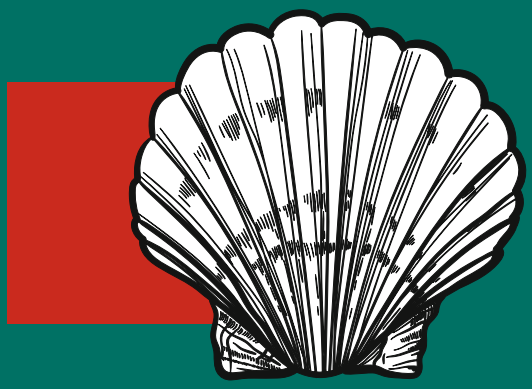
Holger SCHWICHTENBERG

Für Windows, Linux und macOS

PowerShell 7 und Windows PowerShell 5



DAS PRAXISBUCH

5. Auflage

Im Internet: Codebeispiele und PowerShell-Kurzreferenz

HANSER

www.IT-Visions.de
Dr. Holger Schwichtenberg

Schwichtenberg

PowerShell 7 und Windows PowerShell 5 – das Praxisbuch

Holger Schwichtenberg

PowerShell 7 und Windows PowerShell 5

Das Praxisbuch

5., aktualisierte Auflage

HANSER

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Lektorat: Sylvia Hasselbach
Copy editing: Matthias Bloch, Bochum, und Sandra Gottmann, Wasserburg
Umschlagdesign: Marc Müller-Bremer, www.rebranding.de, München
Umschlagrealisation: Max Kostopoulos
Titelmotiv: © shutterstock.com/Irina Kolesnichenko
Satz: Eberl & Koesel Studio, Kempten (revised version)
Druck und Bindung: Hubert & Co. GmbH & Co. KG BuchPartner, Göttingen
Printed in Germany

Print-ISBN: 978-3-446-47296-9
E-Book-ISBN: 978-3-446-47446-8
E-Pub-ISBN: 978-3-446-47574-8

Inhalt

Vorwort

Liebe Leserin, lieber Leser,

willkommen zur aktuellen Auflage meines PowerShell-Buchs! Es handelt sich hierbei um die fünfte Auflage des Windows PowerShell 5-Buches und die neunte Auflage des PowerShell-Buches insgesamt, das erstmalig 2007 bei Addison-Wesley erschienen ist.

Was ist das Thema dieses Buchs?

Das vor Ihnen liegende Fachbuch behandelt die Windows PowerShell in der Version 5.1 sowie die plattformneutrale PowerShell 7.2 von Microsoft wie auch ergänzende Werkzeuge von Microsoft und Drittanbietern (z. B. PowerShell Community Extensions). Es gibt in dem Buch auch Ausblicke auf die PowerShell 7.3, die derzeit in der Entwicklung ist.

Das Buch ist aber auch für Sie geeignet, wenn Sie noch Windows PowerShell 2.0/3.0/4.0/5.0 oder PowerShell Core 6.x bzw. PowerShell 7.0/7.1 einsetzen. Welche Funktionen neu hinzugekommen sind, wird jeweils in diesem Buch erwähnt.

Wer bin ich?

Mein Name ist Holger Schwichtenberg, ich bin derzeit 49 Jahre alt und habe im Fachgebiet Wirtschaftsinformatik promoviert. Ich lebe (in Essen, im Herzen des Ruhrgebiets) davon, dass mein Team und ich im Rahmen unserer Firma *www.IT-Visions.de* anderen Unternehmen bei der Entwicklung von .NET-, Web- und PowerShell-Anwendungen beratend und schulend zur Seite stehen. Zudem entwickeln wir im Rahmen der MAXIMAGO GmbH (*www.MAXIMAGO.de*) Software im Auftrag von Kunden in zahlreichen Branchen.

Es ist nur ein Hobby, IT-Fachbücher zu schreiben, denn damit kann man als Autor kaum Geld verdienen. Dieses Buch ist, unter Mitzählung aller nennenswerten Neuauflagen, das 92. Buch, das ich allein oder mit Co-Autoren geschrieben habe. Meine weiteren Hobbys sind Mountain Biking, Fotografie und Reisen.

Natürlich verstehe ich das Bücherschreiben auch als Werbung für die Arbeit unserer Unternehmen, und wir hoffen, dass der ein oder andere von Ihnen uns beauftragen wird, Ihre Organisation durch Beratung, Schulung und Auftragsentwicklung zu unterstützen.

Wer sind Sie?

Damit Sie den optimalen Nutzen aus diesem Buch ziehen können, möchte ich – so genau es mir möglich ist – beschreiben, an wen sich dieses Buch richtet. Hierzu habe ich einen Fragebogen ausgearbeitet, mit dem Sie schnell erkennen können, ob das Buch für Sie geeignet ist.

Sind Sie Systemadministrator in einem Windows-Netzwerk?	○ Ja	○ Nein
Laufen die für Sie relevanten Computer mit den von PowerShell unterstützten Betriebssystemen? (Windows 7/8/8.1/10/11, Windows Server 2008/2008 R2/2012/2012 R2/2016/2019/2022, macOS, Linux)	○ Ja	○ Nein
Sie besitzen zumindest rudimentäre Grundkenntnisse im Bereich des (objektorientierten) Programmierens?	○ Ja	○ Nein
Wünschen Sie einen kompakten Überblick über die Architektur, Konzepte und Anwendungsfälle der PowerShell?	○ Ja	○ Nein
Sie können auf Schritt-für-Schritt-Anleitungen verzichten?	○ Ja	○ Nein
Sie können auf formale Syntaxbeschreibungen verzichten und lernen lieber an aussagekräftigen Beispielen?	○ Ja	○ Nein
Sie erwarten nicht, dass in diesem Buch **alle** Möglichkeiten der PowerShell detailliert beschrieben werden?	○ Ja	○ Nein
Sind Sie, nachdem Sie ein Grundverständnis durch dieses Buch gewonnen haben, bereit, Detailfragen in der Dokumentation der PowerShell, von .NET und WMI nachzuschlagen, da das Buch auf rund 1400 Seiten nicht alle Details erläutern, sondern – in dem Sinn „Hilfe zur Selbsthilfe" – nur **ausgewählte Aspekte** darstellen kann, anhand deren Sie dann Ihre eigenen Lösungen für Ihre spezifischen Szenarien entwickeln?	○ Ja	○ Nein

Wenn Sie alle obigen Fragen mit „Ja" beantwortet haben, ist dieses Fachbuch richtig für Sie. In anderen Fällen sollten Sie sich erst mit einführender Literatur beschäftigen.

Was ist neu in diesem Buch?

Gegenüber der vorherigen Auflage zur PowerShell 5.1/PowerShell 7.0 wurde das Buch um die neuen Commandlets, Funktionen und Operationen in PowerShell 7.1 und 7.2 erweitert.

Zudem wurden die bestehenden Inhalte des Buchs an vielen Stellen optimiert. Das Kapitel zu „Docker-Container" wurde in weiten Teilen überarbeitet. Zum Dateisystem, zur Dokumentenverarbeitung, zum Netzwerk, zu Hyper-V und zu Azure DevOps-Pipelines habe ich Praxislösungen ergänzt.

Zudem wurde das Feedback einiger Leser eingearbeitet, um Beispiele und Texte zu optimieren.

Sind in diesem Buch alle Features der PowerShell beschrieben?

Die PowerShell umfasst mittlerweile mehrere Tausend Commandlets mit jeweils zahlreichen Optionen. Zudem gibt es unzählige Erweiterungen mit vielen Hundert weiteren Commandlets. Außerdem existieren zahlreiche Zusatzwerkzeuge. Es ist allein schon aufgrund der Vorgaben des Verlags für den Umfang des Buchs nicht möglich, alle Commandlets und Parameter hier auch nur zu erwähnen. Zudem habe ich – obwohl ich selbst fast jede Woche mit der PowerShell in der Praxis arbeite – immer noch nicht alle Commandlets und alle Parameter jemals selbst eingesetzt.

Ich beschreibe in diesem Buch, was ich selbst in der Praxis, in meinen Schulungen und bei Kundeneinsätzen verwende. Es macht auch keinen Sinn, hier jedes Detail der PowerShell zu dokumentieren. Stattdessen gebe ich Ihnen **Hilfe zur Selbsthilfe**, damit Sie die Konzepte gut

verstehen und sich dann Ihre spezifischen Lösungen anhand der Dokumentation selbst erarbeiten können.

Wie aktuell ist dieses Buch?

Die Informationstechnik hat sich immer schon schnell verändert. Seit aber auch Microsoft die Themen „Agilität“ und „Open Source“ für sich entdeckt hat, ist die Entwicklung nicht mehr nur schnell, sondern zum Teil rasant:

- Es erscheinen in kurzer Abfolge immer neue Produkte.
- Produkte erscheinen schon in frühen Produktstadien als „Preview“ mit Versionsnummern wie 0.1.
- Produkte ändern sich sehr häufig, teilweise im Abstand von drei Wochen (z. B. Visual Studio und Azure DevOps).
- Aufwärts- und Abwärtskompatibilität ist kein Ziel bei Microsoft mehr. Es wird erwartet, dass Sie Ihre Lösungen ständig den neuen Gegebenheiten anpassen.
- Produkte werden nicht mehr so ausführlich dokumentiert wie früher. Teilweise erscheint die Dokumentation erst deutlich nach dem Erscheinen der Software. Oft bleibt die Dokumentation auch dauerhaft lückenhaft.
- Produkte werden schnell auch wieder abgekündigt, wenn sie sich aus der Sicht der Hersteller bzw. aufgrund des Nutzerfeedbacks nicht bewährt haben.

HINWEIS: Nicht nur Microsoft geht so vor, sondern viele andere Softwarehersteller (z. B. Google) agieren genauso.

Unter diesen neuen Einflussströmen steht natürlich auch dieses etablierte Fachbuch. Leider kann man ein gedrucktes Buch nicht so schnell ändern wie Software. Verlage definieren nicht unerhebliche Mindestauflagen, die abverkauft werden müssen, bevor neu gedruckt werden darf. Das E-Book ist keine Alternative. Die Verkaufszahlen zeigen, dass nur eine kleine Menge von Lesern technischer Literatur ein E-Book statt eines gedruckten Buchs kauft. Das E-Book wird offenbar nur gerne als Ergänzung genommen. Das kann ich gut verstehen, denn ich selbst lese auch lieber gedruckte Bücher und nutze E-Books nur für eine Volltextsuche.

Daher kann es passieren, dass – auch schon kurz nach dem Erscheinen dieses Buchs – einzelne Informationen in diesem Buch nicht mehr zu neueren Versionen passen. Wenn Sie so einen Fall feststellen, schreiben Sie bitte eine Nachricht an mich (siehe unten). Ich werde dies dann in Neuauflagen des Buchs berücksichtigen.

Zudem ist zu beachten, dass zwischen Abgabe des Manuskripts beim Verlag und Auslieferung des Buchs aus der Druckerei an den Buchhandel meist vier bis fünf Monate liegen.

Welche PowerShell-Versionen werden besprochen?

Das Buch bespricht sowohl die Windows PowerShell 5.1 als auch die PowerShell 7.2. Es gibt in dem Buch auch Ausblicke auf die PowerShell 7.3, die derzeit in der Entwicklung ist.

- Bei der Windows PowerShell 5.1 wird die RTM-Version besprochen, die Microsoft in der aktuellen Version von Windows 10/11 bzw. Windows Server 2019/2022 mitliefert.

- Bei PowerShell 7.2 nutzen wir die RTM-Version vom 8. November 2021 ein.
- Bei PowerShell 7.3 gibt es zum Redaktionsschluss erst die Version Preview 2. Die PowerShell 7.3 wird voraussichtlich Ende 2022 erscheinen.

Warum behandelt das Buch auch noch Version 5.1 und nicht nur Version 7.2?

Windows PowerShell 5.1 ist heute in den Unternehmen in Deutschland der Standard, denn diese Version der PowerShell wird mit Windows 10/11 und Windows Server 2016, Windows Server 2019 sowie Windows Server 1709, Windows Server 1909 und Windows Server 2022 ausgeliefert.

Die PowerShell 7.2 wird bisher mit keinem einzigen Betriebssystem ausgeliefert, sondern muss getrennt heruntergeladen und installiert werden. Eine Zusatzinstallation ist in vielen Unternehmen mit stark abgeschotteten Systemen gar nicht möglich.

Ein zweites Argument für die Beibehaltung der Version 5.1 in diesem Fachbuch ist, dass die PowerShell 7.2 der Windows PowerShell 5.1 funktional immer noch nicht ganz ebenbürtig ist. Einige Befehle sind weiterhin nur in der Windows PowerShell verfügbar.

Daher wird die Windows PowerShell 5.1 auch weiterhin eine große Bedeutung haben und in diesem Buch auch weiterhin behandelt.

Welche Betriebssysteme werden besprochen?

Der Schwerpunkt des Buchs liegt auf der Nutzung der PowerShell unter Windows. Es gibt Hinweise und Beispiele für die Nutzung der PowerShell unter Linux (am Beispiel Ubuntu) und macOS.

Bei Windows gibt es Hinweise auf Unterschiede zwischen verschiedenen Windows-Varianten (Client/Server) und Windows-Versionen.

Auch wenn Windows 11 bereits erschienen ist, ist Windows 10 das im professionellen Einsatz vorherrschende Betriebssystem. Das Buch geht auf existierende kleinere Unterschiede zwischen Windows 10 und Windows 11 ein, die meisten Screenshots sind aber mit Windows 10 gemacht. Einige Screenshots sind mit älteren Windows-Versionen geschossen, was aber kein Problem ist, denn inhaltlich hat sich nichts geändert (nur optisch an der Titelleiste und der Schriftart).

Woher bekommt man die Beispiele aus diesem Buch?

Unter *http://www.powershell-doktor.de/leser* biete ich ein **ehrenamtlich betriebenes** Webportal für Leser meiner Bücher an. Bei der Erstregistrierung müssen Sie das Losungswort **Boba Fett** angeben. Nach erfolgter Registrierung erhalten Sie dann ein persönliches Zugangskennwort per E-Mail.

In diesem Portal können Sie

- die Codebeispiele aus diesem Buch in einem Archiv herunterladen,
- eine PowerShell-Kurzreferenz „Cheat Sheet“ (zwei DIN-A4-Seiten als Hilfe für die tägliche Arbeit) kostenlos herunterladen sowie
- Feedback zu diesem Buch geben (Bewertung abgeben und Fehler melden).

Kurzreferenz ("Cheat Sheet") Windows PowerShell

Autor: Dr. Holger Schwichtenberg (www.IT-Visions.de) V1.5.2 / 22.03.2018

www.IT-Visions.de
Dr. Holger Schwichtenberg

Hilfe

Alle installierten Module
Get-Module -ListAvailable | ft Name, Moduletype, ExportedCommands

Alle Befehle mit "Get-"
*Get-Command Get-**

Alle Befehle aus einem Modul
Get-Command | Where-Object module -like "ActiveDirectory" | FT Name, Module

Komplette Hilfe zu einem Befehl
Get-Help Stop-Process -full

Auflisten aller "About"-Dokumente
Get-Help about

Anzeigen des Hilfedokuments zu WMI
Get-Help about_WMI

Anzeigen aller Eigenschaften der Ergebnisobjekte
Get-Service | Get-Member

Wichtige Navigations-Commandlets

Mit den Navigations-Commandlets kann man nicht nur im Dateisystem, sondern auch anderen flachen und hierarchischen Mengen arbeiten, z.B. Registry (HKLM:, HKCU:), Umgebungsvariablen (env:) , Zertifikaten (cert:), Active Directory (AD:), usw. arbeiten, z.B.

Dir HKLM:\software
New-Item HKLM:\software\ITVisions
RD HKLM:\software\ITVisions

Get-PSDrive	Laufwerksliste
Get-Location (pwd)	Abrufen des aktuellen Standorts
Set-Location (cd)	Festlegung des aktuellen Standorts
Get-Item (gi)	Holt ein Element
Get-ChildItem (dir, ls, gci)	Auflisten der Unterelemente
Get-Content (type, cat, gc)	Abruf eines Elementinhalts (z.B. Dateiinhalt)
Set-Content (sc)	Elementinhalt festlegen
Add-Content (ac)	Elementinhalt ergänzen
New-Item (ni, mkdir)	Erstellen eines Elements (Ast oder Blatt)
Get-ItemProperty (gp)	Attribut abrufen
Set-ItemProperty (sp)	Attribut eines Elements festlegen ggf. anlegen wenn nicht vorhanden
Remove-Item (del, ri, rmdir, rm, erase)	Element löschen
Move-Item (move, mv)	Element verschieben
Copy-Item (copy, cp, cpi)	Element kopieren
Rename-Item (rni, ren)	Element umbenennen

Active Directory-Commandlets

Diese Commandlets erfordern das Active Directory-PowerShell-Modul auf dem Client und ADWS (Active Directory Webservices) auf dem AD-Server.

Get-ADObject	Abruf beliebiger Objekte aus dem AD
Get-ADUser, Get-ADGroup, Get-ADOrganizationalUnit, Get-ADDomain, Get-ADComputer, …	Abruf von spezifischen AD-Elementen
Set-ADObject, Set-ADUser, Set-ADGroup Set-ADComputer, …	Setzen von Eigenschaften eines Objekts
New-ADUser, New-ADGroup, New-ADOrganizationalUnit, …	Anlegen eines neuen AD-Objekts
Remove-ADObject	Löschen eines AD-Objekts
Rename-ADObject	Umbenennen eines AD-Objekts
Move-ADObject	Verschieben eines AD-Objekts
Set-ADAccountPassword	Festlegen eines Kennwortes
Get-ADGroupMember	Liste der Gruppenmitglieder
Add-ADGroupMember	Mitglied einer Gruppe hinzufügen
Remove-ADGroupMember	Mitglied aus einer Gruppe entfernen

Weitere wichtige Commandlets

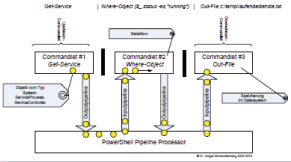
Get-Date / Set-Date	Datum und Zeit abrufen/festlegen
Get-Service	Windows-Systemdienste
Start-/Stop-/Suspend-/Resume-Service	Dienststatus ändern
Get-Process	Laufende Prozesse
Start-Process/Stop-Process	Process starten/beenden
Wait-Process	Warten auf Ende eines Prozesses
Get-Counter	Leistungsindikatoren abrufen
Get-EventLog	Ereignisprotokolleinträge
Write-EventLog	Eintrag im Ereignisprotokoll
Limit-EventLog	Größe des Ereignisprotokolls setzen
Get-Random	Zufallszahl
Find-Module	Module in PowerShell Gallery suchen
Install-Module	Module aus PowerShell Gallery herunterladen und installieren

Pipelining-Grundkonzept

Beliebig viele Commandlets können mit dem Pipe-Symbol | verkettet werden.
Get-Service | Where-Object {$_.status -eq "running"} | Out-File c:\temp\laufendedienste.txt

Alternativ kann man Zwischenergebnisse in Variablen, die mit $ beginnen, ablegen.
$dienste = Get-Service | Where-Object {$_.status -eq "running"}
$dienste | Out-File c:\temp\laufendedienste.txt

Die Pipeline befördert.NET-Objekte. Die Beförderung ist asynchron (außer bei einigen „blockierenden" Commandlets wie Sort-Object).



Wichtige Pipelining-Commandlets

Where-Object (where, ?)	Filtern mit Bedingungen
Select-Object (select)	Abschneiden der Ergebnismenge vorne/hinten bzw. Reduktion der Attribute der Objekte
Sort-Object (sort)	Sortieren der Objekte
Group-Object (group)	Gruppieren der Objekte
Foreach-Object { $_ … } (%)	Schleife über alle Objekte
Get-Member (gm)	Ausgabe der Metadaten (Reflection)
Measure-Object (measure)	Berechnung: -min -max -sum -average
Compare-Object (compare, diff)	Vergleichen von zwei Objektmengen

Vergleichsoperatoren

Da die Zeichen < und > für Umleitungen der Ausgabeströme verwendet werden, kommen PowerShell eher ungewöhnliche Operatoren zum Einsatz.

Vergleich unter Ignorierung der Groß-/Kleinschreibung	Vergleich unter Berücksichtigung der Groß-/Kleinschreibung	Bedeutung
-lt / -ilt	-clt	Kleiner
-le / -ile	-cle	Kleiner oder gleich
-gt / -igt	-cgt	Größer
-ge / -ige	-cge	Größer oder gleich
-eq / -ieq	-ceq	Gleich
-ne / -ine	-cne	Nicht gleich
-like / -ilike	-clike	Ähnlichkeit zwischen Zeichenketten, Einsatz von Platzhaltern (* und ?) möglich
-notlike / -inotlike	-cnotlike	Keine Ähnlichkeit zwischen Zeichenketten, Einsatz von Platzhaltern (* und ?) möglich
-match / -imatch	-cmatch	Vergleich mit regulärem Ausdruck

Vorderseite der PowerShell-Kurzreferenz

Kurzreferenz ("Cheat Sheet") Windows PowerShell

Autor: Dr. Holger Schwichtenberg (www.IT-Visions.de) V1.5.2 / 22.03.2018

www.IT-Visions.de
Dr. Holger Schwichtenberg

-notmatch / -inotmatch	-cnotmatch	Stimmt nicht mit regulärem Ausdruck überein
-is		Typvergleich, z.B. (Get-Date) -is [DateTime]
-in / -contains		Ist enthalten in Menge
-notin / -notcontains		Ist nicht enthalten in Menge

Für die logische Verknüpfung werden –and und –or sowie –not (alias !) verwendet.
Beispiel: *((1 MB + 150 + $a) -gt 1000KB) -and !($a -le 2KB)*
KB, MB, GB, TB und PB sind gültige Abkürzungen für Speichergrößen.

Ein- und Ausgabe-Commandlets

Format-Table (ft)	Tabellenausgabe
Format-List (fl)	detaillierte Liste
Format-Wide (fw)	mehrspaltige Liste
Out-Host (oh)	Ausgabe an Konsolen mit Optionen zur Farbe und seitenweisen Ausgabe
Out-GridView (ogv)	Grafische Tabelle mit Filter- und Sortieroptionen
Out-File	Speichern in Datei
Out-Printer (lp)	Ausgabe an Drucker
Out-Clipboard	Ausgabe in Zwischenablage
Out-Speech	Sprachausgabe (MSCN)
Out-Null	Die Objekte der Pipeline werden nicht weitergegeben
Read-Host	Eingaben von Konsole einlesen
Import-/Export-CSV	CSV-Datei importieren/exportieren
Import-/Export-CLIXML	XML-Datei importieren/exportieren

Benutzerdefinierte Tabellenausgabe
Get-Process | ft @{Label="Nr"; Expression={$_.ID}; Width=5},
@{Label="Name"; Expression={$_.Processname}; Width=20 },
@{Label="Speicher MB"; Expression={$_.WorkingSet64 / 1MB}; Width=11; Format="{0:00000.0}" }

Zeichenketten und Ausdrücke

Einbetten einer Variablen in eine Zeichenkette
"Der Befehl ist $Befehl!"
Hier muss { } zur Abgrenzung vom Doppelpunkt eingesetzt werden
"${Befehl}: erfolgreich ausgeführt"
Der Unterausdruck muss in $( ) geklammert werden
"$($Ergebnis.Count) Objekte in der Ergebnismenge"
Einsatz des Formatoperators
Get-Process | % { "{0,-40} | {1:0,000.00}MB" -f $_.Name, ($_.ws/1MB) }
Ausführen einer Zeichenkette als Befehl
$Befehl = "Get-Service a"*
$Befehl += " | where status -eq 'Running'"
$Ergebnis = Invoke-Expression $Befehl

Objektorientierter Zugriff auf Pipeline-Objekte

Anzahl der Objekte in der Pipeline
(Get-Service | where { $_.status -eq "Running" }).Count

Einzelne Eigenschaften der Pipeline-Objekte ausgeben
(Get-Date).DayOfWeek
(Get-Process).Name
(Get-Process | sort ws -desc)[0].Name

Methodenaufruf in allen Pipeline-Objekten
(Get-Process iexplore | sort ws -desc).Kill()

PowerShell-Datentypen

[char], [string]	[byte], [int], [long]	[XML]
[bool]	[single], [double]	[Array], [Hashtable]
[DateTime]		[WMI], [ADSI]

PowerShell-Skriptsprache

Bedingung
if ((Get-Date).Year -le 2014) { "Alt" } else { "Neu" }

Schleifen
for($i = 1; $i -le 10; $i++) { $i }
while($i -le 10) { $i; $i++ }
do { $i; $i++ } while ($i -le 10)
foreach ($p in (Get-Process iexplore)) { $p.Kill() }

Unterroutinen mit Pflichtparameter und optionalem Parameter
function Get-DLL([Parameter(Mandatory=$true)][string]$root, [string]$filter = "")*
{
return Get-ChildItem $root -Filter "$filter.dll"
}
Get-DLL c:\Windows\System32

Kommentar
 # Dies ist ein Kommentar

.NET Framework-Klassen

PowerShell kann alle auf dem lokalen System vorhandenen .NET-Klassen auch direkt (d.h. ohne Einsatz von Commandlets) verwenden.

Zugriff auf statische Mitglieder
[System.Environment]::MachineName
[System.Console]::Beep(800, 500)

Instanziierung und Zugriff auf Instanzmitglieder
$b = New-Object System.DirectoryServices.DirectoryEntry("WinNT://Server/HS")
$b.FullName
$b.Description = "Autor des PowerShell Cheat Sheets"
$b.SetInfo()

Zusätzliche Assembly laden und nutzen
[System.Reflection.Assembly]::LoadWithPartialName("Microsoft.VisualBasic")
$eingabe = [Microsoft.VisualBasic.Interaction]::InputBox("Frage","Titel")

Component Object Model (COM)

PowerShell kann alle installierten COM-Komponenten verwenden.
$ie = New-Object -com "InternetExplorer.Application"
$ie.Navigate("http://www.powershell-doktor.de")
$ie.visible = $true

Windows Management Instrumentation (WMI)

PowerShell kann alle lokalen oder entfernten WMI-Klassen verwenden.

Liste aller WMI-Klassen aus einem Namensraum von einem Computer
Get-CimClass -Namespace root/cimv2 -Computer MyServer

Liste aller Instanzen einer WMI-Klassen auf einem Computer
Get-CimInstance Win32_LogicalDisk -Namespace root/cimv2 -Computer MyServer

WQL-Abfrage auf einem Computer
*Get-CimInstance -Query "Select * from Win32_NetworkAdapter where adaptertype like '%802%'" -Computer MyServer*

Zugriff auf eine Instanz und Änderung der Instanz
$c = Get-CimInstance Win32_LogicalDisk -Namespace root/cimv2 -Filter "DeviceID='C:'" -Computer MyServer
$c.VolumeName = "System"
Get-CimInstance $c

Alternativ mit alten WMI-Commandlets
$c = [WMI] "\\MyServer\root\cimv2:Win32_LogicalDisk.DeviceID='C:'"
$c.VolumeName = "System"
$c.Put()

Aufruf einer WMI-Methode
Invoke-CimMethod -Path
"\\MyServer\root\cimv2:Win32_Computersystem.Name=MyServer" -Name
"Rename" -ArgumentList "MyNewServer"

Links

technet.microsoft.com/scriptcenter
blogs.msdn.com/PowerShell
www.powershell.com
www.powershell.de
www.it-visions.de/scripting/powershell

Über den Autor

Dr. Holger Schwichtenberg gehört zu den bekanntesten Experten für die Programmierung mit Microsoft-Produkten in Deutschland. Er hat zahlreiche Bücher zu .NET und PowerShell veröffentlicht und spricht regelmäßig auf Fachkonferenzen. Er hat mehrere Bücher zur PowerShell geschrieben. Sie können ihn und sein Team für Schulungen, Beratungen und Projekte buchen.
E-Mail: anfragen@IT-Visions.de

Rückseite der PowerShell-Kurzreferenz