

# Powershell Masterclass

Powershell-Version herausfinden

```
Get-Host
```

Hilfe erhalten

```
Get-Help Get-Command
```

Beschreibungen zu Hilfen

Parameter in [ ] verpflichtend [-Logname] <String>

Parameter komplett in [ ] optional [-ComputerName <String[]>]

## Wichtige Befehle (Cmd-lets)

Aufbau: Verb-Nomen Parameter

CMD-Let	Beschreibung
Get-Uptime	Zeigt an, wie lange der Letzte Boot her ist
Get-ComputerInfo	Zeigt die Computerinfo an
Get-ChildItem	Verzeichnisinhalt anzeigen
Set-Location	Aktuellen Ordner festlegen Set-Location HKLM setzt die Location auf die Registry
Start-Transcript Stop-Transcript	Zum Loggen verwenden
Read-Host Write-Host	Benutzereingabe abfragen Gibt auf die Konsole aus
Start-Process	Programm öffnen Start-Process notepad
New-Item	Erstellt eine neue Datei New-Item -ItemType File -Path C: -Force
Add-Content	Fügt einer Datei Inhalt hinzu
Get-Content	Auslesen einer Datei

<code>Measure-Object</code>	Zählen gleich wie <code>.Count</code> Vorher in Klammern setzen
<code>Measure-Command</code>	Messen von Befehlen. Wie lange dauert ein Befehl. Damit kann die Zeit gemessen werden. <code>Measure-Command { Get-ADComputer -Filter 'enabled -eq \$false'   Set-ADComputer -Enabled \$true }</code>
<code>Foreach-Object</code>	Nimmt jedes Objekt entgegen
<code>Where-Object</code>	Filtert auf bestimmte Eigenschaften <code>Get-Process   Where-Object CPU -GT 10</code>

## Parameter

Option	Beschreibung
<code>-Recurse</code>	Auch für Unterordner
<code>-Confirm</code>	Fordert zur Bestätigung aus <code>-Confirm:\$false</code> Damit wird das Bestätigen umgangen
<code>-AsSecureString</code>	Verschlüsselt abspeichern
<code>-Whatif</code>	Was wäre wenn. Der Befehl wird nicht ausgeführt
<code>-Verbose</code>	Was wird passieren
<code>-PassThru</code>	Gibt nur die Werte wieder, keine Tabellen
<code>-Wrap</code>	Zeilenumbruch
<code>-AutoSize</code>	Passt die Spaltenansicht an

## Alias

`Get-Alias` zeigt alle Aliase an

## Variablen

erstellen `$a`

zu Variablen hinzufügen `$a += 2` Es wird die Variable um 2 erhöht

# Execution Policy

Get-ExecutionPolicy -List	
Restricted	Nichts erlaubt
Unrestricted	Alles erlaubt
RemoteSigned	Aus dem Internet geladene müssen signiert sein
AllSigned	Müssen auch die eigenen signiert sein
Bypass	

## Setzen

`Set-ExecutionPolicy Bypass`

## Dateien freigeben zum ausführen

`Unblock-File`

# Programmbeispiele

[hier zu finden](#)

## Profile erstellen

Darin können verschiedene Einstellungen gespeichert werden, damit Powershell immer mit den gleichen Optionen startet.

`New-Item $PROFILE -ItemType File -Force`

## Hilfe lesen können

- Aufrufen mit Get-Help und dem Befehl
- Beispiele sind besser um etwas zu verstehen
  - [Das gibt es Online](#)

# Pipeline

Was ist die Pipeline

## Piping

- Die Piping Technik ermöglicht die „**Verbindung**“ von PowerShell Code
- Die Verwendung der Pipe (Rohr, Rohrleitung) ist eine **Schlüsseltechnologie** in PowerShell
- | → **ALTGR+<>**
- Klassisch: **Get-Something | Do-Something**

```
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:\>
```

## Format Befehle

Format Befehle sollten immer am Ende eines Befehls stehen und dienen zur Ausgabe auf der Konsole

```
Get-Process | Format-Table Id,ProcessName
```

```
Get-Process | Format-List
```

## Out Befehle

Out-File #Ausgabe als Datei

Out-Printer #Ausgabe auf dem Standard-Drucker

Out-Null #Keine Ausgabe

Out-GridView #Zeigt die Ausgabe als Tabelle an

```
Get-Process | Out-File $home\process.txt
```

## Out-GridView

Mit dem folgenden Skript werden alle AD-User an GridView übergeben. Der Parameter PassThru ermöglicht es eine Auswahl zu treffen. Diese Auswahl wird dann an Disable ADAccount weiter.

```
Get-ADUser -Filter * | Out-GridView -PassThru | Disable-ADAccount -Verbose
```

## Tee-Object

Mit dem CMDlet wird das Ergebnis in der Konsole ausgegeben und als Datei gespeichert.

```
Get-ADUser -Filter * | Tee-Object -FilePath $home\adusers.txt
```

## CSV

```
Get-ADUser -Filter * | Export-Csv $home\aduserscsv.csv
```

```
Import-Csv -Path $home\aduserscsv.csv
```

## \$\_ 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
- Sie beinhalten **das aktuell von der Pipeline verarbeitete Objekt**

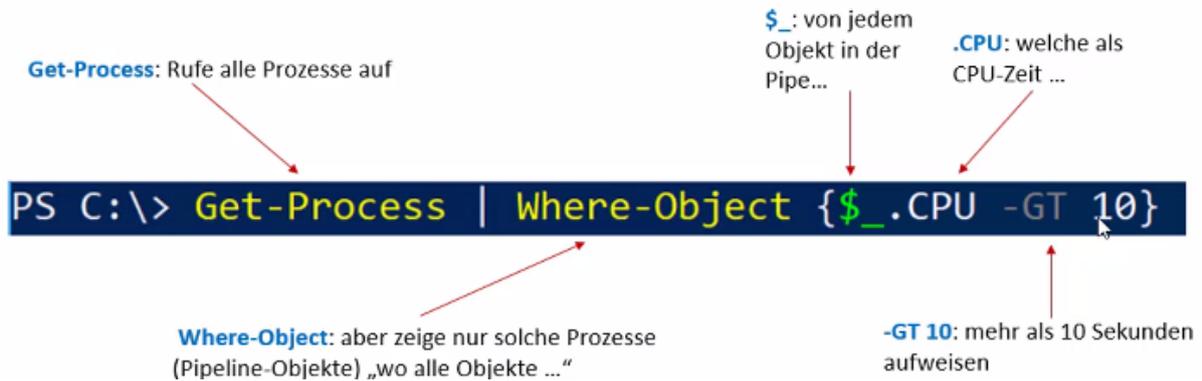
### The PowerShell Pipe

Get-Process **notepad, mspaint** | Stop-Process

`$_` 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

Damit wird in der Pipeline jede Variable in den jeweiligen Durchlauf übergeben.



Ein weiteres Beispiel in dem zwei Filter kombiniert werden

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

## Filtern

[Weitere Infos auf der Website](#)

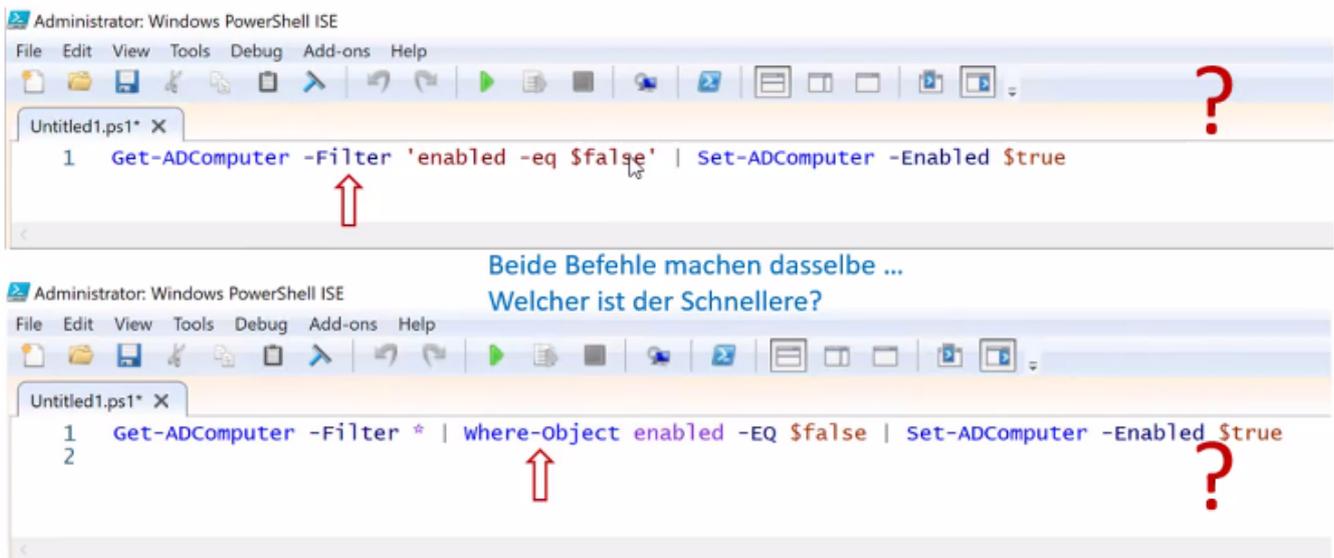
## Filtering

- Mit dem Parameter **-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
```

Der Unterschied zwischen den beiden Filtern ist, dass der obere schneller ist, weil der Filter schon vor der Pipeline ausgeführt wird.

# Filtering – Where-Object vs. Filtering



## Sort-Object und Select-Object

```
# Sort-Object
Get-Process | Sort-Object CPU -Descending

# Select-Object
Get-Process | Select-Object CPU,Id,ProcessName
```

Select-Object macht etwas ähnliches wie Format-Table nur das die Daten nicht verändert werden. Dadurch kann man mit allen Daten weiter arbeiten.

```
# Gemischt
Get-Process | Sort-Object CPU -Descending | Select-Object ProcessName,CPU -First 3
```

### Schlüsselfunktion in Powershell!

Select-Object mit `-ExpandProperty` dann wird wirklich nur der Name ausgegeben und nicht eine Tabelle die die Namen enthält.

```
$comp = Get-ADComputer -Filter * | Select-Object -ExpandProperty Name
Test-Connection -ComputerName $comp #Text-Connection kann Multiple anfragen entgegennehmen und alles in $comp anpingen
```

# Get-Member und Select-Object \*

# Gibt aus was es alles an Möglichkeiten zu einem CMD-Let gibt

Get-Process | Get-Member

# Hiermit kann man alles alles ausgeben

Get-Process | Select-Object -Property \*

---

Revision #8

Created 28 March 2023 05:05:15 by Hermann

Updated 5 April 2023 12:44:47 by Hermann