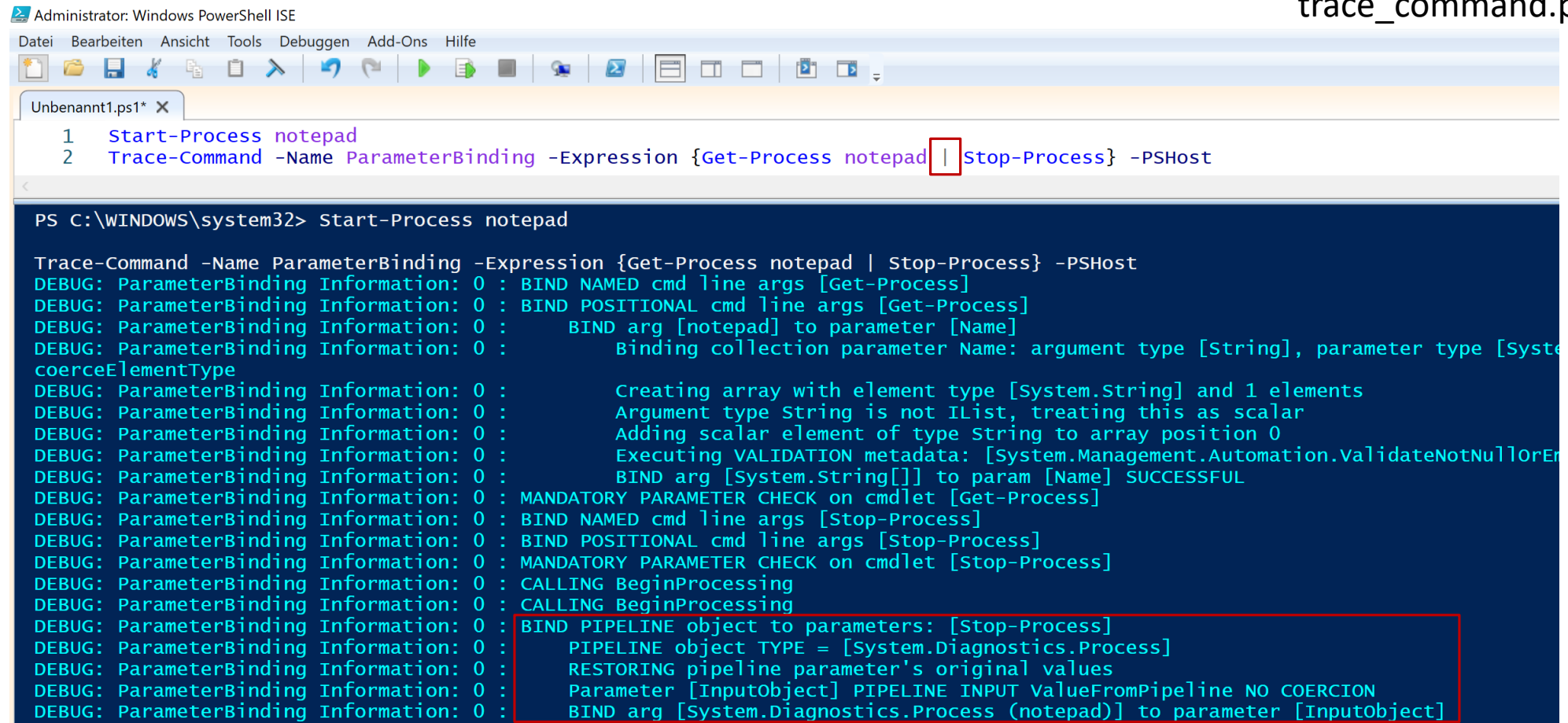
- **Trace-Command** ermöglicht eine **Ablaufverfolgung**
- Was passiert bei **Get-Process notepad** ?

```
Unbenannt1.ps1* x
1 Trace-Command -Name CommandDiscovery,ParameterBinding -Expression {Get-Process notepad} -PSHost
2
3
```

```
DEBUG: CommandDiscovery Information: 0 : Looking up command: Get-Process
DEBUG: CommandDiscovery Information: 0 : Cmdlet found: Get-Process Microsoft.PowerShell.Commands.GetProce
DEBUG: ParameterBinding Information: 0 : BIND NAMED cmd line args [Get-Process]
DEBUG: ParameterBinding Information: 0 : BIND POSITIONAL cmd line args [Get-Process]
DEBUG: ParameterBinding Information: 0 : BIND arg [notepad] to parameter [Name]
DEBUG: ParameterBinding Information: 0 : Binding collection parameter Name: argument type [String]
element type [System.String], no coerceElementType
DEBUG: ParameterBinding Information: 0 : Creating array with element type [System.String] and 1 el
DEBUG: ParameterBinding Information: 0 : Argument type String is not IList, treating this as scala
DEBUG: ParameterBinding Information: 0 : Adding scalar element of type String to array position 0
DEBUG: ParameterBinding Information: 0 : Executing VALIDATION metadata: [System.Management.Automat
DEBUG: ParameterBinding Information: 0 : BIND arg [System.String[]] to param [Name] SUCCESSFUL
DEBUG: ParameterBinding Information: 0 : MANDATORY PARAMETER CHECK on cmdlet [Get-Process]
DEBUG: ParameterBinding Information: 0 : CALLING BeginProcessing
DEBUG: ParameterBinding Information: 0 : CALLING EndProcessing
```

# Looking behind the scenes

trace\_command.ps1



The screenshot shows the Windows PowerShell ISE interface. The script editor contains two lines of code:

```
1 Start-Process notepad
2 Trace-Command -Name ParameterBinding -Expression {Get-Process notepad | Stop-Process} -PSHost
```

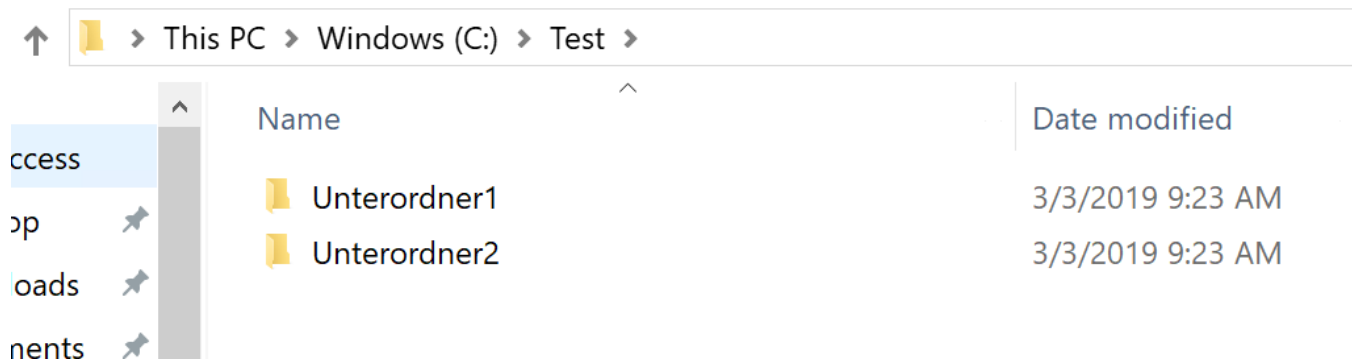
The command prompt shows the execution of the script, resulting in a series of debug messages from the ParameterBinding module. A red box highlights the following output lines:

```
DEBUG: ParameterBinding Information: 0 : BIND PIPELINE object to parameters: [Stop-Process]
DEBUG: ParameterBinding Information: 0 : PIPELINE object TYPE = [System.Diagnostics.Process]
DEBUG: ParameterBinding Information: 0 : RESTORING pipeline parameter's original values
DEBUG: ParameterBinding Information: 0 : Parameter [InputObject] PIPELINE INPUT ValueFromPipeline NO COERCION
DEBUG: ParameterBinding Information: 0 : BIND arg [System.Diagnostics.Process (notepad)] to parameter [InputObject]
```



# Übung – Piping mit Get-ChildItem

1. Erstellen Sie die **Orderstruktur**:



2. Erstellen Sie in Unterordner1 und Unterordner2 eine **Datei**
3. Zeigen Sie mit **Get-ChildItem** nur die **Dateien** an
4. Verbinden Sie **Get-ChildItem** mit der **Pipe** mit **Remove-Item** um alle **Dateien** in den Unterordnern zu **löschen**, aber **NICHT** die **Unterordner** selbst

remove\_multiple\_files\_recurse.ps1

# Übung – Piping mit Active Directory

1. DC: Pipen Sie **Get-ADUser** zu **Set-ADUser** um die StreetAddress eines Benutzers zu ändern
2. Verwenden Sie **Get-ADUser -Filter \*** und **-SearchBase ""** um bei Benutzern einer OU bei City Vienna einzutragen

```
PS C:\> Get-ADUser -Filter * -SearchBase "OU=Finance,OU=UserAccounts,DC=FABRIKAM,DC=COM"
```

Hannah Linux Properties ? X

Member Of	Dial-in	Environment	Sessions		
Remote control	Remote Desktop Services Profile		COM+		
General	Address	Account	Profile	Telephones	Organization

Street: PowerShell Gasse 7

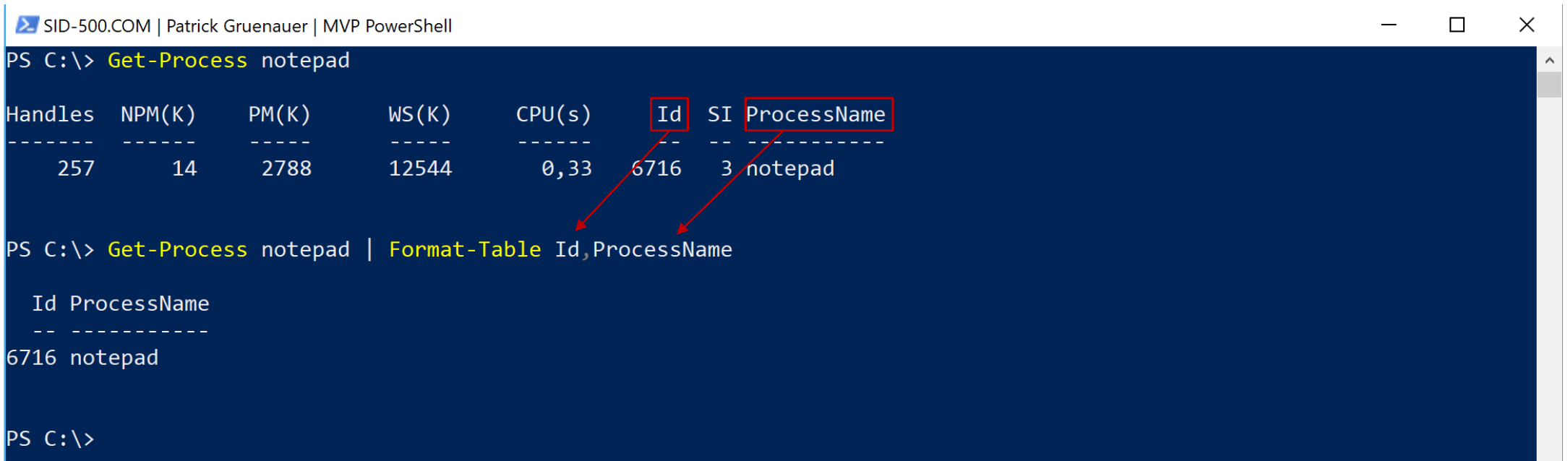
change\_ad\_users.ps1

---

# Formatierung und Ausgabe

# Format-Table

- PowerShell gibt die Ausgabe per default als **Tabelle oder Liste** aus
- Pipe | **Format-Table** gibt die Ausgabe als Tabelle aus
- Limitierung (Default): **max. 10 Spalten**



```
SID-500.COM | Patrick Gruenauer | MVP PowerShell
PS C:\> Get-Process notepad

Handles  NPM(K)  PM(K)  WS(K)  CPU(s)  Id  SI  ProcessName
-----  -
      257     14   2788  12544    0,33  6716  3  notepad

PS C:\> Get-Process notepad | Format-Table Id,ProcessName

   Id ProcessName
   -- -
6716 notepad

PS C:\>
```

# Format-Table

- Erweiterte Formatierung :
  - **-AutoSize**
  - **-Wrap**

```
SID-500.COM | Patrick Gruenauer | MVP PowerShell
PS C:\> Get-Service BDESVC

Status  Name          DisplayName
-----  -
Running BDESVC        BitLocker-Laufwerkverschlüsselungsd...
```

```
SID-500.COM | Patrick Gruenauer | MVP PowerShell
PS C:\> Get-Service BDESVC | Format-Table -AutoSize -Wrap

Status  Name  DisplayName
-----  -
Running BDESVC BitLocker-Laufwerkverschlüsselungsdienst

PS C:\>
```

# Format-List

- **Format-Table** kann bis zu 10 Spalten darstellen
- **Format-List** umgeht diese Beschränkung

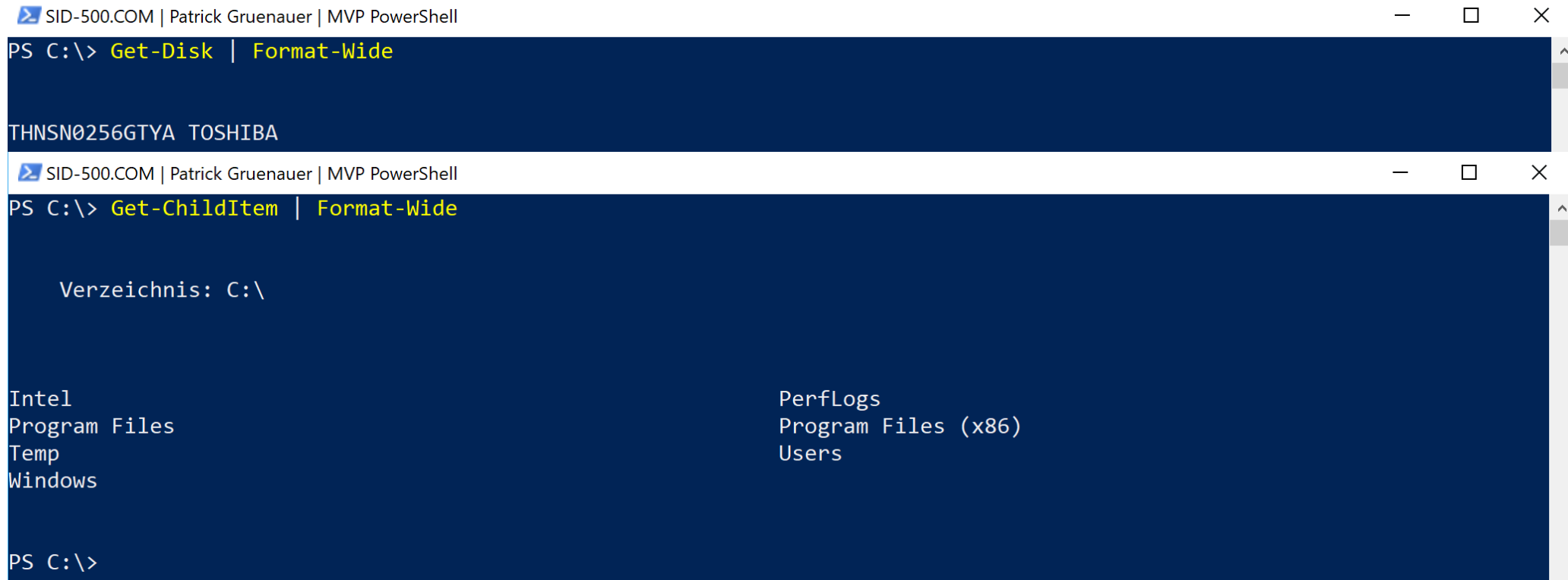
SID-500.COM | Patrick Gruenauer | MVP PowerShell

```
PS C:\> Get-Disk | Format-List

UniqueId       : eui.00080D010018DA30
Number         : 0
Path           : \\?\scsi#disk&ven_nvme&prod_thnsn0256gtya_to#5&736164b&0&000000#{53f56307-b6bf-11d0-94f2-00a0c91ef
                b8b}
Manufacturer   :
Model          : THNSN0256GTYA TOSHIBA
SerialNumber   : 0008_0D01_0018_DA30.
Size           : 238.47 GB
AllocatedSize  : 256060514304
LogicalSectorSize : 512
PhysicalSectorSize : 4096
NumberOfPartitions : 4
PartitionStyle : GPT
IsReadOnly     : False
IsSystem       : True
IsBoot         : True
```

# Format-Wide

- **Format-Wide** kann eine „weite“ Tabelle ausgeben
- **Format-Wide** zeigt nur **eine Eigenschaft** an



```
SID-500.COM | Patrick Gruenauer | MVP PowerShell
PS C:\> Get-Disk | Format-Wide

THNSN0256GTYA TOSHIBA

SID-500.COM | Patrick Gruenauer | MVP PowerShell
PS C:\> Get-ChildItem | Format-Wide

Verzeichnis: C:\

Intel                                PerfLogs
Program Files                        Program Files (x86)
Temp                                  Users
Windows

PS C:\>
```

# Import-CSV | Export-CSV

---

- Ein mächtiges Tool zur Dokumentation: Import und Export: **Import-CSV** und **Export-CSV**

```
Unbenannt1.ps1* X
1  Get-Process | Export-Csv -Path C:\Temp\Processes.csv -Delimiter : -NoTypeInfoation
2  Import-Csv -Path C:\Temp\Processes.csv -Delimiter :
3
<
PS C:\> |
```

import-csv\_export-csv.ps1



# ConvertTo-CSV | ConvertFrom-CSV

```
convertfrom-csv_ad_computer.ps1* x
1 (Get-ADComputer -Filter *).Name |
2 Foreach-Object {Invoke-Command -ComputerName $_ {systeminfo /FO CSV}} |
3 ConvertFrom-Csv |
4 Out-GridView
```

PS C:\>

(Get-ADComputer -Filter \*).Name | Foreach-Object {Invoke-Command -ComputerName \$\_ {systeminfo /FO CSV}} | ConvertFrom-Csv | Out-GridView

Filter

+ Add criteria

Host Name	OS Name	OS Version	OS Manufacturer	OS Configuration	OS Build Type	Registered O
DC01	Microsoft Windows Server 2016 Standard Evaluation	10.0.14393 N/A Build 14393	Microsoft Corporation	Primary Domain Controller	Multiprocessor Free	Windows-Ber
CLIENT001	Microsoft Windows 10 Pro	10.0.15063 N/A Build 15063	Microsoft Corporation	Member Workstation	Multiprocessor Free	Windows-Ber

<https://sid-500.com/2017/08/09/powershell-documenting-your-environment-by-running-systeminfo-on-all-domain-computers/>

convertfrom-csv\_ad\_computer.ps1

# Out-?

```
SID-500.COM | © Patrick Gruenauer | MVP PowerShell [2018-2020]
PS C:\> Get-Command -Verb Out

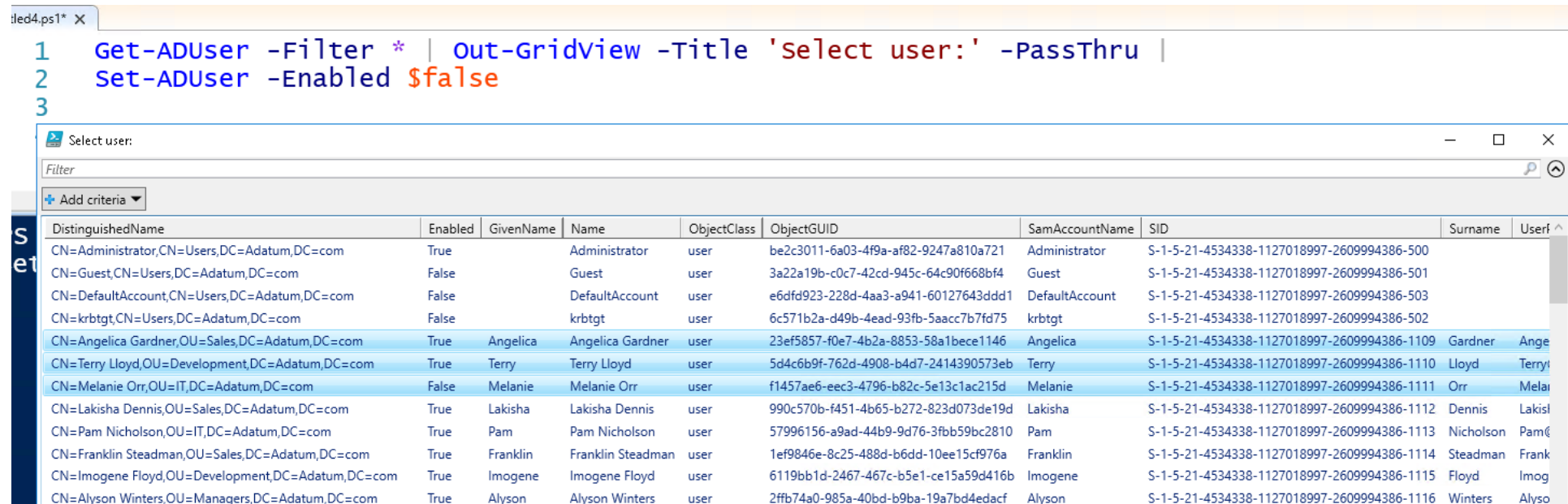
CommandType      Name                                     Version      Source
-----
SID-500.COM | Patrick Gruenauer | MVP PowerShell
PS C:\> Get-Command | Out-Host -Paging

CommandType      Name                                     Version      Source
-----
Alias             Add-ProvisionedAppxPackage             3.0          Dism
Alias             Add-ProvisioningPackage                 3.0          Provisioning
Function          Add-PartitionAccessPath                 2.0.0.0      Storage
<LEERTASTE> Nächste Seite, <WAGENRÜCKLAUF> Nächste Zeile, Q Beenden
Function          Add-PhysicalDisk                         2.0.0.0      Storage
<LEERTASTE> Nächste Seite, <WAGENRÜCKLAUF> Nächste Zeile, Q Beenden
PS C:\> █
```

# Trainer-Demo: Out-GridView

- **Out-GridView** gibt den Output **grafisch** aus
- **-PassThru** speichert die Auswahl für die Weiterverarbeitung

```
1 Get-ADUser -Filter * | Out-GridView -Title 'select user:' -PassThru |
2 Set-ADUser -Enabled $false
3
```



DistinguishedName	Enabled	GivenName	Name	ObjectClass	ObjectGUID	SamAccountName	SID	Surname	Userf
CN=Administrator,CN=Users,DC=Adatum,DC=com	True	Administrator	Administrator	user	be2c3011-6a03-4f9a-af82-9247a810a721	Administrator	S-1-5-21-4534338-1127018997-2609994386-500		
CN=Guest,CN=Users,DC=Adatum,DC=com	False	Guest	Guest	user	3a22a19b-c0c7-42cd-945c-64c90f668bf4	Guest	S-1-5-21-4534338-1127018997-2609994386-501		
CN=DefaultAccount,CN=Users,DC=Adatum,DC=com	False	DefaultAccount	DefaultAccount	user	e6dfd923-228d-4aa3-a941-60127643ddd1	DefaultAccount	S-1-5-21-4534338-1127018997-2609994386-503		
CN=krbtgt,CN=Users,DC=Adatum,DC=com	False	krbtgt	krbtgt	user	6c571b2a-d49b-4ead-93fb-5aacc7b7fd75	krbtgt	S-1-5-21-4534338-1127018997-2609994386-502		
CN=Angelica Gardner,OU=Sales,DC=Adatum,DC=com	True	Angelica	Angelica Gardner	user	23ef5857-f0e7-4b2a-8853-58a1bece1146	Angelica	S-1-5-21-4534338-1127018997-2609994386-1109	Gardner	Ange
CN=Terry Lloyd,OU=Development,DC=Adatum,DC=com	True	Terry	Terry Lloyd	user	5d4c6b9f-762d-4908-b4d7-2414390573eb	Terry	S-1-5-21-4534338-1127018997-2609994386-1110	Lloyd	Terry
CN=Melanie Orr,OU=IT,DC=Adatum,DC=com	False	Melanie	Melanie Orr	user	f1457ae6-eec3-4796-b82c-5e13c1ac215d	Melanie	S-1-5-21-4534338-1127018997-2609994386-1111	Orr	Melari
CN=Lakisha Dennis,OU=Sales,DC=Adatum,DC=com	True	Lakisha	Lakisha Dennis	user	990c570b-f451-4b65-b272-823d073de19d	Lakisha	S-1-5-21-4534338-1127018997-2609994386-1112	Dennis	Lakisha
CN=Pam Nicholson,OU=IT,DC=Adatum,DC=com	True	Pam	Pam Nicholson	user	57996156-a9ad-44b9-9d76-3fbb59bc2810	Pam	S-1-5-21-4534338-1127018997-2609994386-1113	Nicholson	Pam
CN=Franklin Steadman,OU=Sales,DC=Adatum,DC=com	True	Franklin	Franklin Steadman	user	1ef9846e-8c25-488d-b6dd-10ee15cf976a	Franklin	S-1-5-21-4534338-1127018997-2609994386-1114	Steadman	Frank
CN=Imogene Floyd,OU=Development,DC=Adatum,DC=com	True	Imogene	Imogene Floyd	user	6119bb1d-2467-467c-b5e1-ce15a59d416b	Imogene	S-1-5-21-4534338-1127018997-2609994386-1115	Floyd	Imog
CN=Alyson Winters,OU=Managers,DC=Adatum,DC=com	True	Alyson	Alyson Winters	user	2ffb74a0-985a-40bd-b9ba-19a7bd4edacf	Alyson	S-1-5-21-4534338-1127018997-2609994386-1116	Winters	Alyson

# Out-GridView

The screenshot shows a PowerShell console window titled "Unbenannt1.ps1\* X". The script content is as follows:

```
1 Start-Process notepad
2
3 Get-Process | Sort-Object -Property ProcessName |
4 Out-GridView -Title 'Select Process' -PassThru |
5 Stop-Process
```

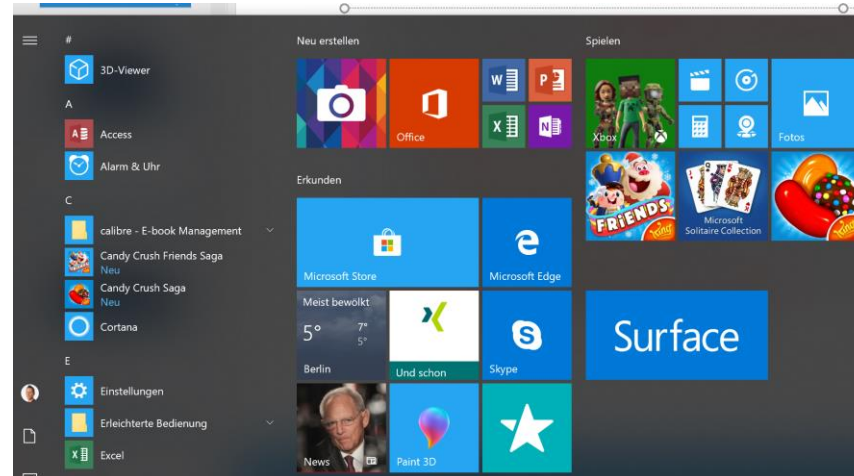
Below the script, the "Select Process" dialog is visible. It includes a "Filter" field and a "+ Kriterien hinzufügen" button. The main area displays a table of running processes:

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
678	36	32100	39468	36,41	7 320	1	ApplicationFrameHost
183	11	8692	14752	0,08	6 520	0	audiodg
539	25	8880	32252	0,34	5 692	1	browser_broker
544	27	21068	528	1,17	6 160	1	Calculator
118	7	6436	568	0,02	3 680	0	conhost
682	22	1824	1012	2,82	768	0	csrss

out-gridview\_start\_stop\_process.ps1

# Übung: Format-Table, Out-File, Out-GridView

1. Rufen Sie auf **Windows 10** mit **Get-AppxPackage** alle Apps ab und zeigen Sie im **Tabellenformat**
  - a. die **Namen** der Apps
  - b. den **Installationsspeicherort**
  - c. die Ausgabe **gut lesbar** (Zeilenumbrüche, Abstände)
2. Pipen Sie (1) an **Out-File** und speichern Sie die Ausgabe als Datei



appx\_format-table.ps1

# Übung – Export-CSV

---

1. DC: Öffnen Sie [export-csv\\_ad\\_users.ps1](#)
2. Ändern Sie ggf. den [Pfad](#) der Ausgabedatei
3. Drücken Sie **F5**. Ist die CSV Datei da?
4. Ändern Sie das Skript, sodass das Attribut [logonCount](#) hinzugefügt wird

	A	B	C	D	E	F	G	H	I
1	Name,"SID",	whenCreated",	LastLogonDate"						
2	patrick,"S-1-5-21-4011808116-897646742-2731590833-500",	"1/11/2019 3:01:18 PM",	"1/14/2020 2:44:47 PM"						
3	Guest,"S-1-5-21-4011808116-897646742-2731590833-501",	"1/11/2019 3:01:18 PM",							
4	krbtgt,"S-1-5-21-4011808116-897646742-2731590833-502",	"1/11/2019 3:03:23 PM",							
5	...	...	...	...	...	...	...	...	...

export-csv\_ad\_users.ps1

# Excel Modul (PSGallery)

```
SID-500.COM | © Patrick Gruenauer | MVP PowerShell [2018-2020]
PS C:\> Get-Command -Module ImportExcel
```

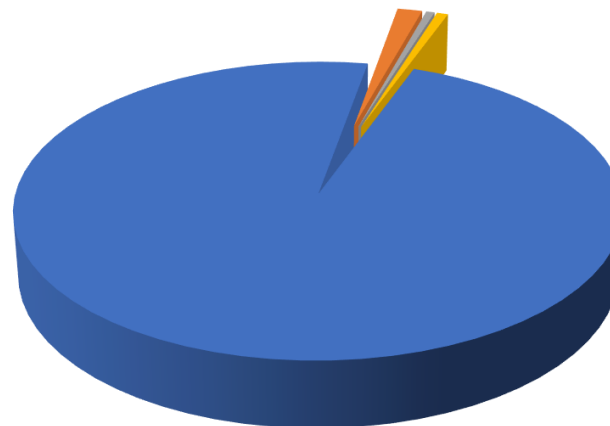
CommandType	Name	Version	Source
Alias	Convert-XlRangeToImage	7.0.1	ImportExcel
Alias	Export-ExcelSheet	7.0.1	ImportExcel
Alias	New-ExcelChart	7.0.1	ImportExcel
Alias	Set-Column	7.0.1	ImportExcel
Alias	Set-Format	7.0.1	ImportExcel
Alias	Set-Row	7.0.1	ImportExcel
Alias	Use-ExcelData	7.0.1	ImportExcel
Function	Add-ConditionalFormatting	7.0.1	ImportExcel
Function	Add-ExcelChart	7.0.1	ImportExcel
Function	Add-ExcelDataValidationRule	7.0.1	ImportExcel
Function	Add-ExcelName	7.0.1	ImportExcel
Function	Add-ExcelTable	7.0.1	ImportExcel
Function	Add-PivotTable	7.0.1	ImportExcel
Function	Add-Worksheet	7.0.1	ImportExcel
Function	BarChart	7.0.1	ImportExcel
Function	Close-ExcelPackage	7.0.1	ImportExcel

# Export-Excel

```
1 Install-Module ImportExcel
2
3 Get-Process | Where-Object Company |
4     Export-Excel C:\Temp\ps.xlsx -Show `
5     -IncludePivotTable -PivotRows Company -PivotData @{Handles="sum"} `
6     -IncludePivotChart -ChartType PieExploded3D
```

Sum of Handles	Ergebnis
Microsoft Corporation	74445
Intel Corporation	1179
Intel	401
Realtek Semiconductor	536
<b>Gesamtergebnis</b>	<b>76561</b>

<https://devblogs.microsoft.com/scripting/introducing-the-powershell-excel-module-2/>



■ Microsoft Corporation  
■ Intel Corporation  
■ Intel  
■ Realtek Semiconductor

export-excel\_pivot.ps1



---

`$_` und `$PSItem`

# \$\_ und \$PSItem

\$\_  
Same as \$PSItem. Contains the current object in the pipeline object. You can use this variable in commands that perform an action on every object or on selected objects in a pipeline.

- \$\_ und \$PSItem sind idente System-Variablen und werden als **Pipeline Variablen** bezeichnet
- \$\_ und \$PSItem beinhalten **das aktuell von der Pipeline verarbeitete Objekt**

## The PowerShell Pipe



\$\_ beinhaltet beim **ersten Durchlauf** das Objekt **notepad**, beim **zweiten Durchlauf** das Objekt **mspaint**

The pipe takes everything on the left of the pipe and forwards it to the command to the right of the pipe

# \$\_

---

- Prozesse mit einer CPU Zeit von größer als 10 Sek. aufrufen → \$\_

**Get-Process:** Rufe alle Prozesse auf

**\$\_:** die von der Pipe kommen ...

**.CPU:** und als CPU-Zeit ...

```
PS C:\> Get-Process | Where-Object { $_.CPU -GT 10 }
```

**Where-Object:** aber zeige nur solche Prozesse (Pipeline-Objekte) „wo alle Objekte ...“

**-GT 10:** mehr als 10 Sekunden aufweisen

# \$\_PSItem

- Prozesse mit einer CPU Zeit von größer als 10 Sek. aufrufen → **\$\_PSItem**

```
Auswählen SID-500.COM | Patrick Gruenauer | MVP PowerShell
PS C:\> Get-Process | Where-Object {$PSItem.CPU -GT 10}
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
614	18	16468	21012	31,09	12568	2	ctfmon
3053	90	69456	105876	115,20	14892	2	explorer
1856	88	64956	110796	53,27	2944	2	MicrosoftEdge
1226	95	104548	139160	19,91	2940	2	MicrosoftEdgeCP
1561	147	211004	65160	79,55	6220	2	MicrosoftEdgeCP
1592	131	213884	69308	52,98	6964	2	MicrosoftEdgeCP
1663	185	306608	266500	76,59	7964	2	MicrosoftEdgeCP
1244	118	146172	189788	21,08	8444	2	MicrosoftEdgeCP
521	33	10912	22604	50,36	8832	2	MicrosoftEdgeCP
1523	125	170136	159308	42,39	11372	2	MicrosoftEdgeCP
1084	97	112124	150104	17,53	11640	2	MicrosoftEdgeCP
1237	81	139576	54728	109,58	14096	2	MicrosoftEdgeCP
1174	75	135168	50400	58,81	16168	2	MicrosoftEdgeCP
1352	115	170964	64604	20,95	16216	2	MicrosoftEdgeCP
826	50	32716	52116	83,66	7824	2	OneDrive
2933	100	109524	160524	50,19	720	2	OUTLOOK
1966	88	231980	286572	262,61	14340	2	POWERPNT
379	28	131396	146216	130,20	13616	2	RuntimeBroker

# \$\_ \$PSItem ?

- Prozesse mit einer CPU Zeit von größer als 10 Sek. aufrufen → ?

```
SID-500.COM | Patrick Gruenauer | MVP PowerShell
PS C:\> Get-Process | Where-Object CPU -GT 10
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
614	18	16024	20628	30,31	12568	2	ctfmon
3039	90	69396	105968	114,48	14892	2	explorer
1862	88	65036	111048	53,25	2944	2	MicrosoftEdge
1226	95	104572	139276	19,91	2940	2	MicrosoftEdgeCP
1561	147	211004	65264	79,55	6220	2	MicrosoftEdgeCP
1592	131	213884	69424	52,98	6964	2	MicrosoftEdgeCP
1670	185	295212	254916	76,30	7964	2	MicrosoftEdgeCP
1238	116	146080	189788	21,06	8444	2	MicrosoftEdgeCP
521	33	10940	22668	50,36	8832	2	MicrosoftEdgeCP
1523	125	170136	159464	42,39	11372	2	MicrosoftEdgeCP
1084	97	112136	150196	17,53	11640	2	MicrosoftEdgeCP
1237	81	139576	54820	109,58	14096	2	MicrosoftEdgeCP
1174	75	135168	50472	58,81	16168	2	MicrosoftEdgeCP
1352	115	170964	64728	20,95	16216	2	MicrosoftEdgeCP
828	51	32328	51888	80,27	7824	2	OneDrive
2929	99	109448	159468	49,97	720	2	OUTLOOK
1999	85	210164	262336	236,69	14340	2	POWERPNT

kein \$\_ oder \$PSItem nötig?

# \$\_ und \$PSItem

---

```
Get-Process | Where-Object -FilterScript {$_.CPU -gt 50}
```

This is more usually written as

```
Get-Process | Where {$_.CPU -gt 50}
```

As of PowerShell v3 if you are filtering on ONE property you can simplify the syntax

```
Get-Process | Where CPU -gt 50
```

which is a truncated version of

```
Get-Process | Where -Property CPU -gt -Value 50
```

When you write it like this its obvious what is happening. Get in the habit of thinking of the syntax in this manner even if you write in the shortened form

If you need to filter on TWO or more properties you have to use the old style syntax

```
Get-Process | Where {$_.CPU -gt 50 -AND $_.Handles -gt 1000}
```

Quelle: [itknowledgeexchange.techtarget.com](http://itknowledgeexchange.techtarget.com)

# Übung – \$\_

---

- Zeigen Sie mit **Get-Process** und **Where-Object** Prozesse an, welche
  - a. mit **s** beginnen
  - b. und eine **CPU** Zeit von unter 5 Sekunden haben

If you need to filter on TWO or more properties you have to use the old style syntax

```
Get-Process | Where {$_.CPU -gt 50 -AND $_.Handles -gt 1000}
```

get-process\_where-object.ps1

---

# Filtering



# Filtering

---

- Mit **-Filter** werden nur **ausgewählte Objekte** an die **Pipe** gesendet

```
02 . Codes . PowerShell Pipe > > disable_all_win8_computer.ps1
1   Get-ADComputer -Properties * -Filter 'operatingsystem -like "*windows 8*" |
2   Set-ADComputer -Enabled $false
3
4
```

disable\_all\_win8\_computer.ps1

# Piping – Active Directory

- Piping in Action: **Member-Server** mit **Restart-Computer** neu starten

Rufe mit **-Filter** alle Server ab, außer Mitglieder der Gruppe 516 (Domain-Controller)

alle\_member\_server\_neu\_starten.ps1 x

```
1 Get-ADComputer -Properties Name -Filter {operatingsystem -like "*server*" -and primarygroupid -ne "516"} |
2 Select-Object -ExpandProperty Name |
3 Restart-Computer -Force -ErrorAction SilentlyContinue
4
```

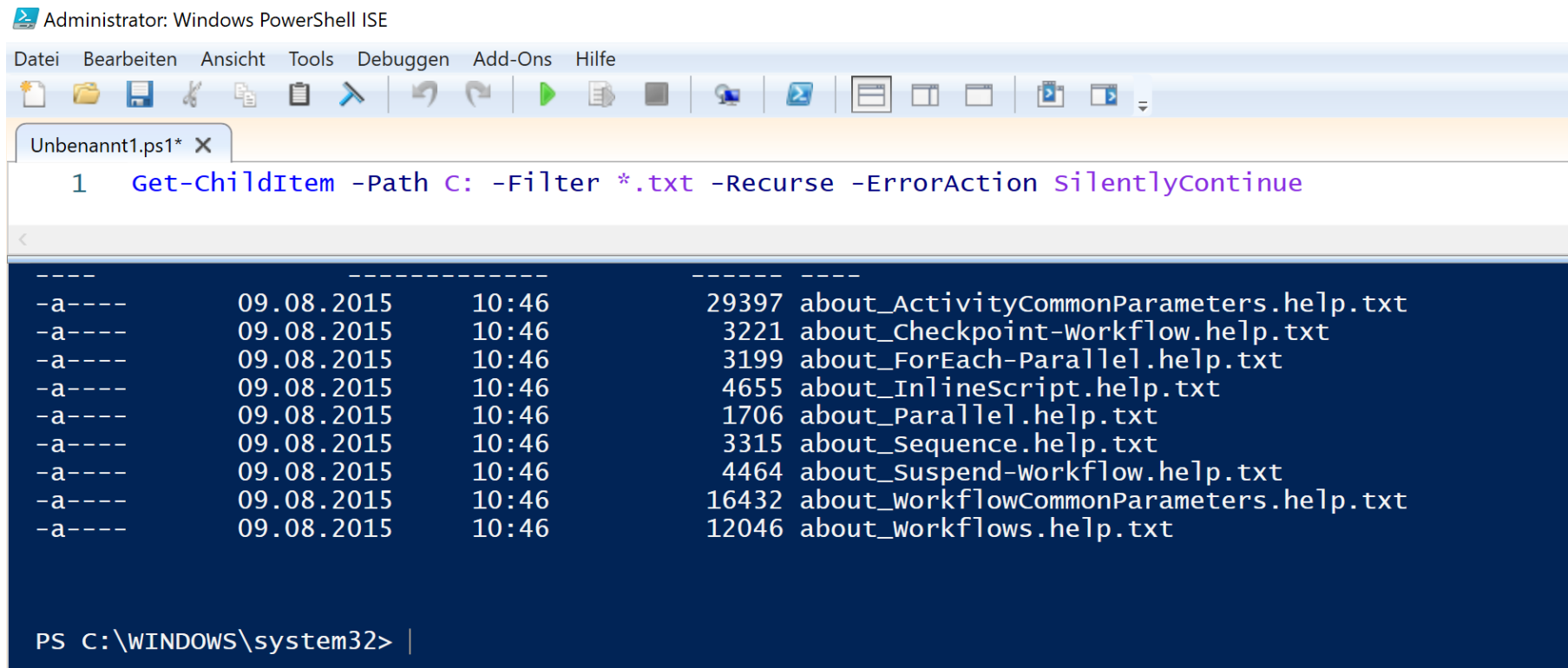
Rufe mit **-ExpandProperty** nur die Objekteigenschaft Name auf. Der Computername (**nur!**) wird an die Pipe weitergereicht

Jeder Name, welcher von der Pipe weitergereicht wird, wird an **Restart-Computer** gesendet. Alle Computer werden der Reihe nach neu gestartet. Der Parameter **-Force** startet auch Computer neu, wenn Benutzer angemeldet sind, **-ErrorAction SilentlyContinue** legt fest, dass auch bei Fehlern (Computer ausgeschaltet) weitergemacht werden soll

alle\_member\_server\_neu\_starten.ps1

# Filtering

- Nach **\*.txt** Dateien suchen (bei Fehlern einfach weiter machen)



The screenshot shows the Windows PowerShell ISE interface. The title bar reads "Administrator: Windows PowerShell ISE". The menu bar includes "Datei", "Bearbeiten", "Ansicht", "Tools", "Debuggen", "Add-Ons", and "Hilfe". The toolbar contains various icons for file operations and execution. The command prompt shows the following command:

```
1 Get-ChildItem -Path c: -Filter *.txt -Recurse -ErrorAction silentlyContinue
```

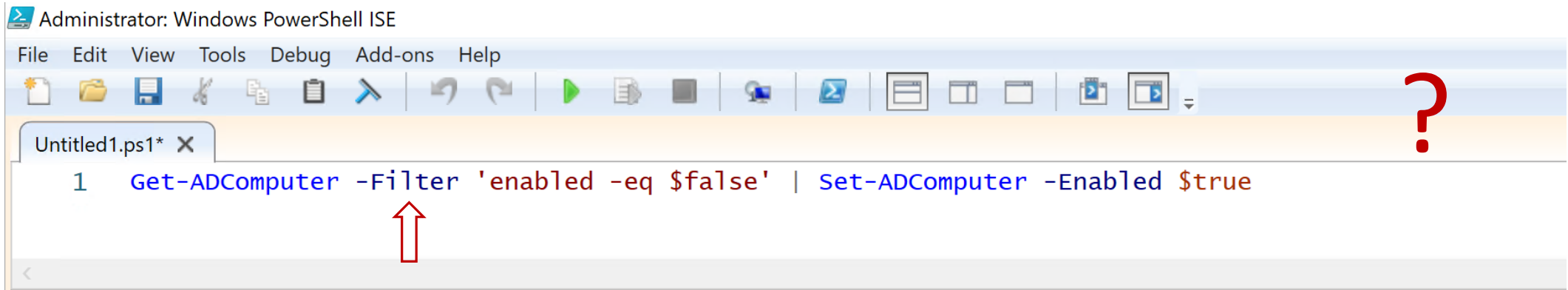
The output of the command is displayed in a dark blue console window with white text. It lists several help files with their sizes, dates, and times:

Size	Date	Time	File Name
29397	09.08.2015	10:46	about_ActivityCommonParameters.help.txt
3221	09.08.2015	10:46	about_Checkpoint-workflow.help.txt
3199	09.08.2015	10:46	about_ForEach-Parallel.help.txt
4655	09.08.2015	10:46	about_InlineScript.help.txt
1706	09.08.2015	10:46	about_Parallel.help.txt
3315	09.08.2015	10:46	about_Sequence.help.txt
4464	09.08.2015	10:46	about_Suspend-workflow.help.txt
16432	09.08.2015	10:46	about_workflowCommonParameters.help.txt
12046	09.08.2015	10:46	about_workflows.help.txt

The prompt at the bottom of the console is "PS C:\WINDOWS\system32> |".

nach\_txt\_suchen.ps1

# Filtering – Where-Object vs. Filtering



Administrator: Windows PowerShell ISE

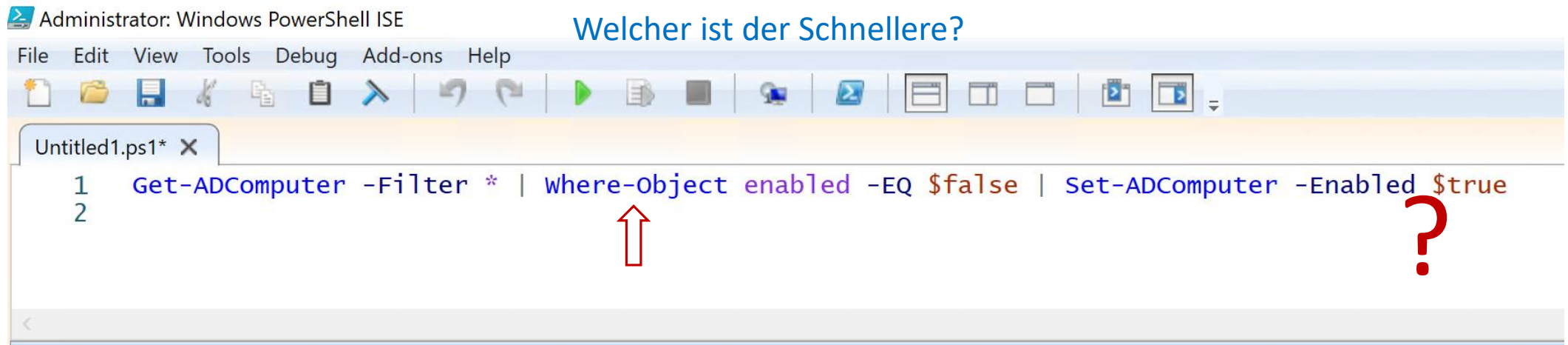
```
File Edit View Tools Debug Add-ons Help
```

Untitled1.ps1\* X

```
1 Get-ADComputer -Filter 'enabled -eq $false' | Set-ADComputer -Enabled $true
```

A red arrow points to the `-Filter` parameter in the command. A large red question mark is positioned on the right side of the screenshot.

Beide Befehle machen dasselbe ...  
Welcher ist der Schnellere?



Administrator: Windows PowerShell ISE

```
File Edit View Tools Debug Add-ons Help
```

Untitled1.ps1\* X

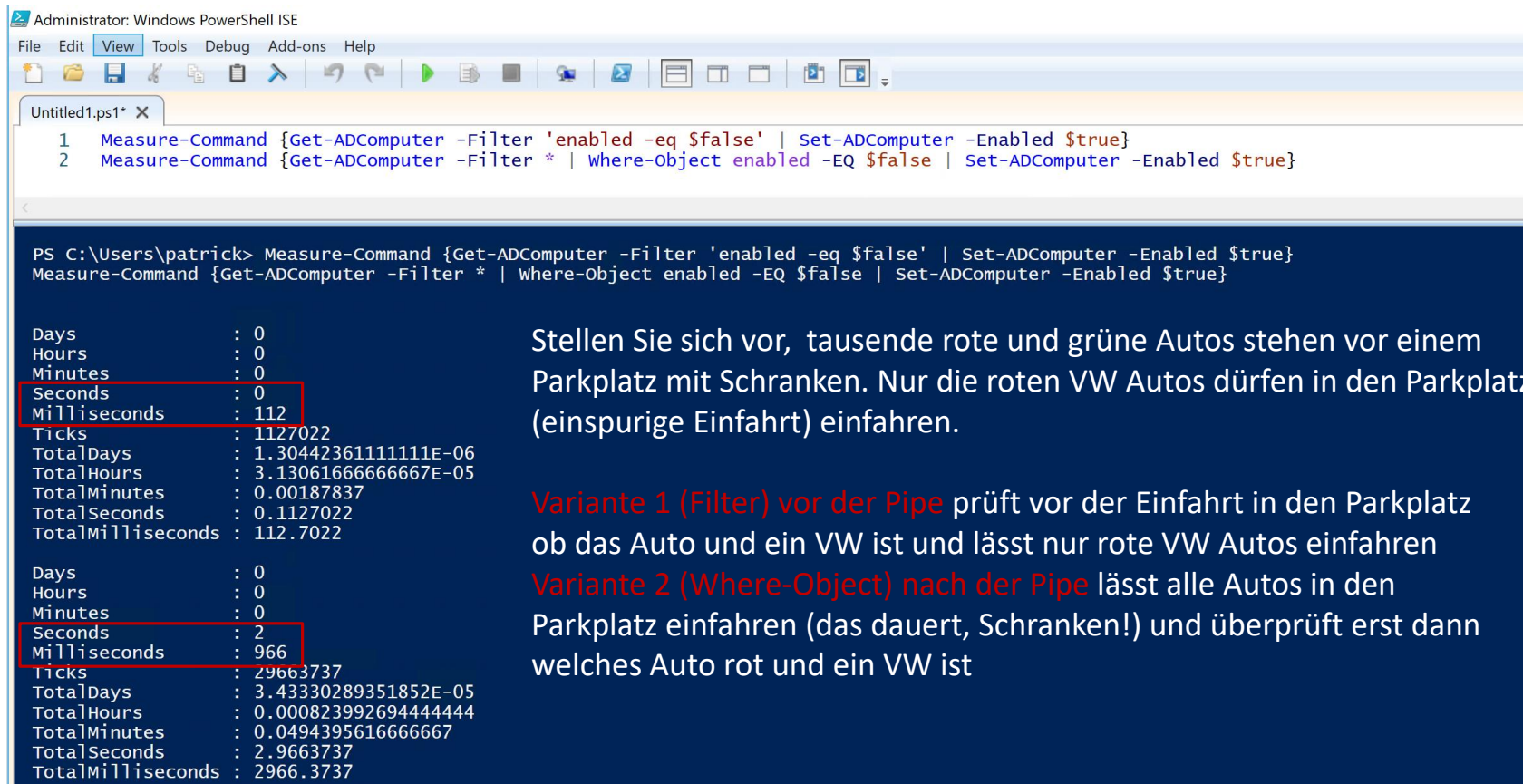
```
1 Get-ADComputer -Filter * | where-Object enabled -EQ $false | Set-ADComputer -Enabled $true
```

```
2
```

A red arrow points to the `where-Object` command in the script. A large red question mark is positioned on the right side of the screenshot.

# Filtering – Where-Object vs. Filtering

- Welcher ist der Schnellere?



```
Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1* x
1 Measure-Command {Get-ADComputer -Filter 'enabled -eq $false' | Set-ADComputer -Enabled $true}
2 Measure-Command {Get-ADComputer -Filter * | Where-Object enabled -EQ $false | Set-ADComputer -Enabled $true}

PS C:\Users\patrick> Measure-Command {Get-ADComputer -Filter 'enabled -eq $false' | Set-ADComputer -Enabled $true}
Measure-Command {Get-ADComputer -Filter * | Where-Object enabled -EQ $false | Set-ADComputer -Enabled $true}

Days           : 0
Hours          : 0
Minutes       : 0
Seconds       : 0
Milliseconds   : 112
Ticks         : 1127022
TotalDays     : 1.304423611111111E-06
TotalHours    : 3.130616666666667E-05
TotalMinutes  : 0.00187837
TotalSeconds  : 0.1127022
TotalMilliseconds : 112.7022

Days           : 0
Hours          : 0
Minutes       : 0
Seconds       : 2
Milliseconds   : 966
Ticks         : 29663737
TotalDays     : 3.43330289351852E-05
TotalHours    : 0.0008239926944444444
TotalMinutes  : 0.04943956166666667
TotalSeconds  : 2.9663737
TotalMilliseconds : 2966.3737
```

Stellen Sie sich vor, tausende rote und grüne Autos stehen vor einem Parkplatz mit Schranken. Nur die roten VW Autos dürfen in den Parkplatz (einspurige Einfahrt) einfahren.

**Variante 1 (Filter) vor der Pipe** prüft vor der Einfahrt in den Parkplatz ob das Auto und ein VW ist und lässt nur rote VW Autos einfahren

**Variante 2 (Where-Object) nach der Pipe** lässt alle Autos in den Parkplatz einfahren (das dauert, Schranken!) und überprüft erst dann welches Auto rot und ein VW ist

# Where-Object

- Steht die **Filtering nicht zur Verfügung** muss **Where-Object** aushelfen
- Beispiel: **Get-Hotfix** → **Where-Object**

```
SID-500.COM | © Patrick Gruenauer | MVP PowerShell [2017-2019]
PS C:\> Get-Hotfix | Where-Object InstalledOn -le (Get-Date).AddDays(-50)
```

Source	Description	HotFixID	InstalledBy	InstalledOn
SURFACEPRO7	Security Update	KB4497727		01.04.2019 00:00:00
SURFACEPRO7	Security Update	KB4508433		03.09.2019 00:00:00
SURFACEPRO7	Security Update	KB4516115	NT-AUTORITÄT\SYSTEM	26.10.2019 00:00:00
SURFACEPRO7	Security Update	KB4521863	NT-AUTORITÄT\SYSTEM	26.10.2019 00:00:00

# Übung – Filtering mit AD User

---

1. Erstellen Sie am DC 100 Benutzerkonten Student1 – Student 100

```
1..100 | ForEach-Object {New-ADUser -Name "Schueler$_" `
  -AccountPassword (ConvertTo-SecureString -AsPlainText '123user#' -Force) `
  -Enabled $true}
```

2. Rufen Sie mit **Get-ADUser -Filter ''** alle Benutzer aus (1) ab
3. Verwenden Sie die **Pipe** und **Disable-ADAccount** um alle zu deaktivieren

 Schueler18	User
 Schueler19	User
 Schueler2	User
 Schueler20	User
 Schueler21	User
 Schueler22	User
 Schueler23	User
 Schueler24	User
...	...

ad\_user\_filtern\_und\_deaktivieren.ps1

# Exkurs: Filtering und Where-Object in Exchange | Office 365

---

```
PS C:\> Get-Mailbox -RecipientTypeDetails UserMailBox -ResultSize Unlimited |  
Get-MailboxStatistics |  
Where-Object {$_.DisconnectDate} |  
Sort-Object -Property TotalItemSize -Descending |  
Select-Object DisplayName,ItemCount,TotalItemSize,LastLogonTime
```

DisplayName	ItemCount	TotalItemSize	LastLogonTime
Patrick Grünauer   sid-500.com	24080	2.957 GB (3,175,425,412 bytes)	02.03.2020 19:17:16
Franz Huber	354	3.794 MB (3,977,801 bytes)	03.01.2020 00:06:54
Alert	35	628.1 KB (643,167 bytes)	02.03.2020 16:45:30

exchange\_online\_mailbox\_commands.ps1



---

# Sort-Object und Select-Object

# Sort-Object

- **Sort-Object** sortiert Objekte

```
SID-500.COM | © Patrick Gruenauer | MVP PowerShell
PS C:\> Get-Process | Sort-Object -Property CPU -Descending
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
6152	0	184	24	5.951,72	4	0	System
4744	71	285280	139972	2.724,45	1132	1	dwm
933	44	279660	52444	1.380,84	2588	1	obs64
2946	78	609996	116032	1.039,36	4132	0	MsMpEng
256	14	75372	16168	1.016,13	14680	0	audiodg
419	18	10588	17148	770,27	2396	1	svchost
1105	85	35664	25096	736,31	11112	1	Steam
4430	1818	176844	120396	729,42	4648	1	explorer
386	10	2552	4140	633,48	2928	0	WUDFHost
329	24	3676	7452	466,89	1272	0	svchost

# Select-Object

- **Select-Object** wählt Objekteigenschaften

```
SID-500.COM | © Patrick Gruenauer | MVP PowerShell
PS C:\> Get-Process | Sort-Object -Property WS | Select-Object -Last 5
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
1537	73	165916	223596	141,84	16904	1	POWERPNT
1385	118	193796	260272	9,41	17048	1	MicrosoftEdgeCP
1629	229	341200	393948	45,59	15176	1	MicrosoftEdgeCP
2305	266	447432	449272	231,98	13440	1	MicrosoftEdgeCP
1234	37	453380	463964	13,45	6068	1	mstsc

```
PS C:\>
```

# Select-Object -ExpandProperty

- **Select-Object -ExpandProperty** zeigt nur den Wert der Eigenschaft. Der Objektname wird nicht ausgegeben → Weiterverarbeitung der Daten

```
Untitled1.ps1* X
1 $a = Get-ADComputer -Filter * | select-object -Property Name
2 $b = Get-ADComputer -Filter * | select-object -ExpandProperty Name
```

```
PS C:\> Test-Connection $b
```

Source	Destination	IPV4Address	IPV6Address	Bytes
DC01	DC01	10.0.0.5	fe80::c999:6c02:5ccc:a9ba%12	32
DC01	DC01	10.0.0.5	fe80::c999:6c02:5ccc:a9ba%12	32
DC01	DC01	10.0.0.5	fe80::c999:6c02:5ccc:a9ba%12	32
DC01	DC01	10.0.0.5	fe80::c999:6c02:5ccc:a9ba%12	32

```
co01
PS C:\> Test-Connection $a
Test-Connection : Testing connection to computer '@{Name=DC01}' failed: A non-recoverable
error occurred during a database lookup
At line:1 char:1
+ Test-Connection $a
+ ~~~~~
```

# .NET Methode

---

- Die .NET Methode `() .Attribut` ist eine Alternative zu `Select-Object -ExpandProperty`

Administrator: Windows PowerShell

```
PS C:\> (Get-ADUser -Filter *).samAccountName  
patrick  
Guest  
krbtgt  
s.stollane
```

Administrator: Windows PowerShell

```
PS C:\> Get-ADUser -Filter * | Select-Object -ExpandProperty samAccountName  
patrick  
Guest  
krbtgt  
s.stollane
```

# Sort-Object und Select-Object

- **Sort-Object** sortiert
- **Select-Object** wählt
- Empfehlung: **Sort-Object** vor **Select-Object** verwenden

```
Untitled2.ps1* X
1 Get-ADUser -Filter * -Properties LastLogonDate |
2 Sort-Object LastLogonDate -Descending |
3 Select-Object -Property Name,SID,LastLogonDate
```

Name	SID	LastLogonDate
patrick	S-1-5-21-4011808116-897646742-2731590833-500	1/25/2020 12:35:08 PM
Hannah Linux	S-1-5-21-4011808116-897646742-2731590833-3603	9/18/2019 1:23:12 PM
admpeter	S-1-5-21-4011808116-897646742-2731590833-40602	6/3/2019 11:22:31 AM
Peter	S-1-5-21-4011808116-897646742-2731590833-40601	6/3/2019 11:11:11 AM
Harald Unsicher	S-1-5-21-4011808116-897646742-2731590833-4606	3/4/2019 5:03:12 PM
Hans Huber	S-1-5-21-4011808116-897646742-2731590833-93120	
Max Meier	S-1-5-21-4011808116-897646742-2731590833-93119	

sort-object\_lastlogondate.ps1

# Custom Properties

- **Select-Object** unterstützt Custom Properties
- Syntax: **@{n='Name'; e={Your Code}}**

Name	LastWriteTime	Length
backup_outlook_full.pst	01.12.2019 12:40:19	3192701952
ScaNv6_instructorPPT_Chapter2_scaling VLANs.pptx	26.09.2019 18:40:49	5494003
ScaNv6_instructorPPT_Chapter1_LAN Design.pptx	26.09.2019 16:07:31	2821815

```
Unbenannt2.ps1* X
1 Get-ChildItem -Path c:\Temp -File -Recurse | Sort-Object Length -Descending |
2 Select-Object -Property Name, LastWriteTime, @{n='Size (MB)'; e={[math]::Round((($_.length/1MB),0)}}
3
4 |
```

Name	LastWriteTime	Size (MB)
backup_outlook_full.pst	01.12.2019 12:40:19	3045
ScaNv6_instructorPPT_Chapter2_scaling VLANs.pptx	26.09.2019 18:40:49	5
ScaNv6_instructorPPT_Chapter1_LAN Design.pptx	26.09.2019 16:07:31	3
2.1.4.4 Packet Tracer - Configure VLANs, VTP, and DTP - ILM.pdf	24.11.2017 09:02:50	0

get-childitem\_custom\_property.ps1

# Trainer-Demo: RAM mit Select-Object auslesen

---

```
Manufacturer BankLabel ConfiguredClockSpeed DeviceLocator Capacity
-----
SK Hynix BANK 0 1867 ChannelA-DIMM0 4294967296
SK Hynix BANK 2 1867 ChannelB-DIMM0 4294967296

Manufacturer BankLabel ConfiguredClockSpeed DeviceLocator RAM (GB)
-----
SK Hynix BANK 0 1867 ChannelA-DIMM0 4
SK Hynix BANK 2 1867 ChannelB-DIMM0 4
```

ram\_auslesen.ps1



# Übung – Sort-Object | Select-Object

1. Melden Sie sich mit einem **Benutzer** am **Client** an um Anmeldungen in AD zu kreieren
2. Rufen Sie am **DC** mit einem **Filter** AD Benutzer ab (Beispiel unten):
  - a. User sind **aktiviert**
  - b. User **krbtgt** nicht anzeigen
  - c. Alle User die den Namen **admin** beinhalten nicht anzeigen
  - d. User **absteigend** nach dem **LastLogonDate** sortieren → **Sort-Object**
  - e. Nur **Name** und **LastLogonDate** anzeigen → **Select-Object**
  - f. Ausgabe in einer **HTML** Datei speichern → **ConvertTo-HTML** und **Out-File C:\temp\userlogons.htm**

```
1 Get-ADUser -Filter 'enabled -eq "true" -and name -notlike "*admin*"'
```

User Last Logon	
Name	LastLogonDate
patrick	1/25/2020 12:35:08 PM
admpeter	6/3/2019 11:22:31 AM
Peter	6/3/2019 11:11:11 AM
Harald Unsicher	3/4/2019 5:03:12 PM

get-aduser\_filter\_sort\_lastlogon\_select\_html.ps1

# Was macht dieses Konstrukt?

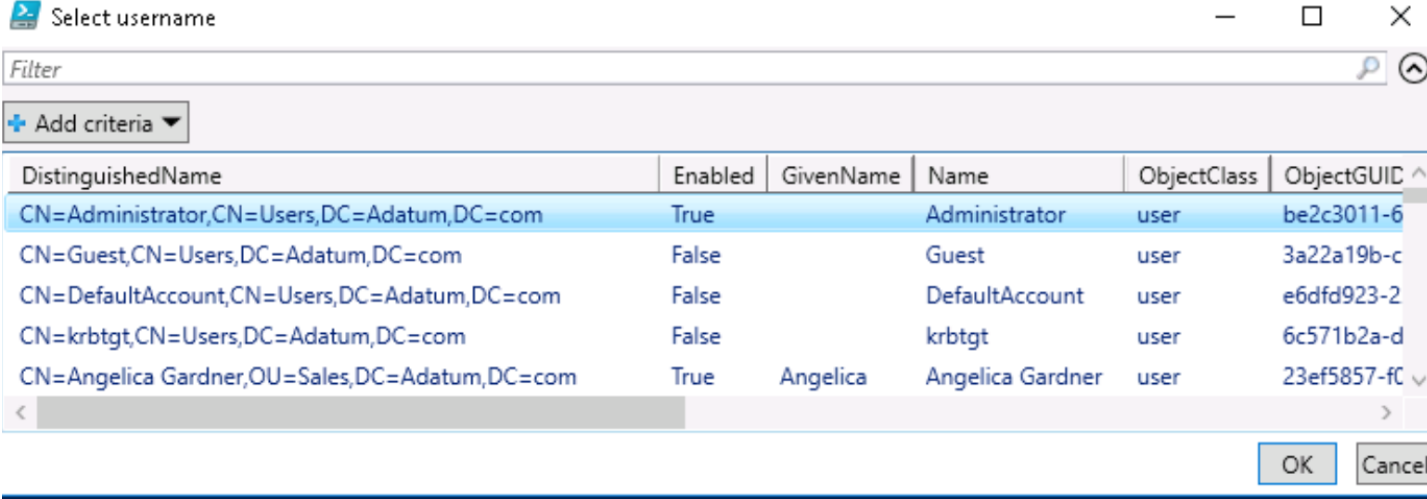
```
Unbenannt1.ps1* x
1 1..1200 | ForEach-Object {
2     1..6 | Get-Random
3 } | Group-Object | Sort-Object Count -Descending |
4 Select-Object Name,Count
5
```

Name	Count
4	212
6	210
1	205
3	202
2	195
5	176

würfel.ps1

# Übung – Out-GridView und Select-Object

- Schreiben Sie am DC ein Skript, welches
  - a. alle User abrufen und
  - b. dann den oder die mit **Out-GridView** gewählten User deaktiviert
- Werkzeuge: **Get-ADUser**, **Out-GridView**, **Set-ADUser**



Filter

+ Add criteria

DistinguishedName	Enabled	GivenName	Name	ObjectClass	ObjectGUID
CN=Administrator,CN=Users,DC=Adatum,DC=com	True		Administrator	user	be2c3011-6
CN=Guest,CN=Users,DC=Adatum,DC=com	False		Guest	user	3a22a19b-c
CN=DefaultAccount,CN=Users,DC=Adatum,DC=com	False		DefaultAccount	user	e6dfd923-2
CN=krbtgt,CN=Users,DC=Adatum,DC=com	False		krbtgt	user	6c571b2a-d
CN=Angelica Gardner,OU=Sales,DC=Adatum,DC=com	True	Angelica	Angelica Gardner	user	23ef5857-fc

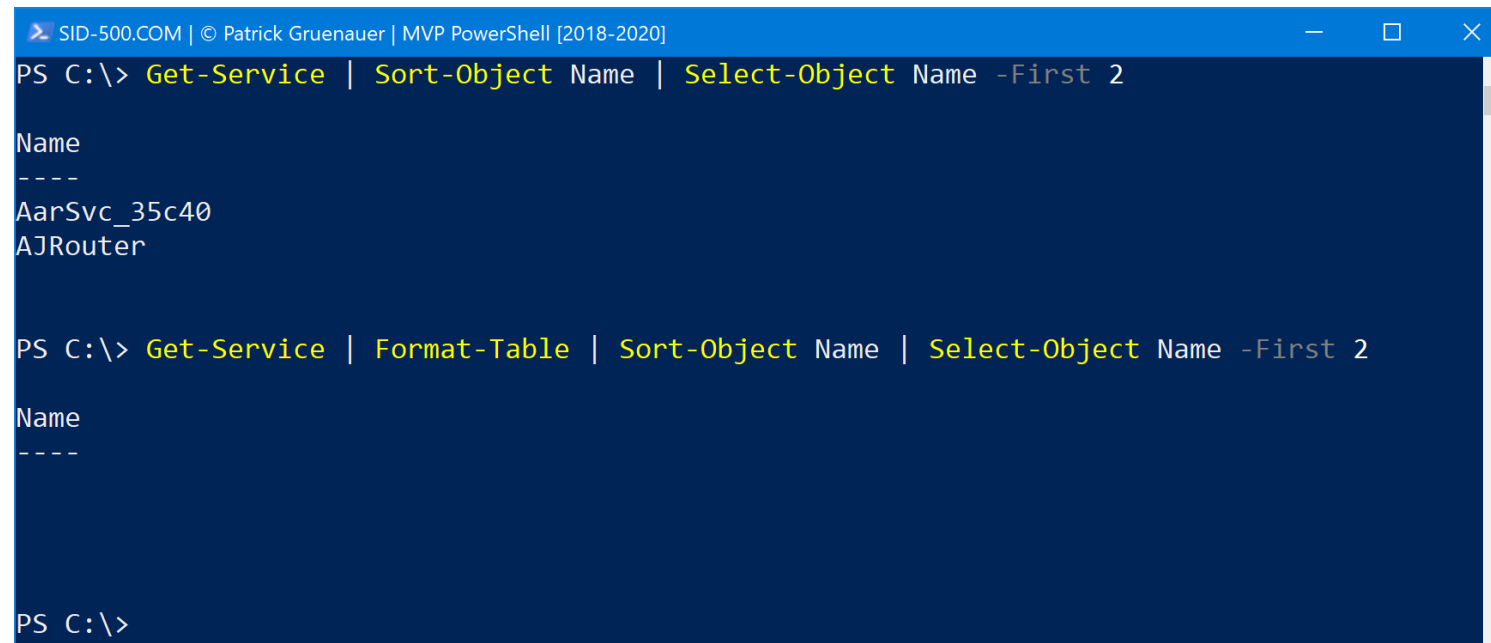
OK Cancel

get-aduser\_set-aduser\_out-gridview.ps1

# Tipps

---

1. **Sort-Object** vor **Select-Object**
2. **Select-Object -ExpandProperty** oder **().Attribut** zur Weiterverarbeitung
3. **Format-Table**, **Format-List** ... sparsam ... und wenn am Ende des Befehls



```
SID-500.COM | © Patrick Gruenauer | MVP PowerShell [2018-2020]
PS C:\> Get-Service | Sort-Object Name | Select-Object Name -First 2
Name
----
AarSvc_35c40
AJRouter

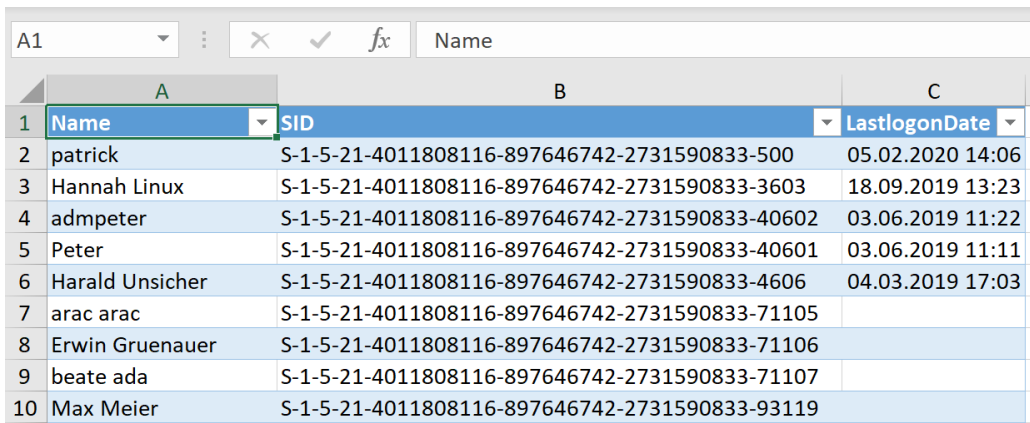
PS C:\> Get-Service | Format-Table | Sort-Object Name | Select-Object Name -First 2
Name
----

PS C:\>
```

# Übung – Export-Excel, Sort-Object, Select-Object

■ Voraussetzung: DC mit Internetzugriff

1. Installieren Sie mit **Install-Module ImportExcel** das Excel Modul
2. Ziel: Excel-Sheet der AD Benutzer sortiert nach LastLogonDate (siehe Screen)



	A	B	C
1	Name	SID	LastLogonDate
2	patrick	S-1-5-21-4011808116-897646742-2731590833-500	05.02.2020 14:06
3	Hannah Linux	S-1-5-21-4011808116-897646742-2731590833-3603	18.09.2019 13:23
4	admpeter	S-1-5-21-4011808116-897646742-2731590833-40602	03.06.2019 11:22
5	Peter	S-1-5-21-4011808116-897646742-2731590833-40601	03.06.2019 11:11
6	Harald Unsicher	S-1-5-21-4011808116-897646742-2731590833-4606	04.03.2019 17:03
7	arac arac	S-1-5-21-4011808116-897646742-2731590833-71105	
8	Erwin Gruenauer	S-1-5-21-4011808116-897646742-2731590833-71106	
9	beate ada	S-1-5-21-4011808116-897646742-2731590833-71107	
10	Max Meier	S-1-5-21-4011808116-897646742-2731590833-93119	

3. Beispiele hier: <https://dfinke.github.io/powershell/2019/07/31/Creating-beautiful-Powershell-Reports-in-Excel.html>

export-excel\_ad\_user\_lastlogon.ps1

---

# Exkurs: Parameterbindung: ByVal, ByPropertyName

# Pipeline

---

## The PowerShell Pipe

  
Get-Process notepad, mspaint | Stop-Process



**The pipe takes everything on the left of the pipe and forwards it to the command to the right of the pipe**

sid-500.com

An welchen Stop-Process Parameter wird notepad und mspaint gebunden?

# Pipeline

## The PowerShell Pipe



- Wozu sollte ich wissen, wie Objekte weitergeben werden?

```
Administrator: Windows PowerShell
PS C:\> Get-Process notepad | Stop-Process ✓
PS C:\> 'notepad' | Stop-Process
Stop-Process : Das Eingabeobjekt kann an keine Parameter des Befehls gebunden werden, da der Befehl keine Pipelineeingaben akzeptiert oder die Eingabe und deren Eigenschaften mit keinem der Parameter übereinstimmen, die Pipelineeingaben akzeptieren.
In Zeile:1 Zeichen:13
+ 'notepad' | Stop-Process
+ ~~~~~
+ CategoryInfo          : InvalidArgument: (notepad:String) [Stop-Process], ParameterBindingException
+ FullyQualifiedErrorId : InputObjectNotBound,Microsoft.PowerShell.Commands.StopProcessCommand

PS C:\>
```



```
SID-500.COM | © Patrick Gruenauer | MVP PowerShell [2018-2020]
PS C:\> 'spooler' | Stop-Service ✓
PS C:\> █
```







Get-Process notepad,mspaint | Stop-Process

# Parameterbindung

- Welche **Stop-Process** Parameter unterstützen die **Pipeline-Technik**?
- An welche Parameter können Objekte der Pipeline überhaupt gesendet werden?
  - **by Value** → einfacher Wert
  - **by PropertyName** → Objekt-Name

```
SID-500.COM | © Patrick Gruenauer | MVP PowerShell
PS C:\> Get-Help Stop-Process -Parameter name | Select pipelineInput

pipelineInput
-----
True (ByPropertyName)
```

```
SID-500.COM | © Patrick Gruenauer | MVP PowerShell [2017-2019]
PS C:\> Get-Help Stop-Process -Parameter InputObject | Select-Object pipelineInput

pipelineInput
-----
True (ByValue)

PS C:\>
```

# Parameterbindung

## The PowerShell Pipe

  
Get-Process notepad,mspaint | Stop-Process

```
SID-500.COM | © Patrick Gruenauer | MVP PowerShell
PS C:\> Get-Help Stop-Process -Parameter * | Where-Object pipelineinput -like *true*

-Id <Int32[]>
  Specifies the process IDs of the processes to stop. To specify multiple IDs, use commas to separate the IDs. To
  find the PID of a process, type `Get-Process`.

  Erforderlich?           true
  Position?               0
  Standardwert            None
  Pipelineeingaben akzeptieren? True (ByPropertyName)
  Platzhalterzeichen akzeptieren? false

-InputObject <Process[]>
  Specifies the process objects to stop. Enter a variable that contains the objects, or type a command or expression
  that gets the objects.

  Erforderlich?           true
  Position?               0
  Standardwert            None
  Pipelineeingaben akzeptieren? True (ByValue)
  Platzhalterzeichen akzeptieren? false

-Name <String[]>
  Specifies the process names of the processes to stop. You can type multiple process names, separated by commas, or
  use wildcard characters.

  Erforderlich?           true
  Position?               named
  Standardwert            None
  Pipelineeingaben akzeptieren? True (ByPropertyName)
  Platzhalterzeichen akzeptieren? false
```



Get-Process notepad,mspaint | Stop-Process

# ByValue

- **Stop-Process** bindet notepad und mspaint an den Parameter **-InputObject** mit **ByValue**. Es wird ein Objekt vom Typ **Prozess** erwartet

```
SID-500.COM | © Patrick Gruenauer | MVP PowerShell
PS C:\> Get-Help Stop-Process -Parameter InputObject

-InputObject <Process[]>
  Specifies the process objects to stop. Enter a variable that contains the objects, or type a command or expression
  that gets the objects.

  Erforderlich?           true
  Position?               0
  Standardwert            None
  Pipelineeingaben akzeptieren? True (ByValue)
  Platzhalterzeichen akzeptieren? false

PS C:\>
```

# ByValue

## The PowerShell Pipe

Get-Process notepad,mspaint | Stop-Process



- **Get-Process** ruft Prozess **Objekte** ab
- Der Wert dieses Prozesses (Value) wird an den Parameter **-InputObject** weitergereicht

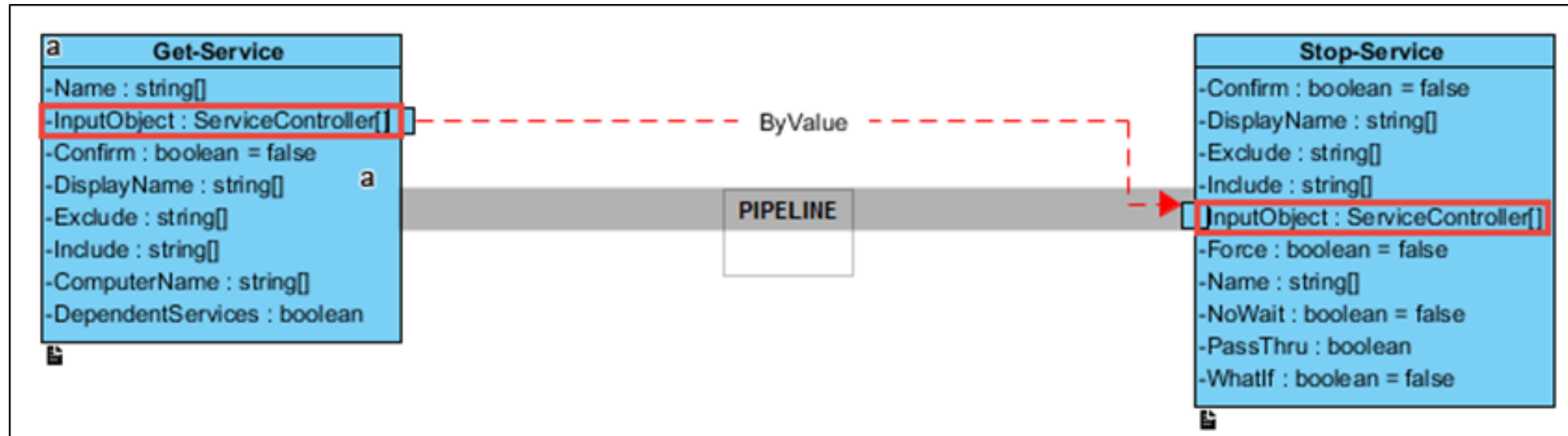
```
SID-500.COM | © Patrick Gruenauer | MVP PowerShell
PS C:\> $a=Get-Process notepad
PS C:\> $a.GetType()

IsPublic IsSerial Name          BaseType
-----
True     False    Process        System.ComponentModel.Component

PS C:\>
```



# ByValue



<https://blogs.technet.microsoft.com/askpfeplat/2016/11/21/two-ways-to-accept-pipeline-input-in-powershell/>



Get-Process notepad,mspaint | Stop-Process

# ByPropertyName

- **Get-Process notepad,mspaint | Stop-Process** übergibt die Pipeline-Objekte notepad und mspaint dem Parameter InputObject mit **ByValue**
- Der Parameter **-InputObject** erwartet ein Pipeline-Objekt vom Typ **Prozess**
- notepad und mspaint sind **Strings**

```
Unbenannt1.ps1* Unbenannt2.ps1* X
1 $a='notepad,mspaint'

PS C:\WINDOWS\system32> $a.GetType()

IsPublic IsSerial Name                                     BaseType
-----
True     True     String                                     System.Object

PS C:\WINDOWS\system32>
```



Get-Process notepad,mspaint | Stop-Process

# ByPropertyName

- ... es wird eine **ParameterBindingException**, also ein Error des Pipeline Binding ausgegeben ...
- Warum? Die beiden Werte sind **keine Prozess Objekte** ... sie sind einfache Zeichen ...

```
SID-500.COM | © Patrick Gruenauer | MVP PowerShell
PS C:\> 'notepad,mspaint' | Stop-Process
Stop-Process : Das Eingabeobjekt kann an keine Parameter des Befehls gebunden werden, da der Befehl keine
Pipelineeingaben akzeptiert oder die Eingabe und deren Eigenschaften mit keinem der Parameter übereinstimmen, die
Pipelineeingaben akzeptieren.
In Zeile:1 Zeichen:21
+ 'notepad,mspaint' | Stop-Process
+ ~~~~~
+ CategoryInfo          : InvalidArgument: (notepad,mspaint:String) [Stop-Process], ParameterBindingException
+ FullyQualifiedErrorId : InputObjectNotBound,Microsoft.PowerShell.Commands.StopProcessCommand
PS C:\>
```





Get-Process notepad,mspaint | Stop-Process

# ByPropertyName

- aber es gibt einen **Parameter**, an welchen die **String** Objekte der Pipe **gebunden** werden können ...

```
SID-500.COM | © Patrick Gruenauer | MVP PowerShell
PS C:\> Get-Help Stop-Process -Parameter Name

-Name <String[]>
    Specifies the process names of the processes to stop. You can type multiple process names, separated by commas, or
    use wildcard characters.

    Erforderlich?           true
    Position?              named
    Standardwert           None
    Pipelineeingaben akzeptieren? True (ByPropertyName)
    Platzhalterzeichen akzeptieren? false
```

- Problem: der Parameter akzeptiert zwar Pipelineinput als String, aber die weitergereichten Objekte müssen den **gleichen Objektnamen wie der Parameter Name** aufweisen = **Name**. Das tun sie aber nicht ...



# ByPropertyName

- Soll: Das Objekt muss dem Namen des Parameters entsprechen

```
Name
----
mspaint
notepad

PS C:\>
```

SID-500.COM | © Patrick Gruenauer | MVP PowerShell

```
PS C:\> Get-Help Stop-Process -Parameter Name
```

- Ist

```
SID-500.COM | © Patrick Gruenauer | MVP PowerShell
PS C:\> $a='notepad,mspaint'
PS C:\> $a
notepad,mspaint
PS C:\>
```



# ByPropertyName

- Lösung: Speichern der Strings notepad und mspaint als **Objekt** mit dem Namen **Name**

```

SID-500.COM | © Patrick Gruenauer | MVP PowerShell
PS C:\> $a=Get-Process notepad,mspaint | Select-Object Name
PS C:\> $a.GetType()

IsPublic IsSerial Name BaseType
-----
True True Object[] System.Array

SID-500.COM | © Patrick Gruenauer | MVP PowerShell
PS C:\> $a
Name
----
mspaint
notepad

PS C:\> Get-Help Stop-Process -Parameter Name

-Name <String[]>
    Specifies the process names of the processes to stop. You can type multiple process names, separated by commas, or use wildcard characters.

Erforderlich? true
Position? named
Standardwert None
Pipelineeingaben akzeptieren? True (ByPropertyName)
Platzhalterzeichen akzeptieren? false
    
```

# ByPropertyName

Get-Process notepad,mspaint | Stop-Process



Administrator: Windows PowerShell ISE

Datei Bearbeiten Ansicht Tools Debuggen Add-Ons Hilfe

Unbenannt2.ps1\* Unbenannt3.ps1\* X

```

1 $a=Get-Process notepad,mspaint | select-object Name
2
3 Trace-Command -Name ParameterBinding -Expression {$a | Stop-Process} -PSHost

```

Der Versuch an den Parameter **-InputObject** zu binden schlägt fehl.  
Grund: Kein Objekt vom Typ Process

```

DEBUG: ParameterBinding Information: 0 : MANDATORY PARAMETER CHECK on cmdlet [Stop-Process]
DEBUG: ParameterBinding Information: 0 : BIND PIPELINE object to parameters: [Stop-Process]
DEBUG: ParameterBinding Information: 0 : PIPELINE object TYPE = [Selected.System.Diagnostics.Process]
DEBUG: ParameterBinding Information: 0 : RESTORING pipeline parameter's original values
DEBUG: ParameterBinding Information: 0 : Parameter [InputObject] PIPELINE INPUT ValueFromPipeline NO COERCION
DEBUG: ParameterBinding Information: 0 : BIND arg [notepad] to parameter [InputObject]
DEBUG: ParameterBinding Information: 0 : Binding collection parameter InputObject: argument type [PSObject], parameter type [System.Diagnostics.Process[]], collection type Array, element type [System.Diagnostics.Process], no coerceElementType
DEBUG: ParameterBinding Information: 0 : Creating array with element type [System.Diagnostics.Process] and 1 elements
DEBUG: ParameterBinding Information: 0 : Argument type PSObject is not IList, treating this as scalar
DEBUG: ParameterBinding Information: 0 : BIND arg [notepad] to param [InputObject] SKIPPED
DEBUG: ParameterBinding Information: 0 : Parameter [Id] PIPELINE INPUT ValueFromPipelineByPropertyName NO COERCION
DEBUG: ParameterBinding Information: 0 : Parameter [Name] PIPELINE INPUT ValueFromPipelineByPropertyName NO COERCION
DEBUG: ParameterBinding Information: 0 : BIND arg [notepad] to parameter [Name]
DEBUG: ParameterBinding Information: 0 : Binding collection parameter Name: argument type [String], parameter type [System.String[]], collection type Array, element type [System.String], no coerceElementType
DEBUG: ParameterBinding Information: 0 : Creating array with element type [System.String] and 1 elements
DEBUG: ParameterBinding Information: 0 : Argument type String is not IList, treating this as scalar
DEBUG: ParameterBinding Information: 0 : Adding scalar element of type String to array position 0
DEBUG: ParameterBinding Information: 0 : BIND arg [System.String[]] to param [Name] SUCCESSFUL
DEBUG: ParameterBinding Information: 0 : MANDATORY PARAMETER CHECK on cmdlet [Stop-Process]
DEBUG: ParameterBinding Information: 0 : CALLING EndProcessing

```

PS C:\WINDOWS\system32>

# ByPropertyName



```

Administrator: Windows PowerShell ISE
Datei Bearbeiten Ansicht Tools Debuggen Add-Ons Hilfe
Unbenannt2.ps1* Unbenannt3.ps1* X
1 $a=Get-Process notepad, mspaint | Select-Object Name
2
3 Trace-Command -Name ParameterBinding -Expression {$a | Stop-Process} -PSHost

DEBUG: ParameterBinding Information: 0 : MANDATORY PARAMETER CHECK on cmdlet [Stop-Process]
DEBUG: ParameterBinding Information: 0 : BIND PIPELINE object to parameters: [Stop-Process]
DEBUG: ParameterBinding Information: 0 : PIPELINE object TYPE = [Selected.System.Diagnostics.Process]
DEBUG: ParameterBinding Information: 0 : RESTORING pipeline parameter's original values
DEBUG: ParameterBinding Information: 0 : Parameter [InputObject] PIPELINE INPUT ValueFromPipeline NO COERCION
DEBUG: ParameterBinding Information: 0 : BIND arg [@{Name=notepad}] to parameter [InputObject]
DEBUG: ParameterBinding Information: 0 : Binding collection parameter InputObject: argument type [PSObject], parameter type [System.Diagnostics.Process[]], collection type Array, element type [System.Diagnostics.Process], no coerceElementType
DEBUG: ParameterBinding Information: 0 : Creating array with element type [System.Diagnostics.Process] and 1 elements
DEBUG: ParameterBinding Information: 0 : Argument type PSObject is not IList, treating this as scalar
DEBUG: ParameterBinding Information: 0 : BIND arg [@{Name=notepad}] to param [InputObject] SKIPPED
DEBUG: ParameterBinding Information: 0 : Parameter [Id] PIPELINE INPUT ValueFromPipelineByPropertyName NO COERCION
DEBUG: ParameterBinding Information: 0 : Parameter [Name] PIPELINE INPUT ValueFromPipelineByPropertyName NO COERCION
DEBUG: ParameterBinding Information: 0 : BIND arg [notepad] to parameter [Name]
DEBUG: ParameterBinding Information: 0 : Binding collection parameter Name: argument type [String], parameter type [System.String[]], collection type Array, element type [System.String], no coerceElementType
DEBUG: ParameterBinding Information: 0 : Creating array with element type [System.String] and 1 elements
DEBUG: ParameterBinding Information: 0 : Argument type String is not IList, treating this as scalar
DEBUG: ParameterBinding Information: 0 : Adding scalar element of type String to array position 0
DEBUG: ParameterBinding Information: 0 : BIND arg [System.String[]] to param [Name] SUCCESSFUL
DEBUG: ParameterBinding Information: 0 : MANDATORY PARAMETER CHECK on cmdlet [Stop-Process]
DEBUG: ParameterBinding Information: 0 : CALLING EndProcessing

PS C:\WINDOWS\system32>
    
```

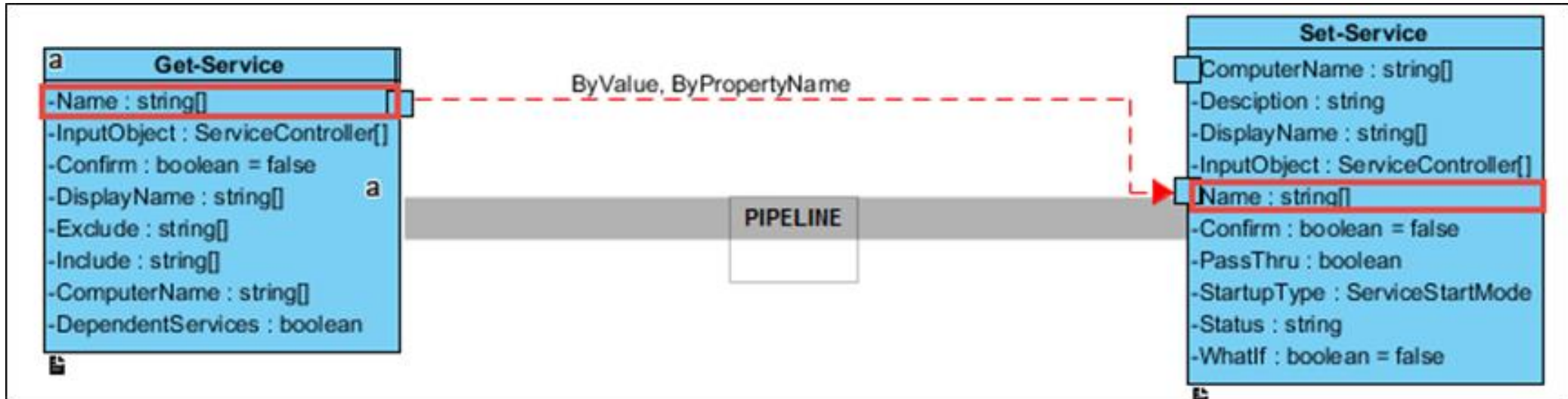
**Der Versuch an den Parameter -Name zu binden ist erfolgreich. Grund: Objekt ist ein String und Propertyname ist Name, dieser entspricht dem Parameter Name**

```

Name
----
mspaint
notepad

PS C:\>
    
```

# ByPropertyName



<https://blogs.technet.microsoft.com/askpfeplat/2016/11/21/two-ways-to-accept-pipeline-input-in-powershell/>

# Beides möglich?

```
Administrator: Windows PowerShell
PS C:\> 'spooler' | Stop-Service
PS C:\> .
```

```
Unbenannt1.ps1* X
1 Get-Help Stop-Service -Parameter * |
2 Where-Object pipelineInput -like *true*

-Name <String[]>
  Specifies the service names of the services to stop. Wildcard
  characters are allowed.

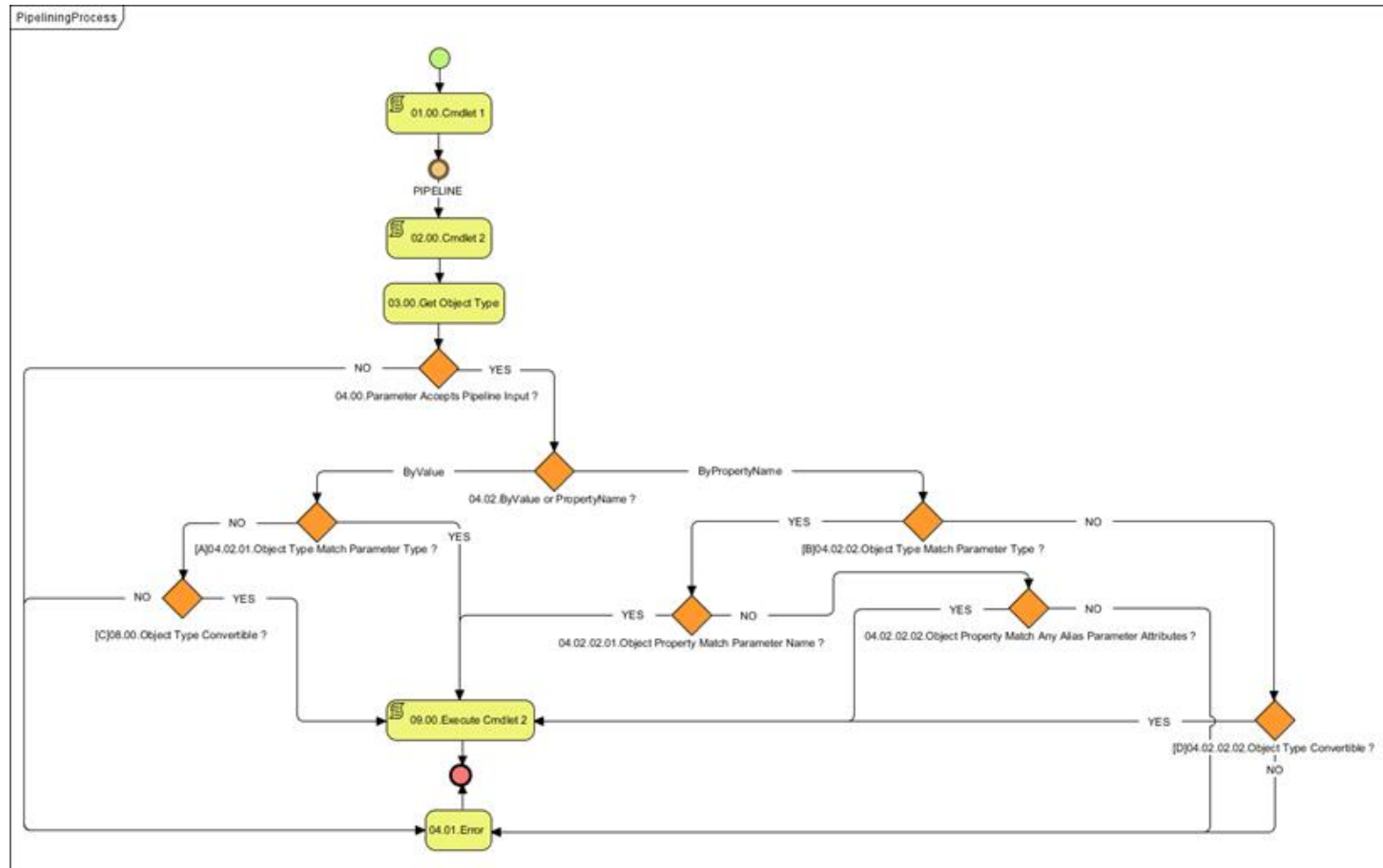
Erforderlich? true
Position? 0
Standardwert None
Pipelineeingaben akzeptieren? True (ByPropertyName, ByValue)
Platzhalterzeichen akzeptieren? false
```

When a parameter is both ByValue and ByPropertyName bound, PowerShell attempts to bind in this order:

1. Bind ByValue with no type conversion
2. Bind ByPropertyName with no type conversion
3. Bind ByValue with type conversion
4. Bind ByPropertyName with type conversion

<https://rkeithhill.wordpress.com/2008/05/09/effective-powershell-item-12-understanding-byvalue-pipeline-bound-parameters/>

# Piping by Value, by PropertyName



<https://blogs.technet.microsoft.com/askpfeplat/2016/11/21/two-ways-to-accept-pipeline-input-in-powershell/>



# Limitierung der Pipe ...

- Manchmal nützt alles nichts ... die linke Seite der Pipe passt so gar nicht zur rechten Seite
- Lösung: **Foreach-Object** oder andere Workarounds ...

```
Untitled5.ps1* X
1 Get-ChildItem -Path C:\Temp\ -Directory |
2 Foreach-Object {New-SmbShare -Name $_.Name -Path $_.Fullname -FullAccess Everyone -Description Test}
```

```
PS C:\> Get-ChildItem -Path C:\Temp\ -Directory |
Foreach-Object {New-SmbShare -Name $_.Name -Path $_.Fullname -FullAccess Everyone -Description Test}
```

Name	ScopeName	Path	Description
BH	*	C:\Temp\BH	Test
GF	*	C:\Temp\GF	Test
IT	*	C:\Temp\IT	Test

```
PS C:\>
```

<https://sid-500.com/2019/05/21/creating-multiple-smb-shares-at-once-bulk-with-powershell/>

# Übung – ByVal, ByPropertyName

1. Rufen Sie mit `$a=Get-ADUser User` einen Benutzer ab
2. Zeigen Sie mit `$a.GetType()` welcher Objekttyp zurückgegeben wird
3. Welcher `Set-ADUser` Parameter erwartet ein `ADUser` Objekt?

`Get-Help Get-ADUser -Parameter * | Where-Object pipelineinput -like *true*`

Wird `ByValue` oder `ByPropertyName` erwartet?

4. Zeigen Sie mit `Trace-Command -Name ParameterBinding -Expression {Get-ADUser h.linux | Set-ADUser -Enabled $false} -PSHost`, dass Ihr Parameter aus (3) verwendet wird

```
Untitled1.ps1* X
1 Trace-Command -Name ParameterBinding -Expression {Get-ADUser h.linux | Set-ADUser -Enabled $false} -PSHost
<
etADUserParameterSet]
DEBUG: ParameterBinding Information: 0 : BIND NAMED args to DYNAMIC parameters
DEBUG: ParameterBinding Information: 0 : BIND arg [False] to parameter [Enabled]
DEBUG: ParameterBinding Information: 0 : COERCE arg to [System.Nullable`1[System.Boolean]]
```

Vielen Dank für die Aufmerksamkeit

---