

Die Skripts zum Buch Windows PowerShell Crashkurs

(Autor: Peter Monadjemi, Verlag Microsoft Press)

Stand: 08/01/2008

Dieses Dokument enthält die Beispiele des Buches – ausgenommen sind die zahlreichen Einzeiler, die man ohnehin am besten selber eintippen sollte (Lerneffekt).

Aktueller Stand: 107 Beispiele

Dazu noch ein nützlicher Tipp (gibt es Tipps, die nicht nützlich sind?), der im Buch leider fehlt. Wenn man gleich zu Beginn einer WPS-Sitzung das Cmdlet **start-transcript** ausführt (das kann man auch in die Profile-Datei einbauen), werden alle Eingaben und vor allem deren Ergebnisse mitprotokolliert. Das ist sehr praktisch, wenn man seine „Gehversuche“ nachvollziehen möchte. Mit **stop-transcript** wird die Protokollierung wieder beendet.

Theoretisch ist es möglich, den Skriptcode in ein WPS-Fenster einzufügen und durch Drücken der [Eingabe]-Taste auszuführen. Praktischer und vor allem reproduzierbarer ist es dagegen, den Text zuerst in eine Textdatei zu kopieren, diese zu speichern und über den Eingabe des Pfades oder durch Voranstellen des Punktes auszuführen.

| |
|----------------|
| Hinweis |
|----------------|

Die meisten Beispiele wurden beim Kopieren in dieses Word-Dokument noch einmal getestet. Eine 100%tige Garantie, dass sie sofort funktionieren gibt es allerdings nicht. Beispiele, in denen „fest verdrahtete“ Servernamen vorkommen müssen ohnehin entsprechend angepasst werden. Bei allen Fragen oder Problemen mit den Skripts bitte an mich unter pm@activetraining.de wenden.

Kapitel 1: Eine erste Rundreise durch die Windows PowerShell

In diesem Kapitel geht es lediglich um eine erste Einführung in die Windows PowerShell. Es gibt daher keine Skripts und umfangreicheren Beispiele.

Kapitel 2: Die Windows PowerShell im Überblick

In diesem Kapitel geht es um die ersten Schritte mit Cmdlets. Es gibt daher ebenfalls keine Skripts und umfangreicheren Beispiele.

Kapitel 3: Umgang mit Dateien und Verzeichnissen

Beispiel:

Die größte Datei in einem Verzeichnis finden

```
# Größte Datei in einem Verzeichnis
$BildDir = $env:userprofile + "\Eigene Dateien\Eigene Bilder"
$BildListe = Get-ChildItem $BildDir | Where-Object
{!(($_.PSISContainer))} | Sort-Object Length -Descending
$BildListe[0]
```

Beispiel:

Große Dateien, die in einem Zeitabschnitt angelegt wurden finden

```
# Große Dateien, die in einem Zeitabschnitt angelegt wurden
Set-Location "$env:userprofile\Eigene Dateien"
Get-Childitem -Recurse | Where-Object {$_.creationtime -gt (get-
date).adddays(-30) -and $_.length -gt 1MB} | Sort-Object
CreationTime
```

Beispiel:

Dateien kopieren und die Anzahl der kopierten Dateien ausgeben

```
$Anzahl = 0
Select-String -path *.log -list Error | foreach-object { Copy-Item
-Path $_.Path -Destination ErrorLogs ; $Anzahl++ }
"$Anzahl Dateien wurden kopiert..."
```

Beispiel:

Alle Mp3-Dateien, die kleiner als 5 Mbyte sind in ein Verzeichnis kopieren

```
# Kopiert Mp3-Dateien kleiner 5 MByte in ein neues Verzeichnis
Set-Location "$env:userprofile\Eigene Dateien\Eigene Musik"
$DateiListe = Get-Childitem "*.mp3"
$Mp3Dir = (Resolve-Path "$env:userprofile\Eigene Dateien\Eigene
Musik\mp3" -ea silentlycontinue)
write-host $DateiListe.Count "Dateien gefunden"
if ($Mp3Dir -eq $Null)
{
    New-Item "$env:userprofile\Eigene Dateien\Eigene Musik\mp3" -
Type Directory
    # Meldung nur zur Kontrolle
    write-host "Mp3-Unterverzeichnis wurde angelegt"
}

$DateiListe | % {
    If ($_.Length -lt 5E6)
    {
        Copy-Item $_ -destination mp3
        $Anzahl +=1
    }
}
write-host $Anzahl "Mp3-Dateien kopiert..."
```

Beispiel:

IP-Adressen aus einem Webserver-Log extrahieren

```
# Weblog nach IP-Adressen durchsuchen und IP-Adressen ausgeben
Clear-Host
$WebLogPfad = "C:\wpstest\Logs\Weblog.log"
$Muster = "[0-9]{2,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}"
$DateiInhalt = Get-Content $WebLogPfad
$m = [regex]::match($DateiInhalt, $Muster)
write-Host "Gefundene IP-Adressen:"
while ($m.Success)
{
    $m.value
    $m = $m.NextMatch()
}
```

Kapitel 4: Die Formatierung der Ausgabe und Export nach CSV, HTML und XML**Beispiel:**

Startzeit und Laufzeit von Prozessen ausgeben

```
# Startzeit und Laufzeit von Prozessen ausgeben
Get-Process | Format-Table @{Expression="Name";Width=20},
@{Label="Startzeit";Expression="StartTime";Format="d"},
@{Label="CPU-Zeit
Total";Expression={$_.TotalProcessorTime.TotalSeconds}} -auto
```

Beispiel:

Aktuelle RSS-Inhalte des WPS-Teamblogs anzeigen

```
#Aktuelle RSS-Inhalte des WPS-Teamblogs anzeigen
$rss = [xml] (New-Object
net.webclient).DownloadString("http://blogs.msdn.com/powershell/rs
s.xml")
$rss.rss.channel.item | Select-Object title
```

Kapitel 5: Praxisalltag mit der Windows PowerShell**Beispiel:**

Auf einen gestarteten Prozess warten

```
#Auf einen gestarteten Prozess warten
$P = [Diagnostics.Process]::Start($env:comspec, "/c ping
192.168.1.100")
$P.WaitForExit()
"Beendet mit Rückgabewert $($P.ExitCode)"
```

Beispiel:

Zu jedem Prozess das Verzeichnis auflisten, aus dem es gestartet wurde

```
#Zu jedem Prozess das Verzeichnis auflisten, aus dem es gestartet
wurde
```

```
Get-Process | Foreach-Object { if ($_.MainModule.FileName -ne $Null) {"Prozess $(($_.Name) ) startet aus $(($_.MainModule.FileName))" } } | Get-Unique
```

Beispiel:

Zu allen Diensten die abhängigen Dienste auflisten

```
#Zu allen Diensten die abhängigen Dienste auflisten
(Get-Service -displayname "Windows-Verwaltungsinstrumentation").DependentServices | Foreach-Object {
    $_.DisplayName }
```

Beispiel:

Die Einträge des heutigen Tages aus dem Systemprotokoll abfragen

```
#Die Einträge des heutigen Tages aus dem Systemprotokoll abfragen
Get-EventLog -logname "Windows PowerShell" | Where-Object
{($_.EntryType -eq "Information") -AND
($_.TimeWritten.ToString("d") -eq (Get-Date).ToString("d"))}
```

Beispiel:

Einträge und Werte eines Schlüssels ohne die PS-Properties auflisten

```
# -----
# Auflisten aller Einträge und deren Werte eines Schlüssels
# -----
Push-Location
cd HKLM:\Software\microsoft\windows\currentversion\run
$RunValues = (Get-ItemProperty -path . | Select-String
.).ToString().Split(";")
$A = $RunValues.Length-1
# Den ersten Eintrag anders behandeln wegen der geschweiften
Klammer
$Eintrag = $RunValues[0].Substring(2).Split('=')[0]
$Wert = $RunValues[0].Substring(1).Split('=')[1]
$Wert = $Wert.SubString(0, $Wert.Length-1)
"Eintrag: $Eintrag Wert: $Wert"
# Gibt es noch weitere Einträge?
if ($A -ge 1) {
    $RunValues = $RunValues[1..$($A)]
    $RunValues | foreach-object { "Eintrag:
$(($_.Substring(1).Split('=')[0]) Wert:
$(($_.Substring(1).Split('=')[1]))" }
}
Pop-Location
```

Beispiel:

Autostart-Einträge im Run-Schlüssel von HKLM auflisten

```
# Autostart-Einträge auflisten
Push-Location
cd hkml:\software\microsoft\windows\currentversion\run
Get-ItemProperty . > c:\Autostart.txt
Get-Content C:\Autostart.txt
Pop-Location
```

Beispiel:

Die Namen der installierten Anwendungen auflisten

```
# Listet die Namen aller installierten Anwendungen auf
$KeyPfad = "Software\Microsoft\Windows\CurrentVersion\Uninstall"
$Key =
[Microsoft.Win32.Registry]::LocalMachine.OpenSubKey($KeyPfad)
$Keys = ($Key).GetSubKeyNames()
$Anzahl = 0
foreach ($K In $Keys) {
    $SubKeyPfad = $KeyPfad + "\" + $K
    $SubKey =
[Microsoft.Win32.Registry]::LocalMachine.OpenSubKey($SubKeyPfad)
    ($SubKey).GetValue("DisplayName")
    $Anzahl++
}
$Key.Close()
write-host "$Anzahl Anwendungen gefunden..."
```

Beispiel:

Remote-Zugriff auf die Registry

Hinweis:

Der Servername muss angepasst werden, sonst wird der lokale Computer verwendet. Das Skript enthält im Buch zudem einen kleinen Fehler, bei **write-host** "Subkey:" fehlte ein Anführungszeichen am Ende.

```
# -----
# Beispiel für einen Remote-Registry-Zugriff
# -----
#$Server = "<Server>"
$Server = $Env:Computername
$Key = "SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkCards"
$Typ = [Microsoft.Win32.RegistryHive]::LocalMachine
$RegKey = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey($Typ,
$Server)
$RegKey = $RegKey.OpenSubKey($Key)
Write-Host "Subkeys:"
Write-Host "-----"
Foreach($Sub In $RegKey.GetSubKeyNames())
{
    $NICPools = "SOFTWARE\Microsoft\Windows
NT\CurrentVersion\NetworkCards\$sub"
    $RegKey = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey($Typ,
$Server)
    $RegKey = $RegKey.OpenSubKey($NICPools)
    Foreach($Wert In $RegKey.GetValueNames())
    {
        if ( $Wert -eq "Servicename")
        {
            $KeyValue= $RegKey.GetValue("$Wert")
            $Keyvalue
        }
    }
}
}
```

Kapitel 6: Die Windows PowerShell einrichten

Beispiel:

Warten auf eine Taste (Pause-Kommando)

```
#Warten auf eine Taste (Pause-Kommando)
function Pause ($Message="Weiter mit Taste...")
{
    Write-Host -NoNewLine $Message
    $Taste = $Host.UI.RawUI.ReadKey("NoEcho, IncludeKeyDown")
    Write-Host ""
    Return $Taste
}
```

Beispiel:

Optisch ansprechender Prompt

```
#Optisch ansprechender Prompt
function Prompt
{
    $Id = 1
    $HistoryItem = Get-History -Count 1
    if($HistoryItem)
    {
        $Id = $HistoryItem.Id + 1
    }
    Write-Host -ForegroundColor DarkGray "`n[$(Get-Location)]"
    Write-Host -NoNewLine "PS:$Id > "
    $Host.UI.RawUI.WindowTitle = "$(Get-Location)"
    "`b"
}
```

Beispiel:

Ein farbiger Prompt

```
# Ein farbiger Prompt
function Prompt
{
    $Motto = "Keep cool", "Stay on the scene", "Lern WPS"
    $Farben = "red", "green", "blue", "yellow", "white", "cyan"
    $R = new-object random
    $PText = $(Get-Location).ToString()
    $PText = "Das Motto des Tages: "
    Write-Host $PText -NoNewline
    $PText = $Motto[$R.next(1, $Motto.Length)]
    for ($i = 0;$i -lt $PText.Length;$i++)
    {
        $F = [ConsoleColor]$R.next(1,16)
        $F = $Farben[$i % $Farben.Length]
        Write-Host ($PText[$i]) -nonewline -foregroundcolor $F
    }
    Write-Host ">"
}
```

Kapitel 7: WMI

Beispiel:

Hotfixes auflisten

```
# -----  
# Eine kleine WMI-Abfrage mit der WPS  
# -----  
$Instanzen = Get-Wmiobject -class Win32_QuickFixEngineering  
ForEach ($Inst In $Instanzen)  
{  
    write-host "Computer:" $Inst.CsName`n  
    write-host "Beschreibung:" $Inst.Description`n  
    write-host "HotFixID:" $Inst.HotfixID`n  
}
```

Beispiel:

Hotfixes erneut auflisten (kürzere Variante)

```
# Hotfixes auflisten  
Get-Wmiobject Win32_QuickfixEngineering | Where-Object  
{$_.Description -ne ''} | Sort-Object FixID | Format-List  
Description, HotFixID
```

Beispiel:

Hotfix-Aufstellung in HTML exportieren

```
#Hotfix-Aufstellung in HTML exportieren  
Get-Wmiobject Win32_QuickfixEngineering | select-object  
Description, HotFixID | where-object {$_.Description -ne ''} |  
convertto-html >HotfixListe.html
```

Beispiel:

Datenträgerkontingente für ein Laufwerk auflisten

```
# Alle Datenträgerkontingente eines Laufwerks auflisten  
  
$Kontingente = Get-Wmiobject -Query "References Of  
{Win32_LogicalDisk.DeviceID='C:'} Where  
ResultClass=Win32_DiskQuota"  
"Kontingente auf Laufwerk C:"  
$Kontingente | Format-Table User, DiskSpaceUsed, @{  
Label="Limit";Expression={$_.Limit.ToString("n0")}}  
-AutoSize
```

Beispiel:

Alle Einträge in einer Programmgruppe auflisten

```
# Einträge in einer Programmgruppe auflisten  
Get-Wmiobject -query "Associators Of  
{Win32_LogicalProgramGroup.Name='All  
Users:Startmenü\Programme\Windows PowerShell 1.0'} Where  
ResultClass=Win32_LogicalProgramGroupItem" | Format-Table Name
```

Beispiel:

Die Namen aller Netzwerkadapter auflisten

```
# Auflisten aller Netzwerkadapter
"Liste der Netzwerkadapter:"
$Liste = Get-Wmiobject -Class Win32_NetworkAdapter
$Liste | Format-Table Servicename
"Anzahl: " + $Liste.Count
```

Beispiel:

IP-Adressen aller Netzwerkadapter auflisten

```
# Auflisten aller IP-Adressen
Get-Wmiobject -Class Win32_NetworkAdapterConfiguration | `
  foreach-object { if ($_.DHCPEnabled -eq $True -and $_.IPAddress -
ne $Null) `
    {$_ .Description + "-" + $_.IPAddress[0] }
}
```

Beispiel:

IP-Adressen aller Netzwerkadapter erneut auflisten aber mit kürzerer Beschreibung

```
# Auflisten aller IP-Adressen
Get-Wmiobject -Class Win32_NetworkAdapterConfiguration | `
  foreach-object { if ($_.DHCPEnabled -eq $True -and $_.IPAddress -
ne $Null) `
    { if ($_.Description.IndexOf("-") -ge 0)
      {
        $Name = $_.Description.SubString(0, $_.Description.IndexOf("-")
) )
      }
      else
      {
        $Name = $_.Description.SubString(0, 48)
      }
      $Name + "1. IP-Adresse: " + $_.IPAddress[0]
    }
  }
```

Beispiel:

Alle Partitionen auf einem Laufwerk ausgeben

```
# Alle Partitionen auf einem Laufwerk ausgeben
$Partitionen = Get-Wmiobject -query "Associators Of
{Win32_LogicalDisk.DeviceID='C:'} Where
ResultClass=CIM_DiskPartition"
"Partitionen auf Laufwerk C:"
new-object string ("=", 72)
$Partitionen | Format-Table Name, Bootable, PrimaryPartition, Size
-AutoSize
```


Beispiel:

Partitioneigenschaften ausgeben

```
#Partitioneigenschaften ausgeben
$Partitionen | Format-Table Name,
@{Label="Groesse";Expression={$_.Size};Format="N0";Alignment="Center"},
@{Label="Bootfaehig";Expression={$_.Bootable};Alignment="Center"},
@{Label="Primaer";Alignment="center";Expression={$_.PrimaryPartition}} -AutoSize
```

Beispiel:

Volumenbezeichnung ändern

```
# Volumenamen für ein Laufwerk ändern
$Laufwerk = Get-Wmiobject -Query "Select * From Win32_LogicalDisk
Where DeviceID='C:'"
if ($Laufwerk -eq $Null)
{
    "Laufwerk existiert ja gar nicht (scheinbar)."
```

```
}
```

```
else
{
    $VolumeNameNeu = Read-Host -Prompt "Neuer Volume-Name?"
    if ($VolumeNameNeu -ne $Null)
    {
        $Laufwerk.VolumeName = $VolumeNameNeu
        $Laufwerk.Put()
        "Volume-Name wurde auf $VolumeNameNeu geändert."
```

```
}
```

```
else
{
    "Volume-Name wurde nicht geändert."
```

```
}
```

```
}
```

Beispiel:

Auf WMI-Eigenschaften direkt zugreifen

```
# Direkter Zugriff auf WMI-Properties
$WMIItems = Get-Wmiobject Win32_ComputerSystem
$Props = $WMIItems | Get-Member -membertype Property
Clear-Host
"Die Eigenschaften von Win32_ComputerSystem und ihre Werte:"
for ($i=0; $i -lt $Props.Count; $i++)
{
    if ($Props[$i].Name.Substring(0,2) -ne "__")
    {
        write-host $Props[$i].Name ": " $WMIItems.($Props[$i].Name)
    }
}
```

Beispiel:

IP-Adressen und MAC-Adressen ausgeben

```
#IP-Adressen und MAC-Adressen ausgeben
$Items = Get-Wmiobject -class "Win32_NetworkAdapterConfiguration"
-filter "IPEnabled = true"
foreach ($Item in $Items)
{
    write-host "MAC-Adresse: " $Item.MACAddress
    write-host "IP-Adresse: " $Item.IPAddress
    write-host "-----"
}
```

Beispiel:

Reagieren auf das Anlegen eines neuen Prozesses

```
# Ein Beispiel für das Warten auf einen WMI-Event
$WMI = New-Object Management.EventQuery
$WMI.QueryString = "SELECT * FROM __InstanceCreationEvent WITHIN 5
WHERE TargetInstance ISA 'Win32_Process'"
$Watch = New-Object Management.ManagementEventWatcher $WMI
"Warte auf das Anlegen eines Prozesses...\n"
$Ergebnis = $Watch.WaitForNextEvent()
"Ein Prozess wurde gestartet..."
$Ergebnis.TargetInstance | fl Name, ProcessID
```

Beispiel:

Notebook-Akku-Check

```
# Prüft, ein Dell-Notebook einen "gefährdeten" Akku enthält
# Beispiel: X5875
$Wc = New-Object net.webclient
$Global:BatSeite =
$wc.downloadstring("https://www.dellbatteryprogram.com/")
$R = [regex]"&nbsp;([A-Z0-9][A-Z0-9][0-9][0-9][0-9])"
$Matches = $R.matches($Global:BatSeite)
Write-Host "Anzahl potenziell gefaehrlicher Akkutypen:
 $($Matches.Count) "
$Global:BatterieListe = $Matches | foreach {
    $_.Groups[1].Captures[0].value
}
# Notebook kann auch mehrere Akkus enthalten
#$BatterieName = Get-WmiObject -Class Win32_Battery | Select-
Object Name
Get-WmiObject -Class Win32_Battery | ForEach {
    $BatName = $_.Name
    $BatName
    break
    # Nur zu Testzwecken
    # $BatName = "X5875"
    Write-Host "Akkumodell: $BatName"
# Kommt Akkutyp ueberhaupt vor?
if ((select-string -pattern $BatName -InputObject $BatterieListe
-Quiet) -eq $True)
{
    $BatterieListe | Where {$BatName -match $_} | ForEach {
```

```
Write-Warning "Betroffen: Akku $Name - Uebereinstimmungen:
*$_*`n"
}
else
{ "Akkutyp nicht in der Liste enthalten" }
}
```

Beispiel:

Prozess starten und gleich wieder terminieren

```
# Beispiel für [WMI]
[.Diagnostics.Process]::Start("Calc.exe")
$ProzHandle = (Get-Process calc).ID
$Proz = [WMI]"Win32_Process.Handle=$ProzHandle"
$Proz.Terminate()
```

Kapitel 8: Windows PowerShell-Skripts

Beispiel:

Summe der Dateigrößen ausgeben

```
#Summe der Dateigrößen ausgeben
function Add-Filesize()
{
    $Ext = $Args[0]
    $Summe = 0
    Get-ChildItem -filter $Ext | foreach { $Summe += $_.Length}
    "$Summe Bytes"
}
```

Beispiel:

Primzahlen ausgeben

```
# Die Primzahlen von 3 bis 100
$noPrim=0
for ($i = 3; $i -le 100; $i+=2 )
{
    for ($j = 3; $j -le [Math]::Sqrt($i); $j+=2 )
    {
        if (($i % $j) -eq 0)
        {
            $noPrim=-1
            break
        }
    }
    if ($noPrim -eq 0)
    {
        $AnzahlPrims +=1
        Write-Host $i " " -NoNewline
    }
    $noPrim = 0
}
Write-Host $AnzahlPrims "Prims gefunden"
```

Beispiel:

ServicePack-Check

```
# Prüfen, ob SPs installiert sind
Get-Content Computerliste.txt | Foreach-Object {
    if ((get-wmiobject Win32_OperatingSystem -Computer `
        $_).ServicePackMajorversion -gt 0)
    {
        "Service-Pack ist auf Computer $_ installiert"
    }
}
```

Beispiel:

Beispiel für if und else

```
# Beispiel für if und else
$Zahl = Read-Host -prompt "Bitte eine Zahl eingeben"
if ($Zahl % 2 -eq 0)
{
    Write-Host Die Zahl $Zahl ist gerade
}
else
{
    Write-Host Die Zahl $Zahl ist ungerade
}
```

Beispiel:

Build-Nummer-Check mit if und else

```
#Build-Nummer-Check mit if und else
if ((get-wmiobject win32_operatingsystem).BuildNumber -eq 2600)
{
    Write-Host Die Buildnummer stimmt
}
else
{
    Write-Host Die Buildnummer stimmt nicht.
}
```

Beispiel:

Log-Einträge zählen mit if

```
#Log-Einträge zählen mit if
if ((Get-ChildItem "C:\Windows\*.log").Count -gt 100)
{
    Write-Host Zu viele Log-Dateien
}
```

Beispiel:

Beispiel für elseif

```
#Beispiel für elseif
$AnzahlUser = 7
```

```
if ($AnzahlUser -le 4)
{
    Write-Host Geringe Auslastung
}
elseif ($AnzahlUser -lt 8)
{
    Write-Host Mittlere Auslastung
}
else
{
    Write-Host Hohe Auslastung
}
```

Beispiel:

Beispiel für switch

```
# Beispiel für switch
$WarnLevel = Read-Host ("Warnlevel (0-3)?")
switch ($WarnLevel)
{
    0 { Write-Host "Kein Anlass zur Besorgnis."}
    1 { Write-Host "Noch alles im gruenen Bereich."}
    2 { Write-Host "Es wird ernst."}
    3 { Write-Host "Nichts wie weg."}
    default { Write-Host "Es liegen uns keine Angaben vor."}
}
```

Beispiel:

Beispiel für switch (2)

```
# switch in Kombination mit Vergleichsoperationen
$AnzahlUser = 9
switch ($AnzahlUser)
{
    {$_ -le 4} { Write-Host Geringe Auslastung}
    {$_ -gt 4 -and $_ -lt 8} { Write-Host Mittlere Auslastung}
    {$_ -ge 8} { Write-Host Hohe Auslastung}
}
```

Beispiel:

Beispiel für break

```
# Break am Ende eines Switch-Zweiges
$AnzahlUser = 4
switch ($AnzahlUser)
{
    {$_ -le 4} { Write-Host Geringe Auslastung;break}
    {$_ -ge 4 -and $_ -lt 8} { Write-Host Mittlere Auslastung}
}
```

Beispiel:

Beispiel für for

```
# Einfaches for-Beispiel
for ($n=0;$n -le 10;$n+=1)
{
```

```
Write-Host $n
}
```

Beispiel:

Fakultätsberechnung

```
# Eine Fakultäts-Variante
function factorial {
    1..[int]$args[0] | foreach { $f = 1 } { $f *= $_ } { $f }
}
```

Beispiel:

Beispiel für for

```
# For-Beispiel - Berechnung der Server-Auslastung
$FestplattenKapazitaet = Read-Host -Prompt
"Festplattenkapazitaet?"
[int]$AnzahlJahre = Read-Host "Anzahl Jahre? (1-10)"
$Kapazitaet = 1000
if ($AnzahlJahre -le 0 -Or $AnzahlJahre -gt 10)
{
    Write-Host "Anzahl Jahre muss zwischen 1 und 10 liegen!"
    break
}
for ($Jahre=0;$Jahre -le $AnzahlJahre;$Jahre+=1)
{
    $Farbig = !$Farbig
    if ($Farbig)
    { Write-Host -ForegroundColor "blue" "Festplattenbelegung nach
$Jahre Jahren = $Kapazitaet" }
    else
    { Write-Host -ForegroundColor "red" "Festplattenbelegung nach
$Jahre Jahren = $Kapazitaet" }

    $Kapazitaet *= [math]::pow(5,2)
}
```

Beispiel:

Beispiel für while

```
# while-Beispiel
while ((Read-Host -Prompt "Noch eine Runde?") -eq "J")
{
    Write-Host "Ihre Glueckszahl ist: " (New-Object
random).Next(1,50)
}
```

Beispiel:

Beispiel für do

```
# do-Beispiel
do
{
    Write-Host "Ihre Glueckszahl ist: " (New-Object
random).Next(1,50)
}
```

```
}  
while ((Read-Host -Prompt "Noch eine Runde?") -eq "J")
```

Beispiel:

Beispiel für foreach

```
# foreach-Beispiel  
$Datei = Get-Content C:\boot.ini -Force  
# Zeilenweise ausgeben  
foreach ($D in $Datei)  
{  
    $Z+=1  
    Write-Host Zeile $Z ":" $D  
}
```

Beispiel:

Beispiel für ein Array

```
# Array mit Objekten per For Each-Alias durchlaufen  
@((Get-Item "C:\Datei1.txt"), (Get-Item "C:\Datei2.txt"), (Get-  
Item "C:\Datei3.txt")) | % { $_.LastWriteTime }
```

Beispiel:

Beispiel für eine Hashtable

```
# Beispiele für eine Hashtable  
$Userliste = @{"AntonH"="antonh@mail.de";  
"RenateR"="renater@mail.de";  
"PETERM"="peterm@mail.de"  
}  
  
Write-Host -ForegroundColor Yellow $Userliste.Count "Namen in der  
Liste"  
# Erst die Werte ausgeben  
Write-Host -ForegroundColor Yellow Die Werte...  
foreach ($N In $Userliste.Values)  
{  
    Write-Host $N  
}  
# Und jetzt die Schlüssel  
Write-Host -ForegroundColor red Die Schlüssel...  
foreach ($N In $Userliste.Keys)  
{  
    Write-Host $N  
}
```

Beispiel:

Einfaches Beispiel für eine Funktion

```
# Die erste Funktion  
function HalloWelt  
{  
    $Zeit = Get-Date -Format "t"  
    "Hallo, Welt - es ist jetzt $Zeit"  
}
```

Beispiel:

Backup von Textdateien

```
# Backup von Textdateien, deren erste Zeile #backup lautet
function BackupTextFiles
{
    dir *.txt | foreach {
        if ((Get-Content $_ -totalCount 1) -eq "#backup")
        {
            copy $_ C:\Backups
        }
    }
}
```

Beispiel:

Rückgabe der in einem Verzeichnis zuletzt geänderten Datei

```
#Rückgabe der in einem Verzeichnis zuletzt geänderten Datei
function LastFile()
{
    Set-Location $Args[0]
    return Get-ChildItem | where {!($_.PSIsContainer)} | sort
LastwriteTime -descending | select -first 1
}
```

Beispiel:

Funktion mit einer variablen Zahl an Parametern

```
# Funktion mit variabler Zahl an Parametern
function Mittelwert
{
    $Args | % { $Summe += $_ }
    $Summe / $Args.Length
}
Mittelwert 10, 20, 30, 44
```

Beispiel:

Funktionsparameter mit einem Typ

```
#Funktionsparameter mit einem Typ
function LastFileEx2()
{
    param([string]$Dir)
    Set-Location $Dir
    return Get-ChildItem | where {!($_.PSIsContainer)} | sort
LastwriteTime -descending | select -first 1
}
```

Beispiel:

Funktion mit Return-Befehl

```
# Fakultäts-Funktion
```



```
function fak
{
    $Wert = 1
    for ($Fak=$Args[0];$Fak -gt 1;$Fak-=1)
    {
        $Wert *= $Fak
    }
    return $Wert
}
```

Beispiel:

Funktion mit Switch-Parameter

```
# Beispiel für switch-Parameter
function OSVersion
(
    [switch] $all
)
{
    $OS = Get-WmiObject -class Win32_OperatingSystem
    if ($all)
    {
        "Build-Nummer: " + $OS.BuildNumber
        "Seriennummer: " + $OS.SerialNumber
        "Service-Pack: " + $OS.ServicePackMajorVersion + "." +
        $OS.ServicePackMinorVersion
    }
    "Version: " + $OS.BuildNumber
}
```

Beispiel:

Funktion mit Switch-Parameter (2)

```
# switch-Parameter mit Wert
function Welcome
(
    [switch] $lang, [String]$Sprache="dt"
)
{
    "*****"
    "WPS-Version " + $Host.Version
    "*****"
    if ($lang)
    {
        switch ($Sprache)
        {
            "dt" { "Willkommen" }
            "en" { "Hi, there" }
            "fr" { "Bon your" }
        }
    }
}
```

Beispiel:

Datumsvergleich

```
#Datumsvergleich
function IstHeute
```

```
( [DateTime] $Datum )
{
    return (Get-Date).Date -eq $Datum.Date
}
```

Beispiel:

Datumsvergleich (2)

```
#Datumsvergleich
function IstHeute
( $Datum )
{
    $De = New-Object Globalization.cultureinfo("de-DE")
    $Datum = [DateTime]::Parse($Datum, $De)
    return (Get-Date).Date -eq $Datum.Date
}
```

Beispiel:

Feststellen, ob ein Datum in einem Bereich liegt

```
#Feststellen, ob ein Datum in einem Bereich liegt
function LiegtImBereich
($Datum, $Tage)
{
    $Datum = [DateTime]::Parse($Datum)
    return (Get-Date).AddDays(-$Tage) -le $Datum
}
```

Beispiel:

Filter-Beispiel

```
# Filter-Beispiel
filter Quadrat
{
    $_ * $_
}
```

Beispiel:

Primzahlen über Filter bestimmen

```
# -----
# Eine kurze Version der Primzahlen
# -----
filter test-prime
{
    $GrenzWert = ($_ / 2) + 1
    for ($i=2;$i -lt $GrenzWert;$i++)
    {
        if (($_ % $i) -eq 0)
        {
            return
        }
    }
    $Global:Anzahl++
    $_
}
```

```

}
$Global:Anzahl = 0

3..100 | test-prime;"$Anzahl Prims gefunden."

```

Beispiel:

Auf die Pipeline direkt zugreifen

```

# Direkt auf die Pipeline zugreifen
function PipeTest
{
    "Anzahl Elemente in der Pipeline: $($($Input).Count)"
    $Input.Reset()
    while ($Input.MoveNext())
    { $Input.Current.Name}
}

1..3 | PipeTest

```

Beispiel:

Fehlerbehandlung bei der Eingabe

```

#Fehlerbehandlung bei der Eingabe
$Zahl = -1
trap {"Fehler - Eingabe muss Zahl sein."; continue}
$Eingabe = Read-Host "Bitte Zahl zwischen 1 und 10 eingeben"
# In Integer-Wert konvertieren
$Zahl = [int]$Eingabe
Write-host "Eingegeben wurde: $Zahl"

```

Beispiel:

Eingabe bei fehlerhafter Eingabe wiederholen

```

# Eingabe solange wiederholen, bis Eingabe korrekt ist

function test
{
    $Global:EingabeOK = $False
    while (!$EingabeOK)
    {
        trap [InvalidCastException] {"Fehler - Eingabe muss Zahl sein."; $Global:EingabeOK=$False;continue}
        $Global:EingabeOK = $True
        $Eingabe = Read-Host "Bitte Zahl zwischen 1 und 10 eingeben"
        $Zahl = [int]$Eingabe
    }
    Write-host "Eingegeben wurde: $Zahl"
}

test

```

Beispiel:

Fehlerinformation ausgeben

```

# Fehlerinformation ausgeben

```

```
function ErrorInfo ()
{
    param ([Management.Automation.ErrorRecord]$ER)
    Write-Host "*****"
    Write-Host "          Fehlerinfo          "
    Write-Host "Fehlertyp: " $ER.Exception.GetType().Name
    Write-Host "Fehlerquelle: " $ER.InvocationInfo.Line
    Write-Host "*****"
}
# Alle Fehler löschen
$Error.Clear()
# Fehler erzeugen
get-item "GibtEsNicht.txt" -ErrorAction SilentlyContinue
# Fehlerinfo anzeigen
ErrorInfo $Error[0]
```

Beispiel:

Verschiedene Fehlertypen abfragen

```
# Verschiedene Ausnahmetypen abfragen
trap [DivideByZeroException] {"Division durch Null ist nicht
erlaubt!";continue}
$Anzahl = 1235
$Anzahl /= $Null
"Der Wert von `$Anzahl ist $Anzahl"
# Die folgende Ausnahme wird nicht abgefangen
$LetzteZiffer = 0
$LetzteZiffer =
$Anzahl.ToString().Substring($Anzahl.ToString().Length, 1)
"Die letzte Ziffer: $LetzteZiffer"
```

Beispiel:

Fehler mit Write-Error schreiben

```
# Beispiel für Write-Error
$Error.Clear()
Get-Childitem *.* | Foreach-Object {
    if ($_.Extension -ne ".ps1")
    {
        Write-Error "Falscher Dateityp" -errorID "PS1" -targetobject
$_
        "Anzahl Fehler: $($Error.Count)"
    }
    else { $_ }
}
```

Beispiel:

Alle Exe-Dateien im Suchpfad auflisten

```
# -----
# Alle Exe-Dateien im Suchpfad auflisten
# -----
$Pfad = $env:Path
$Pfade = $Pfad.Split(";")
$Gesamt = 0
foreach ($P In $Pfade)
{
```

```
Write-Host -foregroundcolor Yellow "Exe-Dateien in $P"
Write-Host "===== "
if ($P -ne "")
{
    Get-Childitem *.exe -path $P | format-wide name
    $Anzahl = (Get-Childitem *.exe -path $P).Count
    $Gesamt += $Anzahl
    Write-Host -foregroundcolor green "Anzahl Dateien: $Anzahl"
}
Write-Host "Gesamte Anzahl an Dateien: $Gesamt"
```

Kapitel 9: Die Windows PowerShell für (etwas) Fortgeschrittene

Beispiel:

Warten auf Esc-Taste (1)

```
# Warten auf Esc-Taste
while ($True)
{
    if ($Host.UI.RawUI.KeyAvailable)
    {
        $Taste = $Host.UI.RawUI.ReadKey("NoEcho, IncludeKeyUp")
        $Taste
        if ($Taste.VirtualKeyCode -eq 27)
        { break }
    }
}
```

Beispiel:

Warten auf Esc-Taste (2)

```
# -----
#Warten auf Esc-Taste
# -----
while ($True)
{
    if ($Host.UI.RawUI.KeyAvailable -and
$Host.UI.RawUI.ReadKey("NoEcho, IncludeKeyUp").VirtualKeyCode -eq
27)
    { break }
}
```

Beispiel:

Ja/Nein-Abfragebox

```
# -----
# Ja/Nein-Abfragebox
# -----
[void][Reflection.Assembly]::LoadWithPartialName("Windows.Forms")
function YesNoAbfrage
{
    $Buttons = [Windows.Forms.MessageBoxButtons]::YesNo
    $YesButton = [Windows.Forms.DialogResult]::Yes
    $Icon = [Windows.Forms.MessageBoxIcon]::Question
    if ([Windows.Forms.MessageBox]::Show("Noch ein Durchlauf",
"Abfrage", $Buttons,
    $Icon) -eq $YesButton)
    { return $True }
    else { return $False }
}

if (YesNoAbfrage -eq $True)
{ "gleich geht es weiter..." }
else { "dann eben nicht..." }
```

Beispiel:

RunAs-Beispiel

```
# -----
# Beispiel fuer Starten mit RunAs-Credentials
# -----
$Cred = Get-Credential
$StartInfo = New-Object Diagnostics.ProcessStartInfo
$StartInfo.UserName = $Cred.UserName
$StartInfo.Password = $Cred.Password
$StartInfo.Arguments = ""
$StartInfo.WorkingDirectory = ""
$StartInfo.FileName = "$env:windir\system32\regedt32.exe"
$StartInfo.UseShellExecute = $False
[Diagnostics.Process]::Start($StartInfo)
```

Beispiel:

Fortschrittsanzeige

```
# -----
# Beispiel für Fortschrittsanzeigen
# -----
$Nr = 0
cd "C:\Programme\PowerShellIDE"
$Anzahl = $Dateien.Length
if (!(Test-Path "C:\WPSkripts"))
{
    mkdir C:\WPSkripts
}
$Dateien = Get-ChildItem -Filter *.ps1
foreach ($D In $Dateien)
{
    $Nr++
    Write-Progress -Activity "Dateien werden kopiert" -Status
    "Bereits kopiert: $Nr" `
    -PercentComplete ($Nr/$Dateien.Length*100)
    Copy-Item -Path $D -Destination "C:\WPSkripts\"
}
```

Beispiel:

FSO-Objekte benutzen

```
# -----
# Laufwerke auflisten
# -----
$Fso = New-Object -ComObject Scripting.FileSystemObject
foreach ($Dr In $Fso.Drives)
{
    if ($Dr.IsReady)
    {
        "Gesamtkapazität: {0:N0} MB" -f $Dr.TotalSize
        "Freier Platz: {0:N0} MB" -f $Dr.AvailableSpace
    }
}
```

Beispiel:

Shortcut anlegen mit Fehlerbehandlung

```
# -----
# Shortcut anlegen
```

```
# -----
$Error.Clear()
trap {"Es gab ein Problem beim Anlegen des COM-Objekts.";continue}
$Wsh = New-Object -ComObject WScript.Shell
if ($Error.Count -eq 0)
{
    trap {"Es gab ein Problem beim Anlegen des Shortcuts.";continue}
    $Sh = $Wsh.CreateShortcut("$Env:Userprofile\Desktop\Calc.lnk")
    $Sh.TargetPath = "C:\Windows\System32\Calc.exe"
    $Sh.Save()
    Write-Host "Shortcut wurde angelegt!"
}
else { Write-Host "Shortcut wurde nicht angelegt!" }
```

Beispiel:**Shortcut-Anlegen (2)**

```
# -----
# Shortcut für WPS-Skript anlegen
# -----
$Error.Clear()
trap {"Es gab ein Problem beim Anlegen des COM-Objekts.";continue}
$Wsh = New-Object -ComObject WScript.Shell
if ($Error.Count -eq 0)
{
    trap {"Es gab ein Problem beim Anlegen des Shortcuts.";continue}
    $Sh =
$Wsh.CreateShortcut("$Env:Userprofile\Desktop\WPS_Skript.lnk")
    $Sh.TargetPath = $Pshome+"\PowerShell.exe"
    $Sh.Arguments = "-noexit -c &'C:\Dokumente und
Einstellungen\Pemo\HalloTest.ps1'"
    $Sh.Save()
    Write-Host "Shortcut wurde angelegt!"
}
else { Write-Host "Shortcut wurde nicht angelegt!" }
```

Beispiel:**Betreff der Nachrichten im Outlook-Posteingang ausgeben**

```
# -----
# Betreff der Nachrichten im Outlook-Posteingang ausgeben
# -----
$OutApp = New-Object -comobject Outlook.Application
$Posteingang = $OutApp.Session.GetDefaultFolder(6)
Write-Host $Posteingang.Items.Count "Eintraege im Posteingang"
foreach ($Eintrag In $Posteingang.Items)
{ $Eintrag.Subject }
```

Beispiel:**Eintrag zum Outlook-Terminkalender anlegen**

```
# -----
# Anlegen eines Outlook-Termins
# -----
$olFolderCalendar = 9
$OutApp = New-Object -com Outlook.Application
$Kalender = $OutApp.Session.GetDefaultFolder($olFolderCalendar)
```



```
$Termin = $Kalendar.Items.Add(1)
$Termin.Start = [datetime]::now.AddDays(3)
$Termin.Subject = "PowerShell richtig lernen"
$Termin.Location = "Irgendwo im Sueden"
$Termin.Save()
"Termin wurde angelegt."
```

Beispiel:

Excel steuern

```
# -----
# Excel steuern
# -----
$ExApp = New-Object -comobject Excel.Application
$ExApp.Visible = $True
$Wb = $ExApp.Workbooks.Add()
$ExApp.ActiveCell.Value = "123"
$Wb.SaveAs("$env:userprofile\Eigene Dateien\123Buch.xls")
$ExApp.Quit()
```

Beispiel:

Daten in Excel-Chart ausgeben

Hinweis:

Das Beispiel (wie alle übrigen »Office-Beispiele« auch) funktioniert leider nicht, wenn Excel 2003 und 2007 zusammen installiert sind. Es erscheint eine »seltsame« Fehlermeldung, die aber in der Ms-Supportdatenbank beschrieben ist.

```
# -----
# Daten in einem Excel-Chart ausgeben
# -----
$xl3DPie = -4102
$xl3DBarStacked = 61
$xlLocationAsObject = 2
$ExApp = New-Object -com Excel.Application
$ExApp.Visible = $True
[void]$ExApp.Workbooks.Add()
$Sh = $ExApp.ActiveSheet
for($i=1;$i -lt 11;$i++)
{
    $z = New-Object Random
    $Sh.Cells.Item(1, $i).Value = $z.Next(1,100)
}
$Bereich = $Sh.Range("A1:J1")
$Bereich.Cells.Count
$Ch = $ExApp.Charts.Add()
$Ch.ChartType = $xl3DPie
$Ch.SetSourceData($Bereich)
# Chart soll in Tabelle1 eingebettet werden
$Ch.Location($xlLocationAsObject, "Tabelle1")
# Titel festlegen - ein wenig umständlich, da es über das
ChartObject-Objekt geht
$Sh.ChartObjects(1).Chart.HasTitle = $True
$Sh.ChartObjects(1).Chart.ChartTitle.Text = "Login-Statistik 2007"
# Zum Schluss Arbeitsmappe speichern
$ExApp.ActiveWorkbook.SaveAs("LoginChart2007")
```

Beispiel:

Alias komfortabler setzen

Hinweis:

Das Skript muss »dot sourced« aufgerufen werden, z.B. `.\09_AliasSetzen.ps1`. Im Skript muss bei **set-alias** ein **-Scope Script** übergeben werden, da sich der Alias ansonsten auf das Skript beschränkt (da ist im Buch nicht der Fall).

```
# -----
# Alias komfortabler setzen
# -----

function Set-Alias-strict($Alias, $Value)
{
    $Cmd = Get-Command $Value
    if ($Cmd -ne $Null)
    {
        Set-Alias $Alias $Value -Scope Script
    }
}
```

Beispiel:

Backup von Ps1-Dateien

```
# -----
# Backup aller Ps1-Dateien des aktuellen Verzeichnisses
# -----

function BackupSkripts
{
    $SkriptPfad = "$env:UserProfile\Eigene Dateien\Eigene Skripts"
    if ((Test-Path $SkriptPfad) -eq $False)
    { New-Item $SkriptPfad -ItemType Directory }
    Get-ChildItem *.ps1 | Copy-Item -Destination $SkriptPfad
}
```

Beispiel:

Dateierweiterungen nach Häufigkeit sortiert ausgeben

```
# -----
# Dateierweiterungen nach Häufigkeit sortiert ausgeben
# -----

Get-Childitem | Foreach-Object {$Ext =
[Io.Path]::GetExtension("$Home\$_");if ($Ext -ne ""){$Ext}} |
group | sort count -descending
```

Beispiel:

Auf numerische Eingabe prüfen

```
# -----
#Abfrage auf Numerisch
# -----

$Betrag = Read-Host "Bitte mal nen Betrag eingeben."
```

```
[void][
Reflection.Assembly]::LoadWithPartialName("Microsoft.VisualBasic")
if ([Microsoft.VisualBasic.Information]::IsNumeric($Betrag) -eq
$False)
{
    Write-Host "Fehler bei der Eingabe! - der Typ ist
$( $Betrag.GetType().Name )"
}
else
{
    Write-Host "Danke, das war numerisch."
}
```

Beispiel:

E-Mail mit Anhang versenden

Hinweis

Damit es funktioniert, müssen die Server-Adressen eingesetzt werden.

```
# -----
# E-Mail mit Anhang versenden
# -----
$MailServer = "<SMTPServerName>"
$AtDatei = New-Object Net.Mail.Attachment("C:\Boot.ini")
$Nachricht = New-Object Net.Mail.MailMessage("admin@wps-zone.de",
"peterm@activetraining.de")
$Nachricht.Subject = "Server-Alarm"
$Nachricht.Body = "Server-Alarm auf PM_R2 - Zeitpunkt: " + (Get-
Date)
$Nachricht.IsBodyHTML = $False
$Nachricht.Attachments.Add($AtDatei)
$MailClient = New-Object Net.Mail.SmtpClient($MailServer)
$MailClient.UseDefaultCredentials = $False
$MailClient.Credentials = New-Object
Net.NetworkCredential("<SMTPBenutzername>", "<SMTPPasswort>")
$MailClient.Send($Nachricht)
Write-Host "Mail wurde versendet..."
```

Beispiel:

Datenbank anlegen

Hinweis

Damit es funktioniert, muss lokal ein Microsoft SQL Server (z.B. SQL Server 2005 Express Edition) installiert sein.

```
# -----
# LoginDB-Datenbank anlegen
# -----
$CnSt = "Server=.\SQLEXPRESS;Trusted_Connection=Yes"
$Cn = New-Object Data.SqlClient.SqlConnection($CnSt)
$Cn.Open()
$Sql = "CREATE DATABASE LoginDB"
$Cmd = $Cn.CreateCommand()
$Cmd.CommandText = $Sql
$Result = $Cmd.ExecuteNonQuery()
Write-Host "Datenbank wurde angelegt"
```

```

$Cn.Close()
$CnSt =
"Server=.\SQLEXPRESS;Trusted_Connection=Yes;Database=LoginDB"
$Cn = New-Object Data.SqlClient.SqlConnection($CnSt)
$Cn.Open()
$Sql = "CREATE TABLE Logins (LoginID int NOT NULL IDENTITY(1,1)
CONSTRAINT pkLoginID PRIMARY KEY, UserID int, Zeitpunkt datetime
NULL)"
$Cmd = $Cn.CreateCommand()
$Cmd.CommandText = $Sql
$Result = $Cmd.ExecuteNonQuery()
Write-Host "Logins-Tabelle wurde angelegt"
$Sql = "CREATE TABLE Users (UserID int NOT NULL IDENTITY(1,1)
CONSTRAINT pkUserID PRIMARY KEY, UserName varchar(64), EMail
varchar(100))"
$Cmd = $Cn.CreateCommand()
$Cmd.CommandText = $Sql
$Result = $Cmd.ExecuteNonQuery()
$Cn.Close()
Write-Host "User-Tabelle wurde angelegt"

```

Beispiel:

Eintrag in Datenbank durchführen

Hinweis

Setzt das Vorhandensein der im letzten Beispiel angelegten LoginDB-Datenbank voraus.

```

# -----
# Eintrag in LoginDB-Datenbank
# -----
$UserID=1
if ($Args.Length -gt 0) { $UserID = $Args[0] }
$CnSt =
"Server=.\SQLEXPRESS;Trusted_Connection=Yes;Database=LoginDB"
$Cn = New-Object Data.SqlClient.SqlConnection($CnSt)
$Cn.Open()
$Sql = "INSERT INTO Logins VALUES(@UserID, @Zeitpunkt)"
$Cmd = $Cn.CreateCommand()
$Cmd.CommandText = $Sql
$P = $Cmd.Parameters.AddWithValue("@UserID", $UserID)
$P = $Cmd.Parameters.AddWithValue("@Zeitpunkt", (Get-Date))
$Result = $Cmd.ExecuteNonQuery()
$Cn.Close()
Write-Host "Eintrag wurde hinzugefuegt"

```

Beispiel:

Einträge in Datenbanktabelle auflisten

Hinweis

Setzt das Vorhandensein der im letzten Beispiel angelegten LoginDB-Datenbank voraus.

```

# -----
# Auflisten aller Eintraege in der LoginDB-Datenbank
# -----

```

```

$CnSt =
"Server=.\SQLEXPRESS;Trusted_Connection=Yes;Database=LoginDB"
$Cn = New-Object Data.SqlClient.SqlConnection($CnSt)
$Cn.Open()
$Sql = "SELECT UserName, EMail, Zeitpunkt FROM Logins, Users WHERE
Logins.UserID=Users.UserID"
$Cmd = $Cn.CreateCommand()
$Cmd.CommandText = $Sql
$Dr = $Cmd.ExecuteReader()
while ($Dr.Read())
{
    Write-Host "User:" $Dr.Item("UserName") "E-Mail:"
$Dr.Item("EMail") "Zeitpunkt:" $Dr.Item("Zeitpunkt") `n
}
$Cn.Close()

```

Beispiel:

Anzahl an Zeilen aller Ps1-Dateien ausgeben

```

# -----
# Kleine Statistik der Ps1-Files im aktuellen Verzeichnis
# -----
Clear-Host
Set-Location $env:userprofile
Write-Host "Alle Ps1-Skripts im Verzeichnis: $env:userprofile"
$Liste = New-Object Collections.SortedList
Set-PSDebug -off
Get-ChildItem -Filter *.ps1 | Foreach-Object {
    $Zeilen = Get-Content $_
    $AnzahlZeilen = $Zeilen.Count
    if ($AnzahlZeilen -eq $Null) { $AnzahlZeilen = 0 }
    "Skript: $_ Anzahl Zeilen: $($Zeilen.Count)";
    $Liste.Add($_.Name, $AnzahlZeilen)
}
# Ab hier wird es recht speziell dank der Klassen DataTable und
# DataView
$Ta = New-Object Data.DataTable
[void]$Ta.Columns.Add("Dateiname", [string])
[void]$Ta.Columns.Add("Zeilen", [int])
foreach ($Key in $Liste.Keys)
{
    $Neu =
    $Ta.NewRow(); $Neu.Item("Dateiname")=$Key; $Neu.Item("Zeilen")=$Liste[$Key];
    $Ta.Rows.Add($Neu)
}
$DvSortiert = New-Object Data.DataView($Ta)
$DvSortiert.Sort = "Zeilen Desc"
$DvSortiert

```

Beispiel:

Namen aller SQL Server-Instanzen auflisten (1)

```

# -----
# Namen aller SQL Server-Instanzen auflisten (1)
# -----
[void][Reflection.Assembly]::LoadWithPartialName("Microsoft.SqlServer.Smo")
$Server = New-Object
Microsoft.SqlServer.Management.Smo.Server(".\SQLEXPRESS")
foreach ($Db In $Server.Databases)
{

```

```
if (!$Db.IsSystemObject)
{ "Datenbankname: " + ($Db.Name) }
}
```

Beispiel:

Die Namen aller SQL Server-Instanzen auflisten (2)

```
# -----
# Namen aller SQL Server-Instanzen auflisten
# -----
$SqlInstanzen = [Data.Sql.SqlDataSourceEnumerator]::Instance
$Ta = $SqlInstanzen.GetDataSources()
foreach ($Zeile In $Ta.Rows)
{
    "Server-Name: " + ($Zeile.Item("ServerName"))
    "Instanz-Name: " + ($Zeile.Item("InstanceName"))
    "Version: " + ($Zeile.Item("Version"))
}
```

Beispiel:

RunAs-Aufruf mit Secure String

```
# -----
# Ein Beispiel für einen RunAs-Aufruf
# -----
$ProgArgs = "Sysdm.cpl"
$ProgArgs = "TimeDate.cpl"
$Computer = $env:Computername
$Cred = get-credential
Trap {"Anmeldung nicht möglich.";continue}
[void][Diagnostics.Process]::Start("control.exe", $ProgArgs,
$Cred.Username, $Cred.Password, $Computer)
```

Beispiel:

Umgang mit einem Secure String

```
# -----
# Umgang mit einem SecureString
# -----
$Pw = Read-Host "Das Kennwort?" -AsSecureString
ConvertFrom-SecureString $Pw | out-file c:\temp\Geheim.txt
$PwNeu = get-content C:\temp\Geheim.txt | ConvertTo-SecureString
# Nur zu Anschauungszwecken
Get-Content C:\temp\Geheim.txt
```

Beispiel:

Verschlüsselten String wieder lesbar machen

```
# -----
# Verschlüsselten String wieder lesbar machen
# -----
$Pw = Read-Host "Pw?" -asSecureString
$PtrStr = [IntPtr]
[Runtime.InteropServices.Marshal]::SecureStringToBSTR($Pw)
```

```
$PwNeu =
[Runtime.InteropServices.Marshal]::PtrToStringBSTR($PtrStr)
$PwNeu
```

Beispiel:

Berechtigungen zu einem Verzeichnis hinzufügen

Hinweis

Damit es funktioniert, muss der Name des Administratorkontos stimmen.

```
# -----
# Beispiel für Set-ACL
# -----
function Add-Security
{
    param ($Objekt,
        [Security.Principal.NTAccount]$Identity,
        [Security.AccessControl.FileSystemRights]$AccessMask,
        [Security.AccessControl.AccessControlType]$Typ)
    if ($Objekt.IsPSContainer)
    {
        $InheritanceFlags =
[Security.AccessControl.InheritanceFlags]::ContainerInherit `
        -bor
[Security.AccessControl.InheritanceFlags]::ObjectInherit
    }
    else
    {
        $InheritanceFlags =
[Security.AccessControl.InheritanceFlags]::None
    }
    $PropagationFlags =
[Security.AccessControl.PropagationFlags]::None
    # Alten ACL holen
    $ObjektACL = Get-Acl $Objekt
    # AccessRule hinzufuegen
    $SecRule = new-object
Security.AccessControl.FileSystemAccessRule($Identity, `
        $AccessMask, $InheritanceFlags, $PropagationFlags, $Typ)
    $ObjektACL.AddAccessRule($SecRule)
    Set-Acl $Objekt $ObjektACL
}

# Neues Verzeichnis anlegen
$Dir = New-Item -itemtype Directory TestACL -force
$ID = new-object Security.Principal.NTAccount("Administrator")
$Rechte = [Security.AccessControl.FileSystemRights]"Read,Write"
$Typ = [Security.AccessControl.AccessControlType]::Allow
Add-Security $Dir $ID $Rechte $Typ
"Ein ACE wurde zu $Dir hinzugefuegt..."
Get-Acl $Dir

# Neue Datei anlegen
$Datei = New-Item -itemtype file TestACL.txt -force
Add-Security $Datei $ID $Rechte $Typ
"Ein ACE wurde zu $Datei hinzugefuegt..."
Get-Acl $Datei
```

Kapitel 10: ADSI und die Windows PowerShell

Dieses Kapitel hätte ich am liebsten noch einmal komplett überarbeitet. Das merkt man auch einigen der Skripts an. Insbesondere die AD-Suche ließe sich etwas eleganter formulieren. Das Anlegen von Benutzern und Gruppen ist nicht einheitlich, vor allem das Hinzufügen eines Users zu einer Gruppe. Es liegt aber auch teilweise an der etwas inkonsistenten und teilweise sogar fehlerhaften Implementierung bei [ADSI]. Hoffen wir, dass es mit der PowerShell 2 besser wird.

Hinweis:

Leider enthalten die Beispiele in Kapitel 10 einen sehr ärgerlichen Flüchtigkeitsfehler. Dieser besteht darin, dass meistens auf das [ADSI] kein "" folgt und damit kein Domänenkontroller angesprochen wird und der Befehl mit einer Fehlermeldung abbricht. Die »Entschuldigung« besteht darin, dass ich nachträglich aus dem Manuskript den LDAP-Pfad für den Server entfernt, die Anführungsstriche aber nicht wieder eingefügt habe).

Alle Skripts in der Beispielsammlung wurden beim Zusammenstellen der Beispiele von mir aber noch einmal getestet.

Beispiel:

Benutzer anlegen

```
# -----
# Anlegen eines neuen Benutzers
# -----
$AD = [ADSI]" "
$UserName = "NeuUser08"
$ADUser = $AD.Create("user", "Cn=$UserName")
$ADUser.Put("sAMAccountName", $UserName)
$ADUser.SetInfo()
Write-Host "Benutzer wurde ordnungsgemaess angelegt."
```

Beispiel:

Gruppe anlegen

```
# -----
# Anlegen einer neuen Gruppe
# -----
$AD = [ADSI]" "
$ADGroup = $AD.Create("group", "cn=NeuGruppe")
$ADGroup.Put("sAMAccountName", "NeuGruppe")
$ADGroup.SetInfo()
Write-Host "Gruppe wurde ordnungsgemaess angelegt"
```

Beispiel:

Namen der Computer im AD auflisten

```
# -----
# Suche im AD nach allen Computern
# -----
$SuchKategorie = "computer"
```



```
$ADDomain = [ADSI]" "

$ADSearcher = New-Object DirectoryServices.DirectorySearcher
$ADSearcher.SearchRoot = $ADDomain
$ADSearcher.Filter = "(objectCategory=$SuchKategorie)"

$PropListe = @("name")
[void] $ADSearcher.PropertiesToLoad.AddRange($PropListe)

$Ergebnisse = $ADSearcher.FindAll()

foreach ($Erg in $Ergebnisse)
{
    $ADComp = $Erg.Properties
    $ADComp.name
}
```

Beispiel:

User-Namen auflisten

```
# -----
# Eigenschaften von Usern im AD auflisten
# -----
$SuchKategorie = "user"

$ADDomain = [ADSI]" "

$ADSearcher = New-Object DirectoryServices.DirectorySearcher
$ADSearcher.SearchRoot = $ADDomain
$ADSearcher.Filter = ("(objectCategory=$SuchKategorie)")

# LDAP-Attributnamen hinzufügen
# Mail muss mit großem M geschrieben werden
$PropListe = "name", "description", "Mail"
[void] $ADSearcher.PropertiesToLoad.AddRange($PropListe)

$Ergebnisse = $ADSearcher.FindAll()

foreach ($Erg in $Ergebnisse)
{
    $ADUserProps = $Erg.Properties
    # Hier muss mail mit kleinem m geschrieben werden
    write-host -fore gray "Name: $($ADUserProps.name), E-Mail:
    $($ADUserProps.mail)"
}
```

Beispiel:

WMI und ADSI kombinieren

```
# -----
# WMI-Abfragen per ADSI
# -----

$ADDomain = [ADSI]" "

$SuchKategorie = "computer"

$ADSearcher = New-Object DirectoryServices.DirectorySearcher
```

```
$ADSearcher.SearchRoot = $ADDomain
$ADSearcher.Filter = ("(objectCategory=$SuchKategorie)")

$PropListe = @("name")
[void]$ADSearcher.PropertiesToLoad.AddRange($PropListe)

$Ergebnisse = $ADSearcher.FindAll()

foreach ($Erg in $Ergebnisse)
{
    $ADComp = $Erg.Properties
    $Comp = $ADComp.name
    write-host "Prüfe $Comp..."
    $Anzahl++
    $OS = get-wmiobject -class Win32_OperatingSystem -Computer
$Comp -errorAction SilentlyContinue
    if ($OS.Version -eq "5.2.3790" -and $OS.CSDVersion -eq
"Service Pack 1")
    {
        write-host -foregroundcolor blue "SP1 ist auf $Comp
installiert"
    }
}
write-host -foregroundcolor red "Fertig - $Anzahl Computer
geprüft."
```

Beispiel:

Lokalen User (ohne AD) anlegen

```
# -----
# Lokalen User anlegen
# -----
$UserName = "Pemo2007"
# Es ist besser, hier den Computernamen anstelle des Punktes
anzugeben
$Comp = "<Servername>"
$AD = [ADSI]"WinNT://$Comp"
$NewUser = $AD.Create("user", $UserName)
$NewUser.Put("Description", "Testaffe")
# Kennwort kann je nach Ricthlinie eine Rolle spielen
$NewUser.SetPassWord("R2D2!123")
$NewUser.SetInfo()
Write-Host "User $UserName wurde angelegt."
```

Beispiel:

Lokale Gruppe (ohne AD) anlegen

```
# -----
# Lokale SAM-Gruppe anlegen
# -----
$GroupName = "Bezugsgruppe07"
# Hier ist der Punkt für den Computernamen ok
$AD = [ADSI]"WinNT://."
$NewGroup = $AD.Create("group", $GroupName)
$NewGroup.Put("Description", "Testgruppe")
$NewGroup.SetInfo()
Write-Host "Gruppe $GroupName wurde angelegt."
```

Beispiel:

Auflisten der lokalen Benutzerkonten per WMI

```
# -----  
# Auflisten der lokalen Benutzerkonten per WMI  
# Der Namensfilter wird in diesem Beispiel nicht genutzt  
# -----  
$UserAccounts = Get-WmiObject Win32_UserAccount -computersname "."  
-filter "LocalAccount=True AND Name Like '%'" | Select-Object  
Domain,Name,Disabled  
$UserAccounts | format-table -a
```