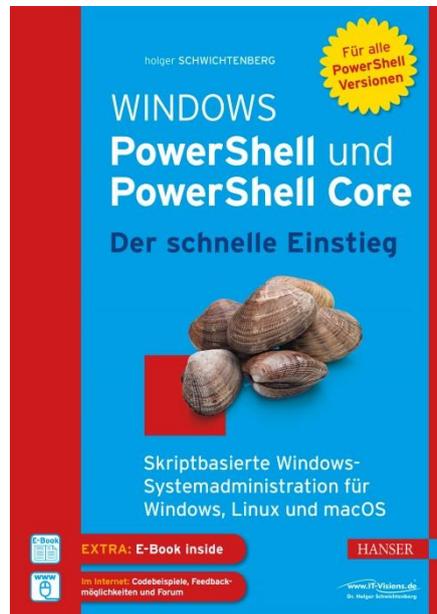


# HANSER



## Leseprobe

zu

# Windows PowerShell und PowerShell Core Der schnelle Einstieg

von Holger Schwichtenberg

ISBN (Buch): 978-3-446-45214-5

ISBN (E-Book): 978-3-446-45281-7

Weitere Informationen und Bestellungen unter

<https://www.hanser-fachbuch.de/>

sowie im Buchhandel

© Carl Hanser Verlag, München

# Inhalt

<b>Vorwort</b> .....	<b>XV</b>
<b>Über den Autor Dr. Holger Schwichtenberg</b> .....	<b>XXI</b>
<b>1 Erste Schritte mit der PowerShell</b> .....	<b>1</b>
1.1 Was ist die PowerShell? .....	1
1.2 Windows PowerShell versus PowerShell Core .....	2
1.3 Windows PowerShell herunterladen und auf anderen Windows- Betriebssystemen installieren .....	2
1.4 Die Windows PowerShell testen .....	7
1.4.1 PowerShell im interaktiven Modus .....	7
1.4.2 Installierte Version ermitteln .....	10
1.4.3 PowerShell im Skriptmodus .....	11
1.4.4 Skript eingeben .....	11
1.4.5 Skript starten .....	12
1.4.6 Skriptausführungsrichtlinie ändern .....	13
1.4.7 Farben ändern .....	16
1.5 Woher kommen die Commandlets? .....	17
1.6 PowerShell Community Extensions (PSCX) herunterladen und installieren .	18
1.7 Den Windows PowerShell-Editor „ISE“ verwenden .....	20
<b>2 Fakten zur PowerShell</b> .....	<b>23</b>
2.1 Geschichte der PowerShell .....	23
2.2 Warum PowerShell einsetzen? .....	24
2.3 Einflussfaktoren auf die Entwicklung der PowerShell .....	28
2.4 Betriebssysteme mit vorinstallierter PowerShell .....	29
2.5 Anbindung an Klassenbibliotheken .....	31
2.6 PowerShell versus WSH .....	31

<b>3</b>	<b>Einzelbefehle der PowerShell</b>	<b>35</b>
3.1	Commandlets	35
3.1.1	Aufbau eines Commandlets	35
3.1.2	Aufruf von Commandlets	36
3.1.3	Commandlet-Parameter	36
3.1.4	Platzhalter bei den Parameterwerten	39
3.1.5	Abkürzungen für Parameter	40
3.1.6	Allgemeine Parameter (Common Parameters)	41
3.1.7	Dynamische Parameter	45
3.1.8	Zeilenumbrüche	45
3.1.9	PowerShell-Module	46
3.1.10	Prozessmodell	47
3.1.11	Aufruf von Commandlets aus anderen Prozessen heraus	47
3.1.12	Namenskonventionen	48
3.2	Aliase	48
3.2.1	Aliase auflisten	49
3.2.2	Neue Aliase anlegen	53
3.2.3	Aliase für Eigenschaften	54
3.3	Ausdrücke	56
3.4	Externe Befehle	57
3.5	Dateinamen	59
3.6	Aufgaben zu diesem Kapitel	59
<b>4</b>	<b>Hilfefunktionen</b>	<b>61</b>
4.1	Auflisten der verfügbaren Befehle	61
4.2	Volltextsuche	63
4.3	Erläuterungen zu den Befehlen	64
4.4	Hilfe zu Parametern	65
4.5	Hilfe mit Show-Command	67
4.6	Hilfefenster	69
4.7	Allgemeine Hilfetexte	70
4.8	Aktualisieren der Hilfedateien	70
4.9	Online-Hilfe	72
4.10	Fehlende Hilfetexte	73
4.11	Dokumentation der .NETKlassen	75
4.12	Aufgaben zu diesem Kapitel	78
<b>5</b>	<b>Objektorientiertes Pipelining</b>	<b>79</b>
5.1	Pipeline-Operator	79
5.2	.NET-Objekte in der Pipeline	80
5.3	Pipeline Processor	82
5.4	Pipelining von Parametern	83

5.5	Pipelining von klassischen Befehlen	86
5.6	Anzahl der Objekte in der Pipeline	87
5.7	Zeilenumbrüche in Pipelines	88
5.8	Zugriff auf einzelne Objekte aus einer Menge	88
5.9	Zugriff auf einzelne Werte in einem Objekt	90
5.10	Methoden ausführen	92
5.11	Analyse des Pipeline-Inhalts	94
5.12	Filtern	105
5.13	Zusammenfassung von Pipeline-Inhalten	108
5.14	„Kastrierung“ von Objekten in der Pipeline	109
5.15	Sortieren	110
5.16	Duplikate entfernen	111
5.17	Gruppierung	112
5.18	Berechnungen	114
5.19	Zwischenschritte in der Pipeline mit Variablen	114
5.20	Verzweigungen in der Pipeline	115
5.21	Vergleiche zwischen Objekten	117
5.22	Zusammenfassung	118
5.23	Aufgaben zu diesem Kapitel	119
<b>6</b>	<b>PowerShell-Skripte</b>	<b>121</b>
6.1	Skriptdateien	121
6.2	Start eines Skripts	123
6.3	Aliase für Skripte verwenden	125
6.4	Parameter für Skripte	125
6.5	Skripte dauerhaft einbinden (Dot Sourcing)	127
6.6	Das aktuelle Skriptverzeichnis	127
6.7	Sicherheitsfunktionen für PowerShell-Skripte	128
6.8	Anforderungsdefinitionen von Skripten	130
6.9	Skripte anhalten	131
6.10	Versionierung und Versionsverwaltung von Skripten	131
6.10.1	Versionierung	131
6.10.2	Versionsverwaltung (Versionskontrolle)	132
6.11	Aufgaben zu diesem Kapitel	133
<b>7</b>	<b>Die PowerShell-Skriptsprache</b>	<b>135</b>
7.1	Hilfe zur PowerShell-Skriptsprache	135
7.2	Befehlstrennung	136
7.3	Kommentare	136
7.4	Variablen	137
7.4.1	Variablen in der PowerShell	137

7.4.2	Typisierung	138
7.4.3	Datentypen/Typbezeichner in PowerShell	139
7.4.4	Typisierungszwang	141
7.4.5	Typkonvertierung (Typumwandlung)	142
7.4.6	Gültigkeitsbereiche (Scope)	143
7.4.7	Variablen leeren oder löschen	144
7.4.8	Variablentyp ermitteln	145
7.4.9	Vordefinierte Variablen	145
7.5	Variablenbedingungen	147
7.6	Zahlen	148
7.7	Zeichenketten (Strings)	150
7.7.1	Zeichenkettenliterals	150
7.7.2	Zeichenketten zusammensetzen	151
7.7.3	Variablenuflösung in Zeichenketten	151
7.7.4	Wiederholte Zeichenketten	153
7.7.5	Leere Zeichenketten	153
7.7.6	Sonderzeichen in Zeichenketten	154
7.7.7	Bearbeitungsmöglichkeiten für Zeichenketten	155
7.7.8	Zeichenketten ersetzen	157
7.7.9	Zeichenketten trennen und verbinden	158
7.8	Reguläre Ausdrücke	159
7.8.1	Mustervergleichsoperatoren	159
7.8.2	Allgemeiner Aufbau von regulären Ausdrücken	161
7.9	Datum und Uhrzeit	165
7.10	Arrays	167
7.10.1	Deklaration	167
7.10.2	Arrayoperationen	167
7.10.3	Array auflisten	169
7.10.4	Arrays verbinden	169
7.10.5	Mehrdimensionale Arrays	170
7.11	ArrayList	170
7.12	Assoziative Arrays (Hash-Tabellen)	171
7.13	Operatoren	172
7.13.1	Vergleichsoperatoren	172
7.13.2	Arithmetische Operatoren	172
7.13.3	Zuweisungsoperator	173
7.13.4	Bit-Operatoren	175
7.13.5	Aufrufoperator	175
7.14	Überblick über die Kontrollkonstrukte	176
7.15	Schleifen	177
7.16	Bedingungen	182
7.17	Unterroutinen (Prozedur/Funktionen)	184
7.17.1	Prozedur versus Funktion	184

7.17.2 Prozeduren . . . . .	185
7.17.3 Funktionen (mit Rückgabewerten) . . . . .	185
7.17.4 Art der Rückgabewerte . . . . .	188
7.17.5 Parameterübergabe . . . . .	188
7.18 Eingebaute Funktionen . . . . .	190
7.19 Fehlerbehandlung . . . . .	191
7.20 Objektorientiertes Programmieren mit Klassen . . . . .	200
7.21 Aufgaben zu diesem Kapitel . . . . .	202
<b>8 Ausgaben . . . . .</b>	<b>203</b>
8.1 Ausgabe-Commandlets . . . . .	203
8.2 Benutzerdefinierte Tabellenformatierung . . . . .	206
8.3 Benutzerdefinierte Listenausgabe . . . . .	209
8.4 Mehrspaltige Ausgabe . . . . .	209
8.5 Out-GridView . . . . .	210
8.6 Standardausgabe . . . . .	212
8.7 Einschränkung der Ausgabe . . . . .	215
8.8 Seitenweise Ausgabe . . . . .	216
8.9 Ausgabe einzelner Werte . . . . .	217
8.10 Details zum Ausgabeoperator . . . . .	220
8.11 Ausgabe von Methodenergebnissen und Unterobjekten in Pipelines . . . . .	223
8.12 Ausgabe von Methodenergebnissen und Unterobjekten in Zeichenketten . . . . .	224
8.13 Unterdrückung der Ausgabe . . . . .	224
8.14 Ausgaben an Drucker . . . . .	225
8.15 Ausgaben in Dateien . . . . .	225
8.16 Umleitungen (Redirection) . . . . .	226
8.17 Fortschrittsanzeige . . . . .	227
8.18 Sprachausgabe . . . . .	227
8.19 Aufgaben zu diesem Kapitel . . . . .	228
<b>9 Benutzereingaben . . . . .</b>	<b>229</b>
9.1 Read-Host . . . . .	229
9.2 Benutzerauswahl . . . . .	230
9.3 Grafischer Eingabedialog . . . . .	231
9.4 Dialogfenster . . . . .	232
9.5 Authentifizierungsdialg . . . . .	232
9.6 Zwischenablage (Clipboard) . . . . .	234
9.7 Aufgaben zu diesem Kapitel . . . . .	235

<b>10</b>	<b>Das PowerShell-Navigationsmodell (PowerShell Provider)</b>	<b>237</b>
10.1	Einführungsbeispiel: Navigation in der Registrierungsdatenbank	237
10.2	Provider und Laufwerke	239
10.3	Navigationsbefehle	241
10.4	Pfadangaben	241
10.5	Beispiel	243
10.6	Eigene Laufwerke definieren	244
10.7	Aufgaben zu diesem Kapitel	245
<b>11</b>	<b>Fernauführung (Remoting)</b>	<b>249</b>
11.1	RPC-Fernabfrage ohne WS-Management	250
11.2	Anforderungen an PowerShell Remoting	251
11.3	Rechte für PowerShell-Remoting	252
11.4	Einrichten von PowerShell Remoting	253
11.5	Überblick über die Fernauführungs-Commandlets	255
11.6	Interaktive Fernverbindungen im Telnet-Stil	256
11.7	Fernauführung von Befehlen	257
11.8	Parameterübergabe an die Fernauführung	261
11.9	Fernauführung von Skripten	262
11.10	Ausführung auf mehreren Computern	263
11.11	Sitzungen	264
11.11.1	Commandlets zur Sitzungsverwaltung	265
11.11.2	Sitzungen erstellen	266
11.11.3	Kopieren von Dateien in Sitzungen	266
11.11.4	Schließen von Sitzungen	267
11.11.5	Sitzungskonfigurationen	267
11.11.6	Zugriffsrechte für Fernaufrufe	267
11.12	Implizites Remoting	269
11.13	Zugriff auf entfernte Computer außerhalb der eigenen Domäne	270
11.14	Verwaltung des WS-Management-Dienstes	274
11.15	PowerShell Direct für Hyper-V	275
11.16	Praxisbeispiel zu PowerShell Direct	277
11.17	Aufgaben zu diesem Kapitel	279
<b>12</b>	<b>Verwendung von .NET-Klassen</b>	<b>281</b>
12.1	Microsoft Developer Network (MSDN)	281
12.2	Erzeugen von Instanzen	282
12.3	Parameterbehaftete Konstruktoren	284
12.4	Initialisierung von Objekten	286
12.5	Nutzung von Attributen und Methoden	286
12.6	Statische Mitglieder in .NET-Klassen und statische .NET-Klassen	289

12.7	Generische Klassen nutzen .....	293
12.8	Zugriff auf bestehende Objekte .....	294
12.9	Laden von Assemblies .....	294
12.10	Objektanalyse .....	297
12.11	Auflistungen (Enumerationen) .....	298
12.12	Verknüpfen von Aufzählungswerten .....	299
12.13	Aufgaben zu diesem Kapitel .....	299
<b>13</b>	<b>Verwendung von COM-Klassen .....</b>	<b>301</b>
13.1	Erzeugen von COM-Instanzen .....	301
13.2	Nutzung von Attributen und Methoden .....	302
13.3	Liste aller COM-Klassen .....	303
13.4	Holen bestehender COM-Instanzen .....	304
13.5	Distributed COM (DCOM) .....	304
13.6	Aufgaben zu diesem Kapitel .....	305
<b>14</b>	<b>Zugriff auf die Windows Management Instrumentation (WMI) ..</b>	<b>307</b>
14.1	WMI in der PowerShell .....	307
14.2	Open Management Infrastructure (OMI) .....	309
14.3	Abruf von WMI-Objektmengen .....	309
14.4	Fernzugriffe .....	310
14.5	Filtern und Abfragen .....	311
14.6	Liste aller WMI-Klassen .....	315
14.7	Hintergrundwissen: WMI-Klassenprojektion mit dem PowerShell-WMI-Objektadapter .....	315
14.8	Beschränkung der Ausgabeliste bei WMI-Objekten .....	320
14.9	Zugriff auf einzelne Mitglieder von WMI-Klassen .....	322
14.10	Werte setzen in WMI-Objekten .....	322
14.11	Umgang mit WMI-Datumsangaben .....	324
14.12	Methodenaufrufe .....	325
14.13	Neue WMI-Instanzen erzeugen .....	326
14.14	Instanzen entfernen .....	327
14.15	Commandlet Definition XML-Datei (CDXML) .....	328
14.16	Aufgaben zu diesem Kapitel .....	330
<b>15</b>	<b>Fehlersuche .....</b>	<b>333</b>
15.1	Detailinformationen .....	333
15.2	Einzelschrittmodus .....	335
15.3	Zeitmessung .....	336
15.4	Ablaufverfolgung (Tracing) .....	336
15.4.1	Tracesources .....	336

15.4.2	Verfolgung eines Einzelbefehls	337
15.4.3	Generelle Ablaufverfolgung	337
15.5	Erweiterte Protokollierung aktivieren	338
15.6	Script-Debugging in der ISE	339
15.7	Kommandozeilenbasiertes Script-Debugging	340
<b>16</b>	<b>Standardeinstellungen ändern mit Profilskripten</b>	<b>343</b>
16.1	Profilpfade	343
16.2	Ausführungsreihenfolge	345
16.3	Beispiel für eine Profildatei	345
16.4	Starten der PowerShell ohne Profilskripte	346
<b>17</b>	<b>PowerShell-Module</b>	<b>347</b>
17.1	Überblick über die Commandlets	347
17.2	Modularchitektur	348
17.3	Aufbau eines Moduls	349
17.4	Module aus dem Netz herunterladen und installieren mit PowerShellGet	350
17.5	Module manuell installieren	357
17.6	Doppeldeutige Namen	357
17.7	Importieren von Modulen	360
17.8	Entfernen von Modulen	363
<b>18</b>	<b>Praxislösungen</b>	<b>365</b>
18.1	Leere Ordner löschen	365
18.2	Fotos nach Aufnahmedatum sortieren	366
18.3	Zufällige Dateisystemstruktur erzeugen	368
18.4	Freigaben anlegen	369
18.4.1	WMI-Klassen	369
18.4.2	Freigaben auflisten	371
18.4.3	Freigaben anlegen (mit WMI)	371
18.4.4	Berechtigungen auf Freigaben setzen	373
18.4.5	Freigaben anlegen (mit PowerShell-Commandlets)	376
18.4.6	Freigaben anlegen auf Basis einer XML-Datei	379
18.5	Netzwerkconfiguration	381
18.6	Massenanlegen von Registry-Schlüsseln	383
18.7	Massenanlegen von Active-Directory-Benutzerkonten	384
18.7.1	Benutzer und Gruppen anlegen aus einer Datenbank (Microsoft Access oder Microsoft SQL Server)	385
18.7.2	Benutzer und Gruppen anlegen aus einer CSV-Datei	399
18.8	Massenanlegen von IIS-Websites	402

18.9	Softwareinstallation .....	404
18.10	Virtuelles System in Hyper-V anlegen .....	406
<b>19</b>	<b>PowerShell Core 6.0 für Windows, Linux und macOS .....</b>	<b>409</b>
19.1	Geschichte der PowerShell Core .....	409
19.2	Vergleich zwischen Windows PowerShell und PowerShell Core .....	410
19.3	Motivation für den Einsatz der PowerShell Core auf Linux und macOS ...	411
19.4	PowerShell Core installieren und testen .....	413
19.4.1	Installation und Test auf Windows .....	413
19.4.2	Installation und Test auf Ubuntu Linux .....	415
19.4.3	Installation und Test auf macOS .....	417
19.5	Funktionsumfang der PowerShell Core .....	418
19.6	Entfallene Commandlets in PowerShell Core .....	420
19.6.1	PowerShell-Kern-Befehle, die in PowerShell Core fehlen .....	420
19.6.2	Befehle, die zusätzlich unter Linux und macOS fehlen .....	423
19.7	Erweiterungsmodule nutzen in PowerShell Core .....	425
19.7.1	Eingebaute Module in PowerShell Core .....	425
19.7.2	Windows-Module in PowerShell Core nutzen .....	427
19.7.3	Module der PowerShell Gallery in PowerShell Core .....	427
19.8	Geänderte Funktionen in PowerShell Core .....	428
19.8.1	Änderungen der Parameter von pwsh.exe .....	428
19.8.2	Geänderte Pfade .....	428
19.8.3	Weitere Änderungen .....	430
19.9	Neue Funktionen der PowerShell Core .....	430
19.9.1	Neue eingebaute Variablen .....	430
19.9.2	Neue Commandlets .....	431
19.9.3	Sonstige Verbesserungen in PowerShell Core .....	431
19.10	PowerShell-Core-Konsole .....	432
19.11	VSCoDe-PowerShell als Editor für PowerShell Core .....	433
19.11.1	PowerShell Core zur Skriptausführung konfigurieren .....	434
19.11.2	PowerShell Core für das Terminalfenster in VSCoDe festlegen ...	436
19.12	Verwendung auf Linux und macOS .....	438
19.12.1	Praxisbeispiel: Linux-Benutzerliste auswerten .....	440
19.12.2	Praxisbeispiel: Offene Ports auswerten .....	441
19.12.3	Praxisbeispiel: Dateien unter Linux und macOS verstecken und versteckte Dateien auflisten .....	443
19.13	PowerShell-Remoting via SSH .....	444
19.13.1	OpenSSH auf Windows .....	444
19.13.2	PowerShell Remoting mit SSH .....	446
19.14	Dokumentation zur PowerShell Core .....	448
19.15	Quellcode zur PowerShell Core .....	450

20	Weiterführende Literatur.....	453
21	Lösungen.....	457
	Index .....	473

# Vorwort

Liebe Leserin, lieber Leser,

herzlich willkommen zu meinem Buch „Windows PowerShell und PowerShell Core – Der schnelle Einstieg: Skriptbasierte Systemadministration für Windows, Linux und macOS“!

## ■ Was ist das Konzept dieses Buchs?

Seit vielen Jahren veröffentliche ich sehr erfolgreich das „PowerShell-Praxisbuch“ im Carl Hanser Verlag, welches mittlerweile auf über 1200 Seiten angewachsen ist. Mit dem vor Ihnen liegenden Buch „PowerShell – Der schnelle Einstieg“ erhalten Sie eine deutlich kompaktere Einführung in die PowerShell. Gegenüber dem Praxisbuch enthält dieses Einsteigerbuch am Ende der meisten Kapitel Aufgaben mit Lösungen zur Kontrolle oder Verfestigung des Lernerfolgs.

Der Schwerpunkt dieses Buchs liegt auf der Windows PowerShell 5.1 im Einsatz auf Windows. Am Ende gibt es aber auch ein Kapitel, das die Unterschiede zu der plattformneutralen PowerShell Core 6.0 und ihrem Einsatz auf Linux und macOS beschreibt.

## ■ Wer bin ich?

Mein Name ist Holger Schwichtenberg, ich bin derzeit 45 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](http://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 5Minds IT-Solutions GmbH & Co. KG Software ([www.5Minds.de](http://www.5Minds.de)) im Auftrag von Kunden aus zahlreichen Branchen.

Es ist mein Hobby und Nebenberuf, IT-Fachbücher zu schreiben. Dieses Buch ist, unter Mitzählung aller nennenswerten Neuauflagen, das 68. Buch, das ich allein oder mit Co-Autoren geschrieben habe. Meine weiteren Hobbys sind Mountain Biking, Laufsport, 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?	<input type="radio"/> Ja	<input type="radio"/> Nein
Laufen die für Sie relevanten Computer mit den von PowerShell 3.0, 4.0, 5.x oder 6.x unterstützten Betriebssystemen? (Windows 7/8/8.1/10, Windows Server 2008/2008 R2/2012/2012 R2/2016) Hinweis: Die PowerShell Core 6.0 für Linux und macOS wird nur als Randthema kurz in diesem Buch behandelt, da es hier bislang kaum Befehle für die PowerShell gibt!	<input type="radio"/> Ja	<input type="radio"/> Nein
Sie besitzen zumindest rudimentäre Grundkenntnisse im Bereich des (objektorientierten) Programmierens?	<input type="radio"/> Ja	<input type="radio"/> Nein
Wünschen Sie einen kompakten Überblick über die Architektur, Konzepte und Anwendungsfälle der PowerShell?	<input type="radio"/> Ja	<input type="radio"/> Nein
Sie können auf Schritt-für-Schritt-Anleitungen verzichten?	<input type="radio"/> Ja	<input type="radio"/> Nein
Sie können auf formale Syntaxbeschreibungen verzichten und lernen lieber an aussagekräftigen Beispielen?	<input type="radio"/> Ja	<input type="radio"/> Nein
Sie erwarten nicht, dass in diesem Buch <b>alle</b> Möglichkeiten der PowerShell/PowerShell Core detailliert beschrieben werden?	<input type="radio"/> Ja	<input type="radio"/> Nein
Sind Sie, nachdem Sie ein Grundverständnis durch dieses Buch gewonnen haben, bereit, Detailfragen in der Dokumentation der PowerShell, von .NET und WMI (oder meinem umfangreicheren „PowerShell Praxisbuch“) nachzuschlagen, da das Buch auf rund 400 Seiten nicht alle Details erläutern kann?	<input type="radio"/> Ja	<input type="radio"/> Nein

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

## ■ Sind in diesem Buch alle Features der PowerShell beschrieben?

Die PowerShell umfasst mittlerweile über 1500 Commandlets mit jeweils zahlreichen Optionen. Zudem gibt es unzählige Erweiterungen mit vielen hundert weiteren Commandlets. Zudem 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 eingesetzt. Ich beschreibe in diesem Buch, was ich selbst in der Praxis, in meinen Schulungen und bei Kundeneinsätzen verwende. Es macht auch keinen Sinn, jedes Detail der PowerShell hier zu dokumentieren. Stattdessen gebe ich Ihnen **Hilfe zur Selbsthilfe**, damit Sie die Konzepte gut verstehen und sich dann Sonderfälle 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 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 häufig. Aufwärts- und Abwärtskompatibilität ist kein Ziel 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.
- Produkte werden schnell auch wieder abgekündigt, wenn sie sich aus der Sicht der Hersteller bzw. aufgrund des Nutzerfeedbacks nicht bewährt haben.

Unter diesen neuen Einflussströmen steht natürlich auch dieses Buch. Leider kann man ein gedrucktes Buch nicht so schnell ändern wie Software. Verlage definieren erhebliche Mindestauflagen, die abverkauft werden müssen, bevor neu gedruckt werden darf. Das E-Book ist keine Alternative. Die Verkaufszahlen zeigen, dass nur eine verschwindend 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 im Leser-Portal (siehe unten). Ich werde dies dann in Neuauflagen des Buchs berücksichtigen.

## ■ Wem ist zu danken?

Folgenden Personen möchte ich meinen Dank für ihre Mitwirkung an diesem Buch aussprechen:

- Frau Sylvia Hasselbach, die mich schon seit 20 Jahren als Lektorin begleitet und die dieses Buchprojekt beim Carl Hanser Verlag koordiniert und vermarktet,
- Frau Petra Kienle, die meine Tippfehler gefunden und sprachliche Ungenauigkeiten eliminiert hat,
- meiner Frau und meinen Kindern dafür, dass sie mir das Umfeld geben, um neben meinem Hauptberuf an Büchern wie diesem zu arbeiten.

## ■ Woher bekommen Sie die Beispiele aus diesem Buch?

Unter <http://www.powershell-doktor.de/leser> biete ich ein **ehrenamtlich betriebenes** Webportal für Leser meiner Bücher an. 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,
- Feedback zu diesem Buch geben (Bewertung abgeben und Fehler melden) und
- technische Fragen in einem Webforum stellen.

Alle registrierten Leser erhalten auch Einladungen zu kostenlosen Community-Veranstaltungen sowie Vergünstigungen bei unseren öffentlichen Seminaren zu .NET und zur PowerShell. Bei der Registrierung müssen Sie das Kennwort **Die letzten Jedi** angeben.

## ■ Wie sind die Programmcodebeispiele organisiert?

Sie erhalten die Beispiele zu diesem Buch in Form einer RAR-Archivdatei. Die Beispiele sind im Archiv organisiert nach Kapitelnamen (verkürzt). In diesem Buch wird für den Zugriff auf die Beispieldateien das X:-Laufwerk verwendet. Bitte legen Sie entweder ein Laufwerk X: an oder ändern Sie den Laufwerksbuchstaben in den Skripten.

```
PowerShell
PS x:\> Dir

Directory: W:\PSE_Skripte

Mode                LastWriteTime         Length Name
----                -
d-----          04.03.2018         20:23     =Praxislösungen
d-----          29.06.2017         23:56     Aliase
d-r---          05.07.2017         09:43     Ausgaben
d-----          02.07.2017         23:29     Benutzereingaben
d-r---          21.04.2017         19:13     COM
d-r---          30.05.2017         00:28     Commandlets
d-r---          04.03.2018         20:22     DOTNET
d-----          08.03.2018         18:21     ErsteSchritte
d-----          24.04.2017         09:55     Fehlersuche
d-r---          29.06.2017         23:34     Hilfe
d-r---          04.03.2018         18:08     Module
d-r---          26.03.2014         12:49     Navigation
d-r---          04.06.2017         11:21     Pipelining
d-----          22.09.2017         19:14     PowerShellLanguage
d-----          29.05.2017         23:57     PowerShellOOP
d-r---          30.06.2017         20:26     Profile
d-----          07.03.2018         18:14     PSCore
d-----          04.07.2017         17:52     Remoting
d-r---          30.08.2017         13:24     Scripting
d-r---          04.11.2017         07:19     Werkzeuge
d-r---          17.05.2016         13:28     WMI
d-r---          26.03.2014         12:49     WPS versus VBS

PS x:\> █
```

## ■ Wo können Sie sich schulen oder beraten lassen?

Unter der E-Mail-Adresse [kundenteam@IT-Visions.de](mailto:kundenteam@IT-Visions.de) stehen mein Team und ich für Anfragen bezüglich Schulung, Beratung und Entwicklungstätigkeiten zur Verfügung – nicht nur zum Thema PowerShell und .NET/.NET Core, sondern zu fast allen modernen Techniken der Entwicklung und des Betriebs von Software in großen Unternehmen. Wir besuchen Sie gerne in Ihrem Unternehmen an einem beliebigen Standort.

## ■ Zum Schluss des Vorworts ...

... wünsche ich Ihnen viel Spaß und Erfolg mit der PowerShell!

*Dr. Holger Schwichtenberg*

*Essen, im Frühjahr 2018*

# Über den Autor

## Dr. Holger Schwichtenberg



- Studienabschluss Diplom-Wirtschaftsinformatik an der Universität Essen
- Promotion an der Universität Essen im Gebiet komponentenbasierter Softwareentwicklung
- Seit 1996 selbstständig als unabhängiger Berater, Dozent, Softwarearchitekt und Fachjournalist
- Leiter des Berater- und Dozententeams bei *www.IT-Visions.de*
- Softwareprojektleiter im Bereich Microsoft/.NET bei der 5minds IT-Solutions GmbH & Co. KG in Gelsenkirchen (*www.5minds.de*)
- Über 65 Fachbücher beim Carl Hanser Verlag, bei O'Reilly, Microsoft Press und Addison-Wesley sowie mehr als 1000 Beiträge in Fachzeitschriften
- Gutachter in den Wettbewerbsverfahren der EU gegen Microsoft (2006–2009)
- Ständiger Mitarbeiter der Zeitschriften *iX* (seit 1999), *dotnetpro* (seit 2000) und *Windows Developer* (seit 2010) sowie beim Online-Portal *heise.de* (seit 2008)
- Regelmäßiger Sprecher auf nationalen und internationalen Fachkonferenzen (z.B. Microsoft TechEd, Microsoft Technical Summit, Microsoft IT Forum, BASTA, BASTA-on-Tour, .NET Architecture Camp, Advanced Developers Conference, Developer Week, OOP, DOTNET Cologne, MD DevDays, Community in Motion, DOTNET-Konferenz, VS One, NRW.Conf, Net.Object Days, Windows Forum)

  
www.IT-Visions.de®  
Dr. Holger Schwichtenberg

  
5Minds  
IT-SOLUTIONS

- Zertifikate und Auszeichnungen von Microsoft:
  - Bereits 14-mal ausgezeichnet als Microsoft Most Valuable Professional (MVP)
  - Zertifiziert als Microsoft Certified Solution Developer (MCSD)
- Thematische Schwerpunkte:
  - Softwarearchitektur, mehrschichtige Softwareentwicklung, Softwarekomponenten, SOA
  - Microsoft .NET Framework, Visual Studio, C#, Visual Basic
  - .NET-Architektur/Auswahl von .NET-Technologien
  - Einführung von .NET Framework und Visual Studio/Migration auf .NET
  - Webanwendungsentwicklung und Cross-Plattform-Anwendungen mit HTML, ASP.NET, ASP.NET Core, JavaScript/TypeScript und Webframeworks wie Angular
  - Enterprise .NET, verteilte Systeme/Webservices mit .NET insbes. Windows Communication Foundation und WebAPI
  - Relationale Datenbanken, XML, Datenzugriffsstrategien
  - Objektrelationales Mapping (ORM), insbesondere ADO.NET Entity Framework und EF Core
  - Windows PowerShell, PowerShell Core und Windows Management Instrumentation (WMI)
- Ehrenamtliche Community-Tätigkeiten:
  - Vortragender für die International .NET Association (INETA)
  - Betrieb diverser Community-Websites: [www.dotnetframework.de](http://www.dotnetframework.de), [www.entwickler-lexikon.de](http://www.entwickler-lexikon.de), [www.windows-scripting.de](http://www.windows-scripting.de), [www.aspnetdev.de](http://www.aspnetdev.de) u. a.
- Firmenwebsites: <http://www.IT-Visions.de> und <http://www.5minds.de>
- Weblog: <http://www.dotnet-doktor.de>
- Kontakt: [kundenteam@IT-Visions.de](mailto:kundenteam@IT-Visions.de) sowie Telefon 02 01-64 95 90-0

# 2

## Fakten zur PowerShell

### ■ 2.1 Geschichte der PowerShell

In der Vergangenheit war Active Scripting manchen Administratoren zu komplex, weil es viel Wissen über objektorientiertes Programmieren und das Component Object Model (COM) voraussetzt. Die vielen Ausnahmen und Ungereimtheiten im Active Scripting erschwerten das Erlernen von Windows Script Host (WSH) und der zugehörigen Komponentenbibliotheken.

Schon im Zuge der Entwicklung des Windows Server 2003 gab Microsoft zu, dass man Unix-Administratoren zum Interview über ihr tägliches Handwerkszeug gebeten hatte. Das kurzfristige Ergebnis war eine große Menge zusätzlicher Kommandozeilenwerkzeuge. Langfristig setzt Microsoft jedoch auf eine Ablösung des DOS-ähnlichen Konsolenfensters durch eine neue Skripting-Umgebung.

Mit dem Erscheinen des .NET Frameworks im Jahre 2002 wurde lange über einen WSH.NET spekuliert. Microsoft stellte jedoch die Neuentwicklung des WSH für das .NET Framework ein, als abzusehen war, dass die Verwendung von .NET-basierten Programmiersprachen wie C# und Visual Basic .NET dem Administrator nur noch mehr Kenntnisse über objektorientierte Softwareentwicklung abverlangen würde.

Microsoft beobachtete in der Unix-Welt eine hohe Zufriedenheit mit den dortigen Kommandozeilen-Shells und entschloss sich daher, das Konzept der Unix-Shells, insbesondere das Pipelining, mit dem .NET Framework zusammenzubringen und daraus eine .NET-basierte Windows Shell zu entwickeln. Diese sollte noch einfacher als eine Unix-Shell, aber dennoch so mächtig wie das .NET Framework sein.

In einer ersten Beta-Version wurde die neue Shell schon unter dem Codenamen „Monad“ auf der Professional Developer Conference (PDC) im Oktober 2003 in Los Angeles vorgestellt. Nach den Zwischenstufen „Microsoft Shell (MSH)“ und „Microsoft Command Shell“ trägt die neue Skriptumgebung seit Mai 2006 den Namen „Windows PowerShell“.

Die PowerShell 1.0 erschien am 6.11.2006 zeitgleich mit Windows Vista, war aber dort nicht enthalten, sondern musste heruntergeladen und nachinstalliert werden.

Die PowerShell 2.0 ist zusammen mit Windows 7/Windows Server 2008 R2 erschienen am 22.7.2009.

Die PowerShell 3.0 ist zusammen mit Windows 8/Windows Server 2012 erschienen am 15.8.2012.

Die PowerShell 4.0 ist zusammen mit Windows 8.1/Windows Server 2012 R2 am 9.9.2013 erschienen.

Die PowerShell 5.0 ist als Teil von Windows 10 erschienen am 29.7.2015. Abweichend von den bisherigen Gepflogenheiten ist die PowerShell 5.0 als Erweiterung für Windows Server 2008 R2 (mit Service Pack 1) und Windows Server 2012/2012 R2 erst deutlich später am 16.12.2015 erschienen. Für Windows 7 und Windows 8.1 sollte es erst gar keine Version mehr geben. Doch am 18.12.2015 hatte Microsoft ein Einsehen mit den Kunden und lieferte die PowerShell 5.0 auch für diese Betriebssysteme nach. Kurioserweise musste Microsoft den Download dann am 23.12.2015 wegen eines gravierenden Fehlers für einige Wochen vom Netz nehmen. Microsoft hatte das Produkt im neuen Agilitätswahn nicht richtig getestet.

Windows Server 2016 (erschieden am 26.9.2016) enthält PowerShell 5.1 und Windows 10 wurde mit dem Windows 10 Anniversary Update (Version 1607, Codename „Redstone 1“) am 2.8.2016 aktualisiert. PowerShell 5.1 ist erst seit 19.1.2017 als Add-on für Windows 7, Windows 8.1, Windows Server 2008 R2, Windows 2012 und Windows 2012 R2 verfügbar.

Eine reduzierte „Core“-Version der Windows PowerShell ist als „Windows PowerShell Core 5.1“ enthalten in Windows Nano Server, im ersten Release 2016 als Standardpaket, im zweiten in Release „1709“ als Option.

Die erste Version der plattformneutralen PowerShell Core (ohne Windows im Namen!) ist mit der Versionsnummer 6.0 am 20.01.2018 erschienen.



**HINWEIS:** Mit Windows 10 hat Microsoft das Auslieferungsverfahren auf „Windows as a Service“ umgestellt. Dies bedeutet, dass Microsoft über Windows Update im Sinne der neuen „agilen“ Strategie nun auch ständig neue Funktionen ausliefert. Dies betrifft ebenso die Windows PowerShell, die dann zukünftig auch auf diesem Wege häufigere Aktualisierungen erfahren kann. Wie häufig dies sein wird, ist zum Redaktionsschluss dieses Buchs noch offen.

Microsoft hat sich seit dem Jahr 2015 für andere Betriebssysteme und die Entwicklung als „Open Source Software“ (OSS) geöffnet. Dies betrifft nun auch die PowerShell: Die PowerShell Core, die am 20.1.2018 als Version 6.0 (in Nachfolge von Windows PowerShell 5.1) erschienen ist, ist OpenSource und läuft nicht nur auf Windows, sondern auch auf macOS und Linux.

## ■ 2.2 Warum PowerShell einsetzen?

Falls Sie eine Motivation brauchen, sich mit der PowerShell zu beschäftigen, wird dieses Kapitel Sie Ihnen liefern. Es stellt die Lösung für eine typische Scripting-Aufgabe sowohl im „alten“ Windows Script Host (WSH) als auch in der „neuen“ PowerShell vor und Sie werden schnell erkennen, welche Vorteile Ihnen die PowerShell bietet.

Zur Motivation, sich mit der PowerShell zu beschäftigen, soll folgendes Beispiel aus der Praxis dienen. Es soll ein Inventarisierungsskript für Software erstellt werden, das die installierten MSI-Pakete mit Hilfe der Windows Management Instrumentation (WMI) von mehreren Computern ausliest und die Ergebnisse in einer CSV-Datei (*softwareinventar.csv*) zusammenfasst. Die Namen (oder IP-Adressen) der abzufragenden Computer sollen in einer Textdatei (*computernamen.txt*) stehen.

Die Lösung mit dem WSH benötigt 90 Codezeilen (inklusive Kommentare und Parametrisierungen). In der PowerShell lässt sich das Gleiche in nur 13 Zeilen ausdrücken. Wenn man auf die Kommentare und die Parametrisierung verzichtet, dann reicht sogar genau eine Zeile. Das PowerShell-Skript läuft in der Windows PowerShell und auch in der PowerShell Core unter Windows, aber nicht unter Linux und macOS, da es dort noch keine Implementierung des für den Zugriff auf die installierte Software notwendigen Web Based Enterprise Management (WBEM) und des Common Information Model (CIM) für die PowerShell gibt.

### Listing 2.1 Softwareinventarisierung – Lösung 1 mit dem WSH

[3\_Einsatzgebiete/Software/Software\_Inventory.vbs]

```
' -----
' Skriptname: Software_inventar.vbs
' Autor: Dr. Holger Schwichtenberg
' -----
' Dieses Skript erstellt eine Liste
' der installierten Software
' -----
Option Explicit

' --- Einstellungen
Const Trennzeichen = ";" ' Trennzeichen für Spalten in der Ausgabedatei
Const Eingabedateiname = "computernamen.txt"
Const Ausgabedateiname = "softwareinventar.csv"
Const Bedingung = "SELECT * FROM Win32_Product where not Vendor like '%Microsoft%'"

Dim objFSO ' Dateisystem-Objekt
Dim objTX ' Textdatei-Objekt für die Liste der zu durchsuchenden computer
Dim i ' Zähler für Computer
Dim computer ' Name des aktuellen computers
Dim Eingabedatei ' Name und Pfad der Eingabedatei
Dim Ausgabedatei ' Name und Pfad der Ausgabedatei

' --- Startmeldung
WScript.Echo "Softwareinventar.vbs"
WScript.Echo "(C) Dr. Holger Schwichtenberg, http://www.Windows-Scripting.de"

' --- Global benötigtes Objekt
Set objFSO = CreateObject("Scripting.FileSystemObject")

' --- Ermittlung der Pfade
Eingabedatei = GetCurrentPfad & "\" & Eingabedateiname
Ausgabedatei = GetCurrentPfad & "\" & Ausgabedateiname

' --- Auslesen der computerliste
Set objTX = objFSO.OpenTextFile(Eingabedatei)

' --- Meldungen
```

```

WScript.Echo "Eingabedatei: " & Eingabedatei
WScript.Echo "Ausgabedatei: " & Ausgabedatei

' --- Überschriften einfügen
Ausgabe _
"computer" & Trennzeichen & _
"Name" & Trennzeichen & _
    "Beschreibung" & Trennzeichen & _
    "Identifikationsnummer" & Trennzeichen & _
    "Installationsdatum" & Trennzeichen & _
    "Installationsverzeichnis" & Trennzeichen & _
    "Zustand der Installation" & Trennzeichen & _
    "Paketzwischenspeicher" & Trennzeichen & _
    "SKU Nummer" & Trennzeichen & _
    "Hersteller" & Trennzeichen & _
    "Version"

' --- Schleife über alle Computer
Do While Not objTX.AtEndOfStream
    computer = objTX.ReadLine
    i = i + 1
    WScript.Echo "=== Computer #" & i & ": " & computer

GetInventar computer

Loop

' --- Eingabedatei schließen
objTX.Close
' --- Abschlußmeldung
WScript.Echo "Softwareinventarisierung beendet!"

' === Softwareliste für einen computer erstellen
Sub GetInventar(computer)

Dim objProduktMenge
Dim objProdukt
Dim objWMIDienst

' --- Zugriff auf WMI
Set objWMIDienst = GetObject("winmgmts:" &
    "{impersonationLevel=impersonate}!\\" & computer &
    "\root\cimv2")
' --- Liste anfordern
Set objProduktMenge = objWMIDienst.ExecQuery _
    (Bedingung)
' --- Liste ausgeben
WScript.Echo "Auf " & computer & " sind " &
objProduktMenge.Count & " Produkte installiert."
For Each objProdukt In objProduktMenge
    Ausgabe _
    computer & Trennzeichen & _
    objProdukt.Name & Trennzeichen & _
    objProdukt.Description & Trennzeichen & _
    objProdukt.IdentifyingNumber & Trennzeichen & _
    objProdukt.InstallDate & Trennzeichen & _
    objProdukt.InstallLocation & Trennzeichen & _
    objProdukt.InstallState & Trennzeichen & _

```

```

    objProdukt.PackageCache & Trennzeichen & _
    objProdukt.SKUNumber & Trennzeichen & _
    objProdukt.Vendor & Trennzeichen & _
    objProdukt.Version
WScript.Echo    objProdukt.Name
Next
End Sub

' === Ausgabe
Sub Ausgabe(s)
Dim objTextFile
' Ausgabedatei öffnen
Set objTextFile = objFSO.OpenTextFile(Ausgabedatei, 8, True)
objTextFile.WriteLine s
objTextFile.Close
'WScript.Echo s
End Sub

' === Pfad ermitteln, in dem das Skript liegt
Function GetCurrentPfad
GetCurrentPfad = objFSO.GetFile (WScript.ScriptFullName).ParentFolder
End Function

```

**Listing 2.2** Softwareinventarisierung – Lösung 2 als PowerShell-Skript  
 [3\_Einsatzgebiete/Software/SoftwareInventory\_WMI\_Script.ps1]

```

# Einstellungen
$InputFileName = "computernamen.txt"
$OutputFileName = "softwareinventar.csv"
$query = "SELECT * FROM Win32_Product where not Vendor like '%Microsoft%'"

# Eingabedatei auslesen
$Computers = Get-Content $InputFileName

# Schleife über alle Computer
$Software = $Computers | ForEach { Get-CimInstance -query $query -computername $_ }
# Ausgabe in CSV
$Software | select Name, Description, IdentifyingNumber, InstallDate,
InstallLocation, InstallState, SKUNumber, Vendor, Version | export-csv
$OutputFileName -notypeinformation

```

**Listing 2.3** Softwareinventarisierung – Lösung 3 als PowerShell-Pipeline-Befehl  
 [3\_Einsatzgebiete/Software/SoftwareInventory\_WMI\_Pipeline.ps1]

```

Get-Content "computers.txt" | ForEach {Get-CimInstance -computername $_ -query
"SELECT * FROM Win32_Product where not Vendor like '%Microsoft%'" } | export-csv
"Softwareinventory.csv" -notypeinformation

```

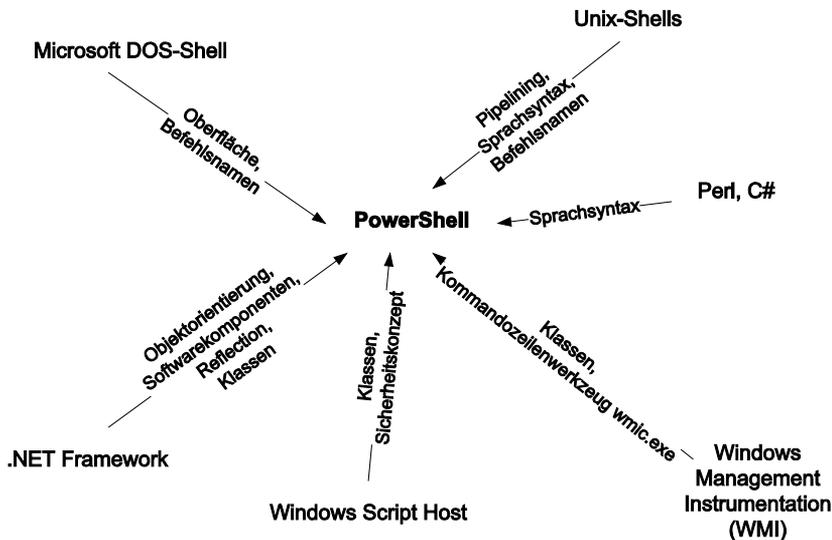
## ■ 2.3 Einflussfaktoren auf die Entwicklung der PowerShell

Die Windows PowerShell ist eine Symbiose aus

- dem DOS-Kommandozeilenfenster,
- den bekannten Skript- und Shell-Sprachen wie Perl, Ruby, ksh und bash,
- dem .NET Framework und
- der Windows Management Instrumentation (WMI).

Die PowerShell ist auf dem .NET Framework implementiert. Sie ist jedoch kein .NET Runtime Host mit der Möglichkeit, Befehle der Common Intermediate Language (CIL) auf der Common Language Runtime (CLR) auszuführen.

Die PowerShell verwendet ein völlig anderes Host-Konzept mit Commandlets, Objekt-Pipelines und einer neuen Sprache, die von Microsoft als PowerShell Language (PSL) bezeichnet wird. Sie ist Perl, Ruby, C# und einigen Unix-Shell-Sprachen sehr ähnlich, aber mit keiner Unix-Shell kompatibel. Nutzer der WMI Command Shell (*wmic.exe*), die mit Windows XP eingeführt wurde, werden sich in der PowerShell schnell zurechtfinden.



**Bild 2.1** Einflussfaktoren auf die Architektur und die Umsetzung der PowerShell



**ACHTUNG:** Die PowerShell ist angetreten, vom Administrator weniger Kenntnisse in Objektorientierung und über Softwarekomponenten zu verlangen, als dies der Vorgänger Windows Script Host (WSH) tat. Tatsächlich kann man in der PowerShell viel erreichen, ohne sich mit dem zu Grunde liegenden .NET Framework zu beschäftigen. Dennoch: Wer alle Möglichkeiten der PowerShell nutzen will, braucht dann aber doch etwas Verständnis für objektorientiertes Programmieren und Erfahrung mit dem .NET Framework.

## ■ 2.4 Betriebssysteme mit vorinstallierter PowerShell

Die folgende Tabelle zeigt, in welchen Betriebssystemen welche Version der PowerShell mitgeliefert wird bzw. wo sie nachträglich installierbar ist.

**Tabelle 2.1** Verfügbarkeit der PowerShell auf verschiedenen Betriebssystemen

Betriebssystem	Mitgelieferte PowerShell	Nachträglich installierbare PowerShell
Windows 2000, Windows 9x, Windows ME, Windows NT 4.0	PowerShell nicht enthalten	Nachträgliche Installation nicht von Microsoft unterstützt
Windows XP	PowerShell nicht enthalten	PowerShell 1.0 und PowerShell 2.0
Windows Server 2003	PowerShell nicht enthalten	PowerShell 1.0 und PowerShell 2.0
Windows Server 2003 R2	PowerShell nicht enthalten	PowerShell 1.0 und PowerShell 2.0
Windows Vista	PowerShell nicht enthalten	PowerShell 1.0 und PowerShell 2.0
Windows Vista	PowerShell 1.0 enthalten	PowerShell 2.0
Windows Server 2008	PowerShell 1.0 enthalten als optionales Features	PowerShell 2.0, PowerShell 3.0
Windows Server 2008 R2	PowerShell 1.0 enthalten	PowerShell 2.0, PowerShell 3.0
Windows 7	PowerShell 2.0 enthalten	PowerShell 3.0, PowerShell 4.0, PowerShell 5.0, PowerShell 5.1, PowerShell Core 6.x

(Fortsetzung nächste Seite)

**Tabelle 2.1** Verfügbarkeit der PowerShell auf verschiedenen Betriebssystemen (*Fortsetzung*)

Betriebssystem	Mitgelieferte PowerShell	Nachträglich installierbare PowerShell
Windows Server 2008 R2	PowerShell 2.0 enthalten	PowerShell 3.0, PowerShell 4.0, PowerShell 5.0, PowerShell 5.1, PowerShell Core 6.x
Windows Server 2008 Core	PowerShell nicht enthalten	PowerShell 3.0, PowerShell Core 6.x
Windows Server 2008 R2 Core	PowerShell 2.0 enthalten als optionales Feature	PowerShell Core 6.x
Windows 8.0	PowerShell 3.0 enthalten	Achtung: PowerShell 4.0 und 5.0/5.1 können nur durch ein (vorheriges) Update auf Windows 8.1 nachgerüstet werden.
Windows Server 2012 inkl. Core	PowerShell 3.0 enthalten	PowerShell 4.0, PowerShell 5.0, PowerShell 5.1, PowerShell Core 6.x
Windows 8.1	PowerShell 4.0 enthalten	PowerShell 5.0, PowerShell 5.1, PowerShell Core 6.x
Windows Server 2012 R2 inkl. Core	PowerShell 4.0 enthalten	PowerShell 5.0, PowerShell 5.1, PowerShell Core 6.x
Windows 10	PowerShell 5.0 enthalten	PowerShell Core 6.x
Windows 10 Creators Update (Redstone 2, Version 1703, April 2017)	PowerShell 5.1 enthalten	PowerShell Core 6.x
Windows Server 2016	PowerShell 5.1 enthalten	PowerShell Core 6.x
Windows Server 1709	PowerShell 5.1 enthalten	PowerShell Core 6.x
Windows Nano Server 2016	Reduzierte PowerShell Core 5.1 enthalten	PowerShell Core 6.x
Windows Nano Server 1709	- (PowerShell Core wurde aus dem Standardinstallationsumfang entfernt, vgl. <a href="https://docs.microsoft.com/de-de/windows-server/get-started/nano-in-semi-annual-channel">https://docs.microsoft.com/de-de/windows-server/get-started/nano-in-semi-annual-channel</a> )	PowerShell Core 5.1, PowerShell Core 6.x
Suse-Linux ab Version 42.1	-	PowerShell Core 6.x
Ubuntu-Linux ab Version 14.04	-	PowerShell Core 6.x
macOS/X ab Version 10.12	-	PowerShell Core 6.x

## ■ 2.5 Anbindung an Klassenbibliotheken

Die Version 1.0 der PowerShell enthielt sehr viele Commandlets für die Pipelining-Infrastruktur, aber nur sehr wenige Befehle, die tatsächlich Bausteine des Betriebssystems in die Pipeline werfen. Prozesse, Systemdienste, Dateien, Zertifikate und Registrierungsdatenbankeinträge sind die magere Ausbeute beim ersten Blick in die Commandlet-Liste. Drei Commandlets eröffnen der PowerShell aber neue Dimensionen: `New-Object` (für .NET- und COM-Objekte) und `Get-WmiObject` bzw. `Get-CimInstance` (für WMI-Objekte). Seit Version 2.0 gibt es – zumindest in Verbindung mit neueren Betriebssystemen – mehr PowerShell-Befehle, die tatsächlich auf das Betriebssystem zugreifen.



**HINWEIS:** Die Option, nicht nur alle WMI-Klassen, sondern auch alle .NET-Klassen direkt benutzen zu können, ist Segen und Fluch zugleich. Ein Segen, weil dem Skriptentwickler damit mehr Möglichkeiten als jemals zuvor zur Verfügung stehen. Ein Fluch, weil nur der Skriptentwickler die PowerShell-Entwicklung richtig beherrschen kann, der auch das .NET Framework kennt. Um die Ausmaße von .NET zu beschreiben, sei die Menge der Klassen genannt. In .NET 2.0 waren es 6358, in .NET 3.5 sind es 10758, in .NET 4.7 sind es 13526.

## ■ 2.6 PowerShell versus WSH

Administratoren fragen sich oft, wie sich die PowerShell im Vergleich zum Windows Script Host (WSH) positioniert, womit man neue Skripting-Projekte beginnen sollte und ob der WSH bald aus Windows verschwinden wird. Die folgende Tabelle trägt Fakten zusammen und bewertet auch die beiden Skripting-Plattformen.

**Tabelle 2.2** Vergleich WSH und Windows PowerShell bzw. PowerShell Core

	Windows Script Host (WSH)	Windows PowerShell (WPS)	PowerShell Core (PS Core)
Erstmals erschienen	1998	2006	2018
Aktueller Versionsstand	5.8	5.1 und Core 5.1	6.0
Betriebssystem(e)	Alle Windows-Betriebssysteme ab Windows 95/NT 4.0	Version 1.0 ab Windows XP, Version 5.1 ab Windows 7 und Windows Server 2008 R2; Windows PowerShell Core 5.1 auf Windows Nano Server 2016	Windows ab Version 7, Windows Server ab Version 2008 R2, diverse Linux-Distributionen, macOS

(Fortsetzung nächste Seite)

**Tabelle 2.2** Vergleich WSH und Windows PowerShell bzw. PowerShell Core (Fortsetzung)

	Windows Script Host (WSH)	Windows PowerShell (WPS)	PowerShell Core (PS Core)
Basis-Programmierframework	Component Object Model (COM)	.NET Framework bzw. .NET Core für PowerShell Core unter Windows Nano Server 2016	.NET Core
Derzeitiger Funktionsumfang	Sehr umfangreich	Funktionsumfang in Form von Commandlets abhängig vom Betriebssystem: <ul style="list-style-type: none"> <li>▪ nur wenige Commandlets vor Windows 7,</li> <li>▪ bessere Unterstützung ab Windows 7,</li> <li>▪ sehr umfangreich erst ab Windows 8 bzw. Windows Server 2012.</li> </ul> Wichtig: Auch ohne Commandlets steht auf den älteren Betriebssystemen aber ein hoher Funktionsumfang zur Verfügung, wenn man COM- oder .NET-Komponenten nutzt, was aber mehr Wissen voraussetzt.	Teilmenge von Windows PowerShell 5.1 und wenige zusätzliche neue Funktionen
Weiterentwicklung der Laufzeitumgebung	Nein, nur noch Beheben von Fehlern und Sicherheitslücken	Nein, nur noch Beheben von Fehlern und Sicherheitslücken	Ja
Weiterentwicklung der Bibliotheken	Gering, gelegentlich erscheinen noch neue COM-Bibliotheken	Ja, zahlreiche Commandlet-Erweiterungen erscheinen immer wieder mit Microsoft-Produkten.	Ja, Microsoft wird hier in den kommenden Jahren viel investieren
Weiterentwicklung der Werkzeuge	Nein	Ja	Ja
Basissyntax	Mächtig	Sehr mächtig	Sehr mächtig
Direkte Skripting-Möglichkeiten	Alle COM-Komponenten mit IDispatch-Schnittstelle einschließlich WMI	Alle .NET-Komponenten, alle COM-Komponenten, alle WMI-Klassen	Alle .NET-Standard-Komponenten. COM und WMI nur unter Windows
Skripting-Möglichkeiten über Wrapper	Alle Betriebssystemfunktionen	Alle Betriebssystemfunktionen	Viele Betriebssystemfunktionen
Werkzeuge von Microsoft	Scriptgeneratoren, Debugger, aber kein Editor	Integrated Scripting Environment (ISE), PowerShell Tools für Visual Studio, PowerShell-Erweiterung für VSCode	PowerShell-Erweiterung für VSCode, unter Windows auch ISE und PowerShell Tools für Visual Studio

	Windows Script Host (WSH)	Windows PowerShell (WPS)	PowerShell Core (PS Core)
Werkzeuge von Drittanbietern	Editoren, Debugger, Scriptgeneratoren	Editoren, Debugger, Scriptgeneratoren	Bisher nur für Windows, siehe „Windows PowerShell“
Einarbeitungsaufwand	Hoch	Mittel bis hoch (je nach Art der PowerShell-Nutzung)	Mittel bis hoch (je nach Art der PowerShell-Nutzung)
Informationsverfügbarkeit	Hoch	Mittlerweile auch sehr hoch	Für die Nutzung unter Windows sehr hoch, für die anderen Betriebssysteme noch sehr gering
Startanwendung	cscript.exe und wscript.exe	powershell.exe	pwsh.exe (Windows) bzw. pwsh (Linux und macOS)



**HINWEIS:** Hinweise zur Umstellung von WSH/VBScript auf die PowerShell finden Sie unter [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-powershell-1.0/ee221101\(v=msdn.10\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-powershell-1.0/ee221101(v=msdn.10)).

# Index

## Symbole

- & 58
- > 226
- >> 226
- 32-Bit 3
- \$\_ 81, 91, 145
- \$PSVersionTable 10
- and 107
- .dll 39, 132, 294
- .exe 132
- Force 43
- .NET 1, 32, 80, 96, 145, 166, 232, 284, 410
  - Bibliothek 281
  - Klasse 281
  - Runtime Host 28
- .NET Core 1, 32, 281, 410, 414, 420
- .NET Framework 23, 28, 251
  - 4.0 3
- .NET Standard 281
- or 107
- .pkg 417
- .ps1 12, 121
- .ps1;ps1 54
- .psd1 349
- [Type] 290
- Verbose 44

## A

- Ablaufverfolgung 1, 336, 337
- About 135
- Accelerator 140
- Accent Grave
  - Gravis 45

- AccessMask 370
- Active Directory 1, 240, 384, 454
- Active-Scripting 128
- Add() 293
- Add-ADGroupMember 385
- Add-Content 366
- Add-LocalGroupMember 420
- Add-Member 105
- Add-PSSnapin 421
- Add-Type 231, 296, 367
- Add-VMHardDiskDrive 406
- Administrator 276
- Administratorrechte 252, 272, 276, 323, 353
- ADO.NET 384
- Alias 35, 48, 237
- Aliaseigenschaft 99, 104
- AliasInfo 49
- AllowClobber 18, 352
- AllSigned 128
- Ankerelement 161
- AppDomain 296
- Apple Software Package 417
- Args 126, 145, 262
- Array 167, 170, 171, 172
- ArrayList 170
- as 142
- ASP.NET Core 412
- Assembly 294, 347, 355
- Audio 288
- Aufnahmedatum 366
- Aufzählung 299
- Ausdruck 56
  - Regulär 159
- Ausdrucksauflösung 152
- Ausdrucksmodus 56
- Ausführungsrichtlinie 128

Ausgabe  
 – mehrspaltig 209  
 – unterdrücken 224  
 Authentifizierung 233  
 Azure 412

**B**

BackgroundColor 146  
 Backspace 154  
 bash 411, 436, 438  
 Basisauthentifizierung 249  
 Bedingung 182  
 Beep 154  
 Beep() 291  
 Befehl  
 – Extern 35, 57  
 Befehls-Add-On 68  
 Befehlseingabefenster 20  
 Befehlsmodus 56  
 Benutzer 454  
 – Massenanlegen 384  
 Benutzereingabe 229  
 Benutzerkontensteuerung 14  
 Bit 3  
 Benutzerkonto 384, 420  
 Benutzername 232  
 Berechnung 114  
 Betriebssystem installieren 406  
 BitLocker 454  
 Bitmap 366, 367  
 BITS 37  
 break 176, 178, 188, 192  
 Bypass 129  
 ByPropertyName 83, 84, 85  
 Byte 148  
 ByVal 83, 85

**C**

C# 23, 135, 454  
 CAB 71  
 Carriage Return 154  
 cat 438, 440  
 cd 438  
 CD 294  
 ChangeAccess 377  
 ChildSession 267  
 Chkdsk() 325

CIL 28  
 CIM 25  
 CimClass 308, 317  
 CimClassProperties 317  
 CimInstance 308, 317, 318  
 CimInstanceProperties 317  
 CimProperty 317  
 class 200  
 ClearCase 133  
 Clear-EventLog 250  
 Clear-Host 190  
 Clear-Variable 144  
 -cmatch 159  
 -cnotmatch 159  
 Clipboard siehe Zwischenablage  
 CLR 10, 28  
 cmd.exe 80  
 Codeeigenschaft 99, 104  
 COM 32, 301  
 – Klasse 303  
 Command Mode 56  
 Commandlet 1, 35, 47, 57, 61, 81, 243  
 – Provider 239  
 CommandNotFoundException 197  
 CommitChanges() 288  
 Common Intermediate Language  
 siehe CIL  
 Common Language Runtime siehe CLR  
 Common Parameter 41  
 compare 118  
 Compare-Object 117, 118  
 Complete-Transaction 421  
 Component Object Model 23  
 Component Object Model siehe COM  
 Computername 125, 430  
 Computerrichtlinie 338  
 Confirm 42 ff.  
 ConfirmPreference 44  
 Continue 43, 176, 178, 188, 192, 194  
 ConvertFrom-CSV 440  
 ConvertTo-SecureString 234  
 COP 79  
 Copy-Item 192, 266  
 Count 87, 169, 188  
 CPU 119, 459  
 CQL 314  
 Create() 326  
 CreateObject() 304  
 CreationTime 366  
 Creators Update 17, 62  
 CSV 54, 229, 383, 399, 402

CSV-Datei 119, 459

CVS 133

## D

DataRow 104, 105

Datei 203

– löschen 37

Dateiname 35

Dateinamenerweiterung 119, 121, 460

Dateisystem 1, 365, 454

Dateisystemfreigabe 370

Dateisystemstruktur 368

Datenbank 454

Datenbankzeile 104

Datenbereich 454

Datendatei 454

Datenmenge 237

Datentyp 138, 145, 171

– .NET 81

– PowerShell 138

DateTime 91, 93, 95, 290

Datum 165

Day 91

DBG 340

DCOM 249, 304, 308

Debug 42, 44

Debugger 22

Debugging 1, 21, 340

Debug-Modus 340

DebugPreference 44

Decimal 148

Defender 454

Deserialisierung 258

Desired State Configuration 454

Desktop Management Task Force siehe DMTF

Dezimalzahl 148

DHCP 381

Diagnose 336

Dialogfenster 232

diff 118

Digest 249

dir 438, 443

DirectoryEntry 104, 105, 141, 284, 285

DirectoryInfo 96, 108

DirectorySearcher 141

Disable-PSRemoting 255

Disable-PSSessionConfiguration 265, 268

Distributed Component Object Model siehe  
DCOM

Distributed COM siehe DCOM

DMTF 249

DNS 277

DnsClient 382

DNSClient 382

DNS-Server 382

do 176

Docker 412, 455

Dokumentation

– .NET 75

Dollarzeichen 114, 152

Domäne

– Beitritt 277

DOS 1

Dot Sourcing 54, 127, 128, 346

DotNetTypes.Format.psixml 203, 212, 214

Double 148

DownloadString() 287

DriveInfo 290

DriveType 298

Drucker 203, 225

DSC 454

DVD 294, 406

## E

echo 57

Echo 56

Eigenschaft 99

Eigenschaftssatz 99, 101

Eingabe 229

Eingabedialog 231

Einzelschrittmodus 335

Elevated 130

Else 182

Emacs 122

Enable-PSRemoting 253

Enable-PSSessionConfiguration 265, 268

Enable-PSSessionConfiguration 268

Enter-PSSession 255, 256, 265, 270, 341,  
446

Enum 299

Enumerationsklasse 298

env 438

Ereignis 454

Ereignisprotokoll 119, 338, 421, 454, 459

Error 146, 197

ErrorAction 42, 43, 44, 194, 195, 196, 366

ErrorActionPreference 44, 146, 196

ErrorBackgroundColor 16

ErrorRecord 192, 195, 198  
 ErrorVariable 43, 195  
 ETS 316  
 Exception 192, 197, 198  
 Exchange Server 75, 454  
 ExecutionPolicy 13, 15  
 EXIF 366  
 exit 176  
 Exit-PSSession 257, 265  
 Export-Alias 54  
 Export-CliXml 229  
 Export-CSV 108, 229, 430  
 Export-ModuleMember 348  
 Expression 206  
 Expression Mode 56  
 Extended Reflection 81  
 Extended Type System siehe ETS

## F

false 38, 43, 140, 145  
 Fehler 43  
 Fehlerbehandlung 191  
 Fehlerklasse 176, 198  
 Fehlermeldung 38  
 Fehlersuche 333  
 Fehlertext 176  
 Fernaufruf 250  
 Fernausführung 125, 249  
 Fernverwaltung 249  
 Fernzugriff 249  
 Festplatte  
 – virtuell 406  
 Field 100  
 FileInfo 96, 108  
 FileSystemRights 299  
 Find-Module 354, 356, 428  
 Firewall 440  
 First 89  
 fish 411  
 For 177, 180  
 Force 41, 43  
 ForEach 92, 113, 176, 180, 459, 460, 462  
 ForEach-Object 91, 92, 101, 113, 114, 118, 169, 181,  
 187, 336, 403  
 ForegroundColor 66  
 Form Feed 154  
 Format 206  
 Formatkennzeichner 219  
 Format-List 79, 204

Format-Table 90, 101, 120, 204, 206, 215, 216,  
 460, 461  
 Format-Volume 406  
 Format-Wide 203, 204, 209, 210  
 Fortschrittsanzeige 227  
 Forward Engineering 79  
 Foto  
 – sortieren 366  
 Framework Class Library 281  
 Freigabe 369, 379  
 FTP 430  
 FullAccess 377  
 function 35, 176, 184, 190, 237  
 Funktion 35, 184, 237  
 – eingebaut 190

## G

GAC 294, 296  
 Ganzzahl 148  
 Gateway 382  
 Get-ADComputer 384  
 Get-ADGroup 384  
 Get-ADGroupMember 385  
 Get-ADObject 39, 384  
 Get-ADOrganizationalUnit 384  
 Get-ADUser 384  
 Get-Alias 49  
 GetAssemblies() 296  
 Get-ChildItem 36, 37, 38, 40, 79, 80, 108, 112,  
 113, 237, 438, 460, 462  
 Get-CimAssociatedInstance 307  
 Get-CimClass 307, 315  
 Get-CimInstance 307, 309, 310, 311, 326, 420  
 Get-Clipboard 234, 235, 358, 421  
 Get-Command 47, 61, 62, 63, 361  
 Get-Content 366, 383, 403  
 Get-Counter 250, 420  
 Get-Credential 67, 232, 233, 273  
 Get-Culture 157  
 Get-Date 93, 165, 166  
 Get-DHCPServer 18  
 Get-DirectoryEntry 152  
 Get-DomainController 18, 19  
 GetDrives() 289  
 Get-EventLog 9, 250, 421, 459  
 Get-ExecutionPolicy 13  
 Get-Filecatalog 75  
 Get-Help 63, 64, 65, 69, 70, 73, 135, 430,  
 432

- Get-History 432
  - Get-Host 432
  - Get-HotFix 250
  - Get-LocalUser 420
  - Get-Location 48
  - GetLongDateString() 93
  - GetLongTimeString() 93
  - Get-LoremIpsum 368
  - Get-Member 94, 97, 99, 100, 109, 118, 291, 297, 318
  - Get-Methode 100
  - Get-Module 207, 347, 359
  - Get-MountPoint 18
  - GetNames() 299
  - Get-NetIPInterface 382
  - GetObject() 304
  - Get-PipelineInfo 94, 96, 108
  - Get-Process 20, 36, 37, 39, 40, 48, 54, 79, 84, 92, 93, 97, 99, 108, 109, 114, 215, 216, 225, 250, 336, 430, 438, 457, 459, 460
  - Get-PSBreakpoint 342
  - Get-PSProvider 240
  - Get-PSRepository 356
  - Get-PSSession 265
  - Get-PSSessionConfiguration 265, 267
  - Get-Random 149
  - Get-Service 66, 67, 84, 87, 94, 108, 126, 225, 250, 430, 431, 448
  - GetShortDateString() 93
  - GetShortTimeString() 93
  - Get-SmbShare 371, 377
  - Get-SmbShareAccess 378
  - GetTempName() 302
  - Getter 100
  - Get-TerminalSession 18
  - GetTimestamp() 450
  - Get-TraceSource 336
  - Get-Transaction 421
  - GetType() 95, 145, 223
  - Get-Unique 111
  - Get-Uptime 431, 450
  - Get-Variable 138, 141, 146
  - Get-WinEvent 250
  - Get-WmiObject 31, 250, 294, 307, 309, 311, 313, 420
  - Gigabyte 148
  - Git 133
  - Github 450
  - Gleichheitszeichen 173
  - global 143
  - Global Assembly Cache siehe GAC
  - gm 118
  - Grafikkarte 310
  - Grant-SmbShareAccess 378
  - Gravis 45, 46, 88, 136, 154
  - grep 412
  - GridView 210
  - Group 112
  - Group-Object 112, 118, 459
  - Gruppe 385, 399, 400, 454
  - Gruppenmitglieder 385
  - Gruppenrichtlinie 338, 454
  - Gruppierung 112
  - GUI 455
  - Gültigkeitsbereich 139, 143
- ## H
- Haltepunkt 21, 339, 341
  - Hardware 454
  - Hash-Tabelle 171, 172, 286, 326
  - Hashtable 171, 172, 286
  - Heimatordner 145
  - Help-Info 71
  - Here-String 150
  - Hexadezimalzahl 148
  - hidden 201
  - Hilfe 61
  - Hilfetext 70
  - Hintergrundübertragungsdienst 37
  - Hit Refresh 412
  - HKCU 237
  - HKLM 237
  - Home 145, 438
  - Host 146
  - Hosting 454
  - HTML 403
  - HTTP 249, 430
  - HTTPS 249, 430
  - Hyper-V 276, 406
- ## I
- i 174
  - I 366
  - if 176, 182
  - imatch 159
  - IIS 239, 402
    - 8.0 403
  - ILSpy 297

Impersonifizierung 233  
 Implizites Remoting 269  
 Import-Alias 54  
 Import-CliXml 229  
 Import-CSV 229, 400, 403, 459  
 Import-Module 348, 355, 361  
 Import-PSSession 270  
 Index 167  
 -inotmatch 159  
 Input 145  
 InputBox 231, 296  
 InputBox() 231  
 InputObject 84, 297  
 inquire 43, 194  
 Installationsordner 3, 104, 145  
 Install-Module 18, 352, 354, 356, 428  
 Instanz 286  
 int 139  
 Int32 139  
 Int64 148  
 Integrated Scripting Environment 20  
 IntelliSense 20, 36  
 InternalHostUserInterface 230  
 International .NET Association XXII  
 Internet Information Services 455  
 Internet Information Services siehe IIS  
 Invoke-CimMethod 307, 326, 420  
 Invoke-Command 255, 257, 259, 262, 263, 264,  
 266, 273, 276, 341  
 Invoke-Expression 176  
 Invoke-WebRequest 430  
 Invoke-WmiMethod 307, 325, 420  
 IPAddress 141, 382  
 IP-Adresse 117, 141, 277, 381  
 ipconfig 57, 87  
 IsCoreCLR 430  
 ISE 122, 227, 433  
 IsLinux 430  
 IsmacOS 430  
 ISO 406  
 IsWindows 430

## J

JEA 454  
 Jeffrey Snover 123  
 Job 454  
 -Join 158  
 Join() 158  
 Join-String 158

JPEG 366  
 Just Enough Administration 454

## K

Kennwort 232, 234  
 Kerberos 249  
 Kill() 92  
 Kilobyte 148  
 Klammer 39  
 - rund 161  
 Klammeraffe 56  
 Klasse 138, 289  
 - COM 282, 301  
 - .NET 200, 281  
 - PowerShell 200  
 - statisch 291  
 Klassenmitglied 289  
 Klassenname 284  
 Known Host 448  
 Kommandozeilenbefehl 57  
 Kommentar 136  
 Konstruktorkfunktion 283, 285, 301  
 Kosinus 290  
 Kreuzzuweisung 175  
 ksh 411

## L

Label 206  
 LastExitCode 145  
 Laufwerk 237, 239, 244  
 Leistungsdaten 454  
 Length 87  
 Limit-EventLog 250  
 Linux 1, 2, 7, 30, 240, 301, 409, 411, 415, 418,  
 423, 438, 446  
 - Dateisystem 438, 443  
 Linux Foundation 412  
 Literal 219  
 LoadFrom() 295  
 LoadWithPartialName() 295  
 Logarithmus 290  
 Loopback 252  
 ls 438

**M**

MachineName 251  
macOS 1, 2, 7, 30, 240, 301, 409, 411, 418, 423, 438  
  - Dateisystem 443  
MAML 70  
man 438  
ManagementClass 104, 105, 141, 308, 311, 316  
ManagementObject 105, 141, 308, 311, 316, 318, 319, 325  
ManagementObjectCollection 316  
ManagementObjectSearcher 141, 311  
-match 159  
Manifest 349  
Match 105  
MaximumDriveCount 245  
MaximumErrorCount 146  
md 366, 367, 383  
measure 114  
Measure-Command 336  
Measure-Object 114, 118, 459  
Megabyte 148  
Mercurial 133  
Message 192  
MessageBox 232  
Methode 99  
Microsoft Access 3  
Microsoft Certified Solution Developer XXII  
Microsoft Command Shell 23  
Microsoft Developer Network 75, 281  
Microsoft Developer Network siehe MSDN  
Microsoft Excel 400  
Microsoft Shell 23  
Microsoft SQL-Server 454  
Microsoft Word 304  
Microsoft.Management.Infrastructure.  
  CimClass 317  
Microsoft.Management.Infrastructure.  
  CimClassProperties 317  
Microsoft.Management.Infrastructure.  
  CimInstance 317  
Microsoft.Management.Infrastructure.  
  CimInstanceProperties 317  
Microsoft.Management.Infrastructure.  
  CimProperty 317  
Microsoft.VisualBasic 231  
Microsoft.VisualBasic.Interaction 231, 304  
Minute 91  
Mitglied

  - WMI 322  
Modul 347, 357, 454  
Module Browser 353  
Modulo 172  
Monad 23  
Month 91  
more 190, 216  
Most Valuable Professional XXII  
Move-ADObject 385  
MSCL 28  
MSDN 173, 281  
MSDN Library 75, 281  
MSFT\_SmbShare 377  
MSFT\_SmbShareAccessControlEntry 378  
MSH siehe Microsoft Shell  
MSI 404

**N**

Nachkommastelle 148, 290  
Name 84  
Nano Server 30, 409, 455  
NanoWbem 307  
Navigation 237  
Navigation Provider 239  
Navigationsbefehl 241  
Navigationsparadigma 237  
NetAdapter 382  
netstat 57  
NetTCPIP 382  
Netzwerk 454  
Netzwerkcenter 254  
Netzwerkkarte 381  
Netzwerkkonfiguration 381  
Neustart 277  
new() 283, 285, 301  
New-ADGroup 384  
New-ADOrganizationalUnit 384  
New-ADUser 384  
New-Button 66  
New-CimInstance 307, 326  
New-CimSession 310  
New-CimSessionOption 310  
New-EventLog 250  
New-Guid 283  
New-Item 238, 383  
New-Itemproperty 383  
New-Line 154  
New-LocalUser 12  
New-Module 347

New-NetIPAddress 382  
 New-Object 282, 283, 284, 291, 294, 301  
 New-Partition 406  
 New-PSDrive 244, 245  
 New-PSSession 253, 255, 265, 266, 268, 269,  
   273, 276  
 New-SmbShare 43, 376, 377  
 New-TimeSpan 166  
 New-VM 406  
 NoAccess 377  
 NoElement 112  
 Non-Terminating Error 191  
 -notmatch 159  
 NoProfile 346  
 Notepad 122  
 Notizeigenschaft 99, 103, 109  
 Now 290  
 NTLM 249  
 null 115, 126, 181, 185, 225

## O

o 367  
 Object[] 188  
 Objektadapter 105, 316  
 Objektorientierung 80  
 ODBC 454  
 OFS 145  
 ogv 203  
 OMI 307, 309  
 OMI siehe Open Management Infrastructure  
 OOP siehe COP  
 Open Management Infrastructure 444  
   - OMI 309  
 Open Source 24  
 OpenSSH 444  
 Operator 107, 158, 167, 172  
 Ordner 37, 59, 241, 365  
   - löschen 365  
 Organisationseinheit 399, 400  
 OSS  
   - Open Source 24  
 OutBuffer 43  
 OutVariable 42, 116  
 Out-Default 203, 212  
 Out-File 203, 226  
 Out-GridView 67, 203, 204, 210, 212  
 Out-Host 203, 216  
 Outlook 111  
 Out-Null 203, 204, 224

Out-Printer 203, 225  
 Out-Speech 203, 227, 354  
 ov siehe OutVariable

## P

PackageManagement 350  
 Parameter 36, 37, 65, 85, 188  
   - Abkürzung 40, 41  
   - Skript 125  
 Parameterliste 126  
 ParentSession 267  
 PassThru 116  
 passwd 440  
 PercentComplete 227  
 Performce 133  
 Performance Counter 454  
 PERL 135  
 Petabyte 148  
 Pfadangabe 241  
 Pflichtparameter 9  
 PHP 135  
 pick-head 462  
 Ping 57  
 Ping-Host 18, 19  
 Pipe 80  
 Pipeline 1, 28, 79, 81, 95, 114, 224, 321, 412  
 Pipeline Processor 82  
 PipelineVariable 43, 116 f. 215  
 Pipelining 79, 172, 237  
 Platzhalter 39, 219  
 Port 440  
 PowerShell 122  
 PowerShell 1, 48, 79  
   - Hosting 1  
   - Laufwerk 237, 244  
   - Skriptsprache 135  
   - Version 1.0 23  
   - Version 2.0 23  
   - Version 3.0 24  
   - Version 4.0 24  
   - Version 5.0 24  
   - Version 5.1 24  
 PowerShell Community Extensions 18  
 PowerShell Core 1, 2, 24, 31, 409, 410  
   - Funktionsumfang 418  
   - installieren 413  
   - Konsole 432  
   - Module 354  
   - Version 5.1 30, 409

- Version 6.0 2
- Version 6.x 409
- WMI 307
- PowerShell Direct 275, 277
- PowerShell Remoting
  - Port 275
- PowerShell Remoting Protocol 249, 251, 444
- PowerShell Remoting siehe Remoting
- powershell.exe 47, 362
- powerShellExePath 434
- PowerShell Gallery 18, 350
- PowerShellGet 350
- PowerShellPlus 122
- Printer 203
- Process 84, 109, 117
- Professional Developer Conference 23
- profile 343
- profile.ps1 295
- Profilskript 343, 346
- Programmiersprache 182
- PromptForChoice() 230
- Property 100, 215
- PropertyDataCollection 316, 317, 318
- Protokollierung 338
- Provider 240
  - PowerShell 239
- Proxy-Commandlet 269
- Prozedur 185
- Prozess 36, 37, 119, 459
  - auflisten 225
- Prozessverwaltung 454
- ps 438
- PSComputername 264
- PSCredential 232, 233
- PSCustomObject 109
- PSCX 19, 203, 234, 352, 354, 355
- PSCX siehe PowerShell Community Extensions
- PSHome 145
- PSHost 145
- PSItem 80
- PSModuleAutoLoadingPreference 146, 360
- PSModulePath 348, 429, 438
- PSObject 105
- PSReadline 432, 433
- PSRP siehe PowerShell Remoting Protocol
- PSScriptRoot 127
- PSSession 265
- PSVariable 138
- Public Network 254
- Pull Request 450
- Punktnotation 90, 286, 322

- Put() 322, 323
- pv siehe PipelineVariable
- pwsh 414, 432
- Python 135

## Q

- Quantifizierer 161
- Quantor 161
- QueryDialect 314

## R

- RDP 277
- ReadAccess 377
- Read-Host 229, 234
- Rechenleistung 119, 459
- recurse 38
- Redirection siehe Umleitung
- Redstone 17, 24, 62
- Referenzkopie 173, 174
- Register-CimIndicationEvent 307
- Register-Packagesource 355
- Register-PSSessionConfiguration 265, 268
- Registrierungsdatenbank 1, 237, 243, 245
- Registry 383, 454
- Regulärer Ausdruck 140, 159
- Remote Procedure Call siehe RPC
- Remote Server Administration Tools 384
- Remoting 125, 249, 277, 446
  - Implizit 269
- Remove-ADGroupMember 385
- Remove-ADObject 384
- Remove-ADUser 43
- Remove-Alias 54, 431
- Remove-CimInstance 307, 328
- Remove-EventLog 250
- Remove-Item 37, 41, 43, 54, 383, 406
- Remove-Module 348, 363
- Remove-NetIPAddress 382
- Remove-NetRoute 382
- Remove-PSBreakpoint 342
- Remove-PSSession 265, 267
- Remove-PSSnapin 421
- Remove-Service 431
- Remove-SmbShare 41, 43, 377
- Remove-Variable 144, 145
- Remove-VM 406
- Remove-WmiObject 307, 327

Rename-ADObject 385  
 Replace 157  
 requires 130  
 Resolve-Assembly 296  
 Resolve-Path 242  
 ResponseHeaders 287  
 Restart-Computer 250  
 Restart-Service 44, 272, 275  
 Restricted 128  
 return 176, 243  
 Revoke-SmbShareAccess 378  
 RPC 250  
 RuntimeException 197

## S

s 420  
 SAPI.SPVoice 228, 303  
 Satya Nadella 412  
 Save-Help 72  
 Save-Module 352  
 Schalter 38  
 Schleife 180  
 Schlüssel 237  
 Schnittstelle 200  
 Scope siehe Gültigkeitsbereich  
 script 143  
 Scripting.FileSystemObject 302  
 SDDL 371  
 Security Descriptor Definition Language 267  
 Select  
 - PowerShell 109  
 Select-Object 9, 79, 80, 89, 101, 105, 109, 113,  
 118, 216, 320, 459  
 Select-String 57, 86  
 Semantic Versioning 141  
 Semikolon 119, 459  
 Serialisierung 96, 258  
 ServiceController 258  
 Session 304  
 Set-ADAccountPassword 385  
 Set-Alias 53  
 Set-CimInstance 307, 324  
 Set-Clipboard 234, 235, 421  
 Set-Date 166  
 Set-DnsClientServerAddress 382  
 Set-ExecutionPolicy 13, 15, 121, 128  
 Set-Item 272  
 Set-Location 48, 237, 438  
 Set-Methode 100  
 Set-NetIPInterface 382  
 Set-PSBreakpoint 340, 342  
 Set-PSDebug 141, 333, 335  
 Set-PSReadlineOption 16, 433  
 Set-PSSessionConfiguration 265  
 Set-Service 431  
 Set-StrictMode 141  
 Setter 100  
 Set-TraceSource 337  
 Set-Variable 138, 146  
 Set-VMDVdDrive 406  
 Set-VolumeLabel 18  
 Set-WmiInstance 307, 324  
 Set-WSManQuickConfig 254  
 Shell 1, 79  
 Show-Command 67, 68  
 Show-EventLog 250  
 Show-Service 250  
 ShowWindow 69  
 Sicherheit 454  
 - PowerShell 128  
 Sicherheitsmodell 1  
 Sicherheitsrichtlinie 129  
 Signatur  
 - digital 453  
 SilentlyContinue 43, 194, 366  
 Simple Object Access Protocol siehe SOAP  
 Sitzung 264, 265, 266, 267  
 Sitzungskonfiguration 267  
 Skip 89  
 SkipNetworkProfileCheck 255  
 Skript 121, 123  
 - PowerShell 121  
 Skriptausführungsrechte 13  
 Skriptausführungsrichtlinie 14  
 Skriptblock 143, 257  
 Skriptdatei 121  
 Skripteigenschaft 99, 103  
 Skriptfenster 20  
 Snap-In 347, 362, 421  
 SOAP 249  
 Software 454  
 Softwareentwickler 281  
 Softwareinstallation 404  
 Softwarekomponente 294  
 Sortieren 110  
 Sort-Object 79, 80, 81, 105, 110, 112, 113, 118, 187,  
 460, 462  
 -Split 158, 159  
 Speech 203  
 SpeechSynthesizer 228

- Speicher 87
  - Sprachausgabe 203, 227, 303
  - SqlConnection 286
  - SQLPS 239
  - SQL Server 412
  - SSH 445, 446
  - sshs 445
  - StackTrace 145
  - Standarddrucker 225
  - Start-Process 118
  - Start-Service 259
  - Start-Sleep 131
  - Start-VM 406
  - static 200
  - Status 227
  - Stop 43, 194
  - Stop-Computer 250
  - Stop-Process 92, 118
  - Stop-Service 37
  - Stop-VM 406
  - Stopwatch 450
  - Streaming 82
  - String 150, 157
  - Subnetz-Maske 382
  - Subversion 133
  - Suse 30
  - Suspend 42
  - Switch 38, 176, 182
  - System32 113, 119, 460
  - System.ApplicationException 197
  - System.Boolean 241
  - System.Collections.Hashtable 171
  - System.Console 291
  - System.Data 283
  - System.Data.SqlClient.SqlConnection 286
  - System.DateTime 165, 285, 287, 290
  - System.Diagnostics.Process 81, 92, 96, 212
  - Systemdienst 83
  - System.DirectoryServices 283
  - System.Directoryservices.DirectoryEntry 285, 288
  - System.Enum 299
  - System.Environment 256
  - System.Int32 139, 148
  - System.IO.DriveInfo 289, 292, 294, 298
  - System.IO.DriveType 298
  - System.Management 283
  - System.Management.Automation 75
  - System.Management.Automation.PathInfo 242, 243
  - System.Management.ManagementObject 166
  - System.Math 290
  - System.Media.SoundPlayer 288
  - System.Net.WebClient 287
  - System.Object 94, 96
  - System.Random 150, 285
  - System.Reflection 294
  - System.ServiceProcess.ServiceController 96
  - System.String 150, 155, 440
  - System.TimeSpan 166
  - System.Type 95, 145, 223
  - System.ValueType 173
  - System.Windows.Forms 294, 366
- ## T
- t 367, 420
  - Tabellenformatierung 206
  - Tabulator 154
  - TCP/IP 382
  - tcsh 411
  - Team Foundation Server siehe TFS
  - Tee-Object 115, 116
  - Telnet 256
  - Terminating Error 191
  - Terrabyte 148
  - Test-Connection 19, 85, 86
  - Test-FileCatalog 73
  - Test-ModuleManifest 349
  - Test-Path 241
  - Textdatei 119, 459
  - Texteingabefeld 231
  - TextInfo 157
  - Textpad 122
  - TFS 133
  - ThrottleLimit 264
  - throw 176, 177, 197
  - TIFF 366
  - TimeSpan 336
  - ToLower() 157
  - Ton 291
  - ToString() 94, 96, 293
  - TotalProcessorTime 223
  - ToTitleCase() 157
  - ToUpper() 157
  - Tracing 336
  - Transaktion 421, 453
  - Trap 176, 191, 192, 197
  - true 140, 145
  - Trusted Host 272
  - Try-Catch-Finally 191, 197

Typ 139, 281  
 Typadapter 139, 140, 168  
 Typbezeichner 139  
 Type Cast siehe Typkonvertierung  
 types.ps1xml 54, 55, 56, 104, 105  
 Typisierung 138  
 Typkennzeichner siehe Typbezeichner  
 Typkonvertierung 110, 142  
 Typname siehe Typbezeichner

## U

UAC 14, 125, 130  
 Überladung 190  
 Ubuntu 30, 415  
 ufw 440  
 Umgebungsvariable  
 – Linux 438  
 Umleitung 226  
 Undefined 129  
 Undo-Transaction 421  
 Universal Coordinated Time 324  
 Unix 1, 23, 79, 80, 135  
 Unregister-PSSessionConfiguration 265, 269  
 Unrestricted 128  
 Unterordner 96  
 Unterroutine 184  
 Unterschlüssel 238  
 until 176  
 Update-Help 71, 72  
 User Account Control siehe Benutzerkonten-  
 steuerung  
 User Settings 434

## V

ValidateLength 147  
 ValidatePattern 147  
 ValidateRange 147  
 ValidateScript 147  
 ValidateSet 147  
 ValueType 173  
 Variable 22, 95, 116, 137, 145, 219, 237  
 – Auflösung 151  
 – eingebaut 145, 430  
 – vordefiniert 145, 430  
 Variablenauflösung 151, 152, 219  
 Variablenkennzeichner 116, 137  
 Variablentypisierung 138

Verbindungszeichenfolge 286  
 Verbose 42, 44  
 VerbosePreference 44, 146  
 Vergleich 117  
 Vergleichsoperator 105  
 Verzeichnisdienst 104  
 Verzweigung 115  
 VHD 406  
 View 214  
 Virtualisierung 406  
 Virtuelle Machine siehe VM  
 Virus 129  
 Visual Basic .NET 23  
 Visual Studio Code 433  
 Visual Studio Team Services siehe VSTS  
 VM 406  
 VMBus 276  
 VMGUID 276  
 VMName 276  
 void 225  
 VolumeLabel 288  
 VSCode-PowerShell 433  
 VSTS 133

## W

WarningAction 42, 43, 194  
 WarningVariable 43  
 Warnung 43  
 WBEM 25  
 WebAdministration 403  
 Webserver 239  
 Website 403  
 Wertemenge 167  
 Wertkopie 173, 174  
 WhatIf 42 ff., 146  
 WhatIfPreference 44  
 Where-Object 48, 49, 79, 80, 81, 93, 105, 107,  
 114, 118, 225, 460  
 while 176  
 Width 206  
 Win32 OpenSSH 444  
 Win32\_ACE 373  
 Win32\_CDROMDrive 320  
 Win32\_Currenttime 166  
 Win32\_LocalTime 166  
 Win32\_LogicalDisk 325  
 Win32 OpenSSH 444  
 Win32\_Product 404  
 Win32\_SecurityDescriptor 373

- Win32\_Share 371, 372
  - Win32\_Trustee 373
  - Win32\_UserAccount 117
  - Win32\_VideoController 310, 320
  - Windows 8 314, 403
  - Windows 10 2, 24, 30, 276
    - Anniversary Update 24
  - Windows as a Service 24
  - Windows Management Framework 3
  - Windows Management Instrumentation 28
  - Windows Nano Server 30
  - Windows PowerShell 1, 2
  - Windows Remote Management 249
  - Windows Remote Management siehe WinRM
  - Windows Script Host 23, 31, 128
  - Windows Server 2003 23
  - Windows Server 2012 314, 403
  - Windows Server 2016 2, 24, 30, 276
  - Windows Workflow Foundation 420
  - Windows XP 28
  - WinRM 249, 252, 253, 254
  - WMI 1, 25, 28, 166, 249, 307, 311, 381, 420
    - Command Shell 28
    - Query Language 313
    - Repository 315
  - WMIClass 307, 311
  - WMISEARCHER 307, 311, 313
  - Word 111
  - Workflow 454
  - WorkingSet 104
  - WorkingSet64 81
  - Workspace Settings 434
  - Wörterbuch 111
  - WPF 65, 297, 430
  - WQL 314
  - Write-Clipboard 235
  - Write-Error 217
  - Write-EventLog 250, 421
  - Write-Host 66, 203, 217
  - Write-Output 56
  - Write-Progress 227, 330
  - Write-Tar 18
  - Write-Warn 217
  - WSMan 239, 274
  - WS-Management 249, 250, 253, 274, 308, 310
  - www.IT-Visions.de XXII
- X**
- XML 70, 229, 379
- Y**
- Year 91
- Z**
- Zahl 148
  - Zahlenliteral 148
  - Zeichenkette 150, 151, 158, 219
    - ersetzen 157
    - Operation 156
    - trennen 158
    - verbinden 158
  - Zeilenumbruch 45, 88
    - Pipeline 88
  - Zeitmessung 336
  - Zertifikatsspeicher 1, 237
  - zsh 411
  - Zufallszahl 149, 150
  - Zugriff verweigert 324
  - Zuweisungsoperator 173
  - Zwischenablage 234
  - Zwischenschritt 114