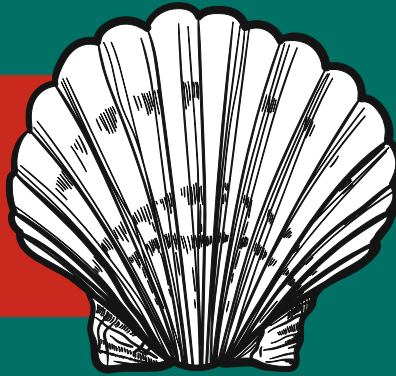


Holger SCHWICHTENBERG

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



Bleiben Sie auf dem Laufenden!

Unser **Computerbuch-Newsletter** informiert Sie monatlich über neue Bücher und Termine. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter:

www.hanser-fachbuch.de/newsletter



Holger Schwichtenberg

PowerShell 7 und Windows PowerShell 5

Das Praxisbuch

5., aktualisierte Auflage

HANSER

Alle in diesem Buch enthaltenen Informationen, Verfahren und Darstellungen wurden nach bestem Wissen zusammengestellt und mit Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Buch enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autoren und Verlag übernehmen infolgedessen keine juristische Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Informationen – oder Teilen davon – entsteht.

Ebenso übernehmen Autoren und Verlag keine Gewähr dafür, dass beschriebene Verfahren usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Buch berechtigt deshalb auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.



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.

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren) – auch nicht für Zwecke der Unterrichtsgestaltung – reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2022 Carl Hanser Verlag München, www.hanser-fachbuch.de

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	XXIV
Über den Autor	XXXII
Teil A: PowerShell-Basiswissen	1
1 Fakten zur PowerShell	3
1.1 Was ist die PowerShell?	3
1.2 Geschichte der PowerShell	4
1.3 Welche Varianten und Versionen der PowerShell gibt es?	6
1.4 Windows PowerShell versus PowerShell Core versus PowerShell 7.x	6
1.5 Motivation zur PowerShell	8
1.6 Betriebssysteme mit vorinstallierter PowerShell	11
1.7 Support der PowerShell	13
1.8 Einflussfaktoren auf die Entwicklung der PowerShell	15
1.9 Anbindung an Klassenbibliotheken	16
1.10 PowerShell versus WSH	17
2 Erste Schritte mit der PowerShell	20
2.1 Windows PowerShell herunterladen und auf anderen Windows-Betriebssystemen installieren	20
2.2 Die Windows PowerShell testen	24
2.3 Woher kommen die PowerShell-Befehle?	33
2.4 PowerShell Community Extensions (PSCX) herunterladen und installieren ...	34
2.5 Den Windows PowerShell-Editor „ISE“ verwenden	41
2.6 PowerShell 7 installieren und testen	45
3 Einzelbefehle der PowerShell	57
3.1 Commandlets	57
3.2 Aliase	70
3.3 Ausdrücke	78
3.4 Externe Befehle (klassische Kommandozeilenbefehle)	79
3.5 Dateinamen	81

4	Hilfefunktionen	82
4.1	Auflisten der verfügbaren Befehle	82
4.2	Praxistipp: Den Standort eines Kommandozeilenbefehls suchen	83
4.3	Anzahl der Befehle	84
4.4	Volltextsuche	86
4.5	Erläuterungen zu den Befehlen	86
4.6	Hilfe zu Parametern	87
4.7	Hilfe mit Show-Command	89
4.8	Hilfefenster	90
4.9	Allgemeine Hilfetexte	92
4.10	Aktualisieren der Hilfedateien	92
4.11	Online-Hilfe	94
4.12	Fehlende Hilfetexte	95
4.13	Dokumentation der .NET-Klassen	96
5	Objektorientiertes Pipelining	98
5.1	Befehlsübersicht	98
5.2	Pipeline-Operator	99
5.3	.NET-Objekte in der Pipeline	100
5.4	Pipeline Processor	101
5.5	Pipelining von Parametern	103
5.6	Pipelining von klassischen Befehlen	105
5.7	Zeilenumbrüche in Pipelines	107
5.8	Schleifen	108
5.9	Zugriff auf einzelne Objekte aus einer Menge	111
5.10	Zugriff auf einzelne Werte in einem Objekt	112
5.11	Methoden ausführen	114
5.12	Analyse des Pipeline-Inhalts	116
5.13	Filtern	131
5.14	Zusammenfassung von Pipeline-Inhalten	136
5.15	„Kastrierung“ von Objekten in der Pipeline	136
5.16	Sortieren	137
5.17	Duplikate entfernen	138
5.18	Gruppierung	139
5.19	Objekte verbinden mit Join-String	145
5.20	Berechnungen	146
5.21	Zwischenschritte in der Pipeline mit Variablen	146
5.22	Verzweigungen in der Pipeline	147
5.23	Vergleiche zwischen Objekten	149
5.24	Weitere Praxislösungen	150

6	PowerShell-Skripte	152
6.1	Skriptdateien	152
6.2	Start eines Skripts	154
6.3	Aliase für Skripte verwenden	155
6.4	Parameter für Skripte	156
6.5	Skripte dauerhaft einbinden (Dot Sourcing)	157
6.6	Das aktuelle Skriptverzeichnis	158
6.7	Sicherheitsfunktionen für PowerShell-Skripte	158
6.8	Skripte mit vollen Rechten (Elevation)	160
6.9	Blockierte PowerShell-Skripte	161
6.10	PowerShell-Skripte im Kontextmenü des Windows Explorers	162
6.11	Anforderungsdefinitionen von Skripten	164
6.12	Skripte anhalten	165
6.13	Versionierung und Versionsverwaltung von Skripten	165
7	PowerShell-Skriptsprache	168
7.1	Hilfe zur PowerShell-Skriptsprache	168
7.2	Befehlstrennung	168
7.3	Kommentare	169
7.4	Variablen	170
7.5	Variablenbedingungen	182
7.6	Zahlen	183
7.7	Zeichenketten (Strings)	187
7.8	Reguläre Ausdrücke	197
7.9	Datum und Uhrzeit	203
7.10	Objekte	204
7.11	Arrays	205
7.12	ArrayList	208
7.13	Assoziative Arrays (Hash-Tabellen)	209
7.14	Operatoren	210
7.15	Überblick über die Kontrollkonstrukte	214
7.16	Bedingungen	219
7.17	Unterroutinen (Prozedur/Funktionen)	222
7.18	Eingebaute Funktionen	228
7.19	Fehlerausgabe	229
7.20	Fehlerbehandlung	231
7.21	Laufzeitfehler erzeugen	243
7.22	Objektorientiertes Programmieren mit Klassen	243

8	Ausgaben	246
8.1	Ausgabe-Commandlets	246
8.2	Benutzerdefinierte Tabellenformatierung	249
8.3	Benutzerdefinierte Listenausgabe	251
8.4	Mehrspaltige Ausgabe	251
8.5	Out-GridView	252
8.6	Standardausgabe	254
8.7	Einschränkung der Ausgabe	256
8.8	Seitenweise Ausgabe	256
8.9	Ausgabe einzelner Werte	258
8.10	Details zum Ausgabeoperator	260
8.11	Ausgabe von Methodenergebnissen und Unterobjekten in Pipelines	263
8.12	Ausgabe von Methodenergebnissen und Unterobjekten in Zeichenketten	264
8.13	Unterdrückung der Ausgabe	264
8.14	Ausgaben an Drucker	265
8.15	Ausgaben in Dateien	266
8.16	Umleitungen (Redirection)	266
8.17	Fortschrittsanzeige	267
8.18	Sprachausgabe	267
9	Das PowerShell-Navigationsmodell (PowerShell Provider)	269
9.1	Einführungsbeispiel: Navigation in der Registrierungsdatenbank	269
9.2	Provider und Laufwerke	270
9.3	Navigationsbefehle	272
9.4	Pfadangaben	273
9.5	Beispiel	275
9.6	Eigene Laufwerke definieren	276
10	Fernausführung (Remoting)	277
10.1	RPC-Fernabfrage ohne WS-Management	278
10.2	Anforderungen an PowerShell Remoting	279
10.3	Rechte für PowerShell-Remoting	280
10.4	Einrichten von PowerShell Remoting	280
10.5	Überblick über die Fernausführungs-Commandlets	283
10.6	Interaktive Fernverbindungen im Telnet-Stil	283
10.7	Fernausführung von Befehlen	285
10.8	Parameterübergabe an die Fernausführung	289
10.9	Fernausführung von Skripten	290
10.10	Ausführung auf mehreren Computern	291
10.11	Sitzungen	292
10.12	Implizites Remoting	297

10.13	Zugriff auf entfernte Computer außerhalb der eigenen Domäne	298
10.14	Verwaltung des WS-Management-Dienstes	301
10.15	PowerShell Direct für Hyper-V	302
10.16	Praxislösung zu PowerShell Direct	304
11	PowerShell-Werkzeuge	307
11.1	PowerShell-Standardkonsole	307
11.2	Windows Terminal	322
11.3	Erweiterung der Konsolen	327
11.4	PowerShell Integrated Scripting Environment (ISE)	329
11.5	PowerShell Script Analyzer	340
11.6	PowerShell Analyzer	345
11.7	PowerShell Tools for Visual Studio	346
11.8	PowerShell Pro Tools for Visual Studio	348
11.9	Visual Studio Developer PowerShell	348
11.10	NuGet Package Manager Console (PMC)	351
11.11	Visual Studio Code mit PowerShell-Erweiterung	352
11.12	PowerShell-Erweiterungen für andere Editoren	354
11.13	PowerShell Web Access (PSWA)	355
11.14	Azure Cloud Shell	360
11.15	ISE Steroids	360
11.16	PowerShellPlus	361
11.17	PoshConsole	364
11.18	PowerGUI	365
11.19	PrimalScript	365
11.20	CIM Explorer for PowerShell ISE	367
12	Windows PowerShell Core 5.1 in Windows Nano Server	369
12.1	Installation	369
12.2	PowerShell-Skriptsprache	369
12.3	Werkzeuge	369
12.4	Fehlende Funktionen	370
13	PowerShell 7 für Windows, Linux und macOS	371
13.1	Motivation für den Einsatz der PowerShell 7 auf Linux und macOS	371
13.2	Basis der PowerShell 7	372
13.3	Identifizierung der PowerShell 7	373
13.4	Funktionsumfang der PowerShell 7	373
13.5	Entfallene Befehle in PowerShell 7	376
13.6	Erweiterungsmodule nutzen in PowerShell 7	382
13.7	Geänderte Funktionen in PowerShell 7	387

13.8	Neue Funktionen der PowerShell 7	389
13.9	PowerShell 7-Konsole	392
13.10	Praxislösung: Fallunterscheidung für PowerShell-Varianten	393
13.11	VSCoDe-PowerShell als Editor für PowerShell 7	394
13.12	Verwendung von PowerShell 7 auf Linux und macOS	398
13.13	PowerShell-Remoting via SSH	404
13.14	Performance-Vorteile der PowerShell 7	407
13.15	Dokumentation zur PowerShell 7	408
13.16	Quellcode zur PowerShell 7	410

Teil B: PowerShell-Aufbauwissen **413**

14 Verwendung von .NET-Klassen **415**

14.1	.NET versus .NET Core	415
14.2	Ermitteln der verwendeten .NET-Version	416
14.3	.NET-Bibliotheken	417
14.4	Microsoft Docs	419
14.5	Überblick über die Verwendung von .NET-Klassen	420
14.6	Erzeugen von Instanzen	420
14.7	Parameterbehaftete Konstruktoren	422
14.8	Initialisierung von Objekten	423
14.9	Nutzung von Attributen und Methoden	424
14.10	Statische Mitglieder in .NET-Klassen und statische .NET-Klassen	426
14.11	Generische Klassen nutzen	429
14.12	Zugriff auf bestehende Objekte	431
14.13	Laden von Assemblies	431
14.14	Liste der geladenen Assemblies	433
14.15	Verwenden von NuGet-Assemblies	434
14.16	Objektanalyse	436
14.17	Aufzählungstypen (Auflistungen/Enumerationen)	437

15 Verwendung von COM-Klassen **441**

15.1	Unterschiede zwischen COM und .NET	441
15.2	Erzeugen von COM-Instanzen	442
15.3	Abruf der Metadaten	442
15.4	Nutzung von Attributen und Methoden	443
15.5	Liste aller COM-Klassen	444
15.6	Holen bestehender COM-Instanzen	445
15.7	Distributed COM (DCOM)	445

16	Zugriff auf die Windows Management Instrumentation (WMI)	447
16.1	Einführung in WMI	447
16.2	WMI in der PowerShell	474
16.3	Open Management Infrastructure (OMI)	476
16.4	Abruf von WMI-Objektmenngen	476
16.5	Fernzugriffe	477
16.6	Filtern und Abfragen	478
16.7	Liste aller WMI-Klassen	481
16.8	Hintergrundwissen: WMI-Klassenprojektion mit dem PowerShell-WMI-Objektadapter	482
16.9	Beschränkung der Ausgabeliste bei WMI-Objekten	486
16.10	Zugriff auf einzelne Mitglieder von WMI-Klassen	488
16.11	Werte setzen in WMI-Objekten	488
16.12	Umgang mit WMI-Datumsangaben	490
16.13	Methodenaufrufe	491
16.14	Neue WMI-Instanzen erzeugen	492
16.15	Instanzen entfernen	493
16.16	Commandlet Definition XML-Datei (CDXML)	493
17	Dynamische Objekte	497
17.1	Erweitern bestehender Objekte	497
17.2	Komplett dynamische Objekte	499
18	Einbinden von C# und Visual Basic .NET	501
19	Win32-API-Aufrufe	503
20	Benutzereingaben	506
20.1	Read-Host	506
20.2	Benutzerauswahl	507
20.3	Grafischer Eingabedialog	508
20.4	Dialogfenster	509
20.5	Authentifizierungsdialg	509
20.6	Zwischenablage (Clipboard)	511
21	Fehlersuche	512
21.1	Detailinformationen	512
21.2	Einzelstschrittmodus	513
21.3	Zeitmessung	514
21.4	Ablaufverfolgung (Tracing)	515
21.5	Erweiterte Protokollierung aktivieren	517

21.6	Script-Debugging in der ISE	518
21.7	Kommandozeilenbasiertes Script-Debugging	518
22	Transaktionen	520
22.1	Commandlets für Transaktionen	520
22.2	Start und Ende einer Transaktion	521
22.3	Zurücksetzen der Transaktion	522
22.4	Mehrere Transaktionen	523
23	Standardeinstellungen ändern mit Profilskripten	524
23.1	Profilpfade	524
23.2	Ausführungsreihenfolge	526
23.3	Beispiel für eine Profildatei	526
23.4	Starten der PowerShell ohne Profilskripte	528
24	Digitale Signaturen für PowerShell-Skripte	529
24.1	Zertifikat erstellen	529
24.2	Skripte signieren	531
24.3	Verwenden signierter Skripte	533
24.4	Mögliche Fehlerquellen	533
25	Hintergrundaufträge („Jobs“)	534
25.1	Voraussetzungen	534
25.2	Architektur	534
25.3	Starten eines Hintergrundauftrags	535
25.4	Hintergrundaufträge abfragen	536
25.5	Warten auf einen Hintergrundauftrag	537
25.6	Abbrechen und Löschen von Aufträgen	537
25.7	Analyse von Fehlermeldungen	537
25.8	Fernausführung von Hintergrundaufträgen	538
25.9	Praxislösung: Einen Job auf mehreren Computern starten	538
26	Geplante Aufgaben und zeitgesteuerte Jobs	540
26.1	Geplante Aufgaben (Scheduled Tasks)	540
26.2	Zeitgesteuerte Jobs	544
27	PowerShell-Workflows	550
27.1	Ein erstes Beispiel	550
27.2	Unterschiede zu einer Function bzw. einem Skript	554
27.3	Einschränkungen bei Workflows	555
27.4	Workflows in der Praxis	556
27.5	Workflows in Visual Studio erstellen	564

28	Ereignissystem	582
28.1	WMI-Ereignisse	582
28.2	WMI-Ereignisabfragen	582
28.3	WMI-Ereignisse seit PowerShell 1.0	584
28.4	Registrieren von WMI-Ereignisquellen seit PowerShell 2.0	585
28.5	Auslesen der Ereignisliste	586
28.6	Reagieren auf Ereignisse	588
28.7	WMI-Ereignisse seit PowerShell-Version 3.0	590
28.8	Registrieren von .NET-Ereignissen	590
28.9	Erzeugen von Ereignissen	591
29	Datenbereiche und Datendateien	593
29.1	Datenbereiche	593
29.2	Datendateien	595
29.3	Mehrsprachigkeit/Lokalisierung	596
30	Desired State Configuration (DSC)	599
30.1	Grundprinzipien	600
30.2	DSC für PowerShell 7	600
30.3	Ressourcen	601
30.4	Verfügbare DSC-Ressourcen	602
30.5	Eigenschaften einer Ressource	605
30.6	Aufbau eines DSC-Dokuments	605
30.7	Commandlets für die Arbeit mit DSC	606
30.8	Ein erstes DSC-Beispiel	606
30.9	Kompilieren und Anwendung eines DSC-Dokuments	607
30.10	Variablen in DSC-Dateien	609
30.11	Parameter für DSC-Dateien	610
30.12	Konfigurationsdaten	611
30.13	Entfernen einer DSC-Konfiguration	614
30.14	DSC Pull Server	617
30.15	DSC-Praxislösung 1: IIS installieren	624
30.16	DSC-Praxislösung 2: Software installieren	626
30.17	DSC-Praxislösung 3: Software deinstallieren	628
30.18	Realisierung einer DSC-Ressource	629
30.19	Weitere Möglichkeiten	629
31	PowerShell-Snap-Ins	630
31.1	Einbinden von Snap-Ins	630
31.2	Liste der Commandlets	634

32 PowerShell-Module	635
32.1 Überblick über die Commandlets	635
32.2 Modularchitektur	636
32.3 Aufbau eines Moduls	637
32.4 Module aus dem Netz herunterladen und installieren mit PowerShellGet	638
32.5 Module manuell installieren	644
32.6 Doppeldeutige Namen	644
32.7 Auflisten der verfügbaren Module	646
32.8 Importieren von Modulen	647
32.9 Entfernen von Modulen	650
33 Ausgewählte PowerShell-Erweiterungen	651
33.1 PowerShell-Module in Windows 8.0 und Windows Server 2012	652
33.2 PowerShell-Module in Windows 8.1 und Windows Server 2012 R2	654
33.3 PowerShell-Module in Windows 10 und Windows Server 2019	656
33.4 PowerShell Community Extensions (PSCX)	660
33.5 PowerShellPack	664
33.6 www.IT-Visions.de: PowerShell Extensions	666
33.7 Quest Management Shell for Active Directory	666
33.8 Microsoft Exchange Server	668
33.9 System Center Virtual Machine Manager	669
33.10 PowerShell Management Library for Hyper-V (pshyperv)	669
33.11 PowerShell Configurator (PSConfig)	670
34 Delegierte Administration/Just Enough Administration (JEA) ..	672
34.1 JEA-Konzept	672
34.2 PowerShell-Sitzungskonfiguration erstellen	672
34.3 Sitzungskonfiguration nutzen	676
34.4 Delegierte Administration per Webseite	677
35 Tipps und Tricks zur PowerShell	678
35.1 Alle Anzeigen löschen	678
35.2 Befehlsgeschichte	678
35.3 System- und Hostinformationen	679
35.4 Anpassen der Eingabeaufforderung (Prompt)	680
35.5 PowerShell-Befehle aus anderen Anwendungen heraus starten	681
35.6 ISE erweitern	682
35.7 PowerShell für Gruppenrichtlinienskripte	683
35.8 Einblicke in die Interna der Pipeline-Verarbeitung	686

Teil C:PowerShell im Praxiseinsatz	687
36 Dateisystem	689
36.1 Laufwerke	690
36.2 Ordnerinhalte	695
36.3 Dateieigenschaften verändern	702
36.4 Eigenschaften ausführbarer Dateien	703
36.5 Kurznamen	705
36.6 Lange Pfade	705
36.7 Dateisystemoperationen	706
36.8 Praxislösung: Dateien umorganisieren	706
36.9 Praxislösung: Zufällige Dateisystemstruktur erzeugen	708
36.10 Praxislösung: Leere Ordner löschen	709
36.11 Praxislösung: Geschwindigkeitsmessung des Dateisystems (beim Kopieren von Dateien)	711
36.12 Einsatz von Robocopy in der PowerShell	712
36.13 NTFS-Komprimierung	715
36.14 Dateisystemkataloge	716
36.15 Papierkorb leeren	716
36.16 Dateieigenschaften lesen	717
36.17 Praxislösung: Fotos nach Aufnahmedatum sortieren	717
36.18 Datei-Hash	718
36.19 Finden von Duplikaten	719
36.20 Verknüpfungen im Dateisystem	721
36.21 Komprimierung	726
36.22 Dateisystemfreigaben	730
36.23 Überwachung des Dateisystems	741
36.24 Dateiversionsverlauf	742
36.25 Windows Explorer öffnen	743
36.26 Windows Server Backup	743
37 Festplattenverschlüsselung mit BitLocker	745
37.1 Übersicht über das BitLocker-Modul	746
37.2 Verschlüsseln eines Laufwerks	747
38 Dokumente	748
38.1 Textdateien	748
38.2 CSV-Dateien	750
38.3 Analysieren von Textdateien	753
38.4 INI-Dateien	757
38.5 XML-Dateien	757

38.6	HTML- und Markdown-Dateien	769
38.7	JSON-Dateien	772
38.8	Binärdateien	783
38.9	Praxislösung: Grafikdateien verändern	784
38.10	Praxislösung: Drucken vieler Dateien	785
39	Microsoft Office	786
39.1	Allgemeine Informationen zur Office-Automatisierung per PowerShell	786
39.2	Praxislösung: Terminserien aus Textdateien anlegen in Outlook	787
39.3	Praxislösung: Outlook-Termine anhand von Suchkriterien löschen	789
39.4	Praxislösung: Grafiken aus einem Word-Dokument (DOCX) extrahieren	790
40	Datenbanken	793
40.1	ADO.NET-Grundlagen	793
40.2	Beispieldatenbank	799
40.3	Datenzugriff mit den Bordmitteln der PowerShell	800
40.4	Hilfsroutinen für den Datenbankzugriff (DBUtil.ps1)	813
40.5	Datenzugriff mit den PowerShell-Erweiterungen	816
40.6	Datenbankzugriff mit SQLPS	820
40.7	Datenbankzugriff mit SQLPSX	820
41	Microsoft-SQL-Server-Administration	821
41.1	PowerShell-Integration im SQL Server Management Studio	822
41.2	SQL-Server-Laufwerk „SQLSERVER:“	823
41.3	Die SQLPS-Commandlets	826
41.4	Die SQL Server Management Objects (SMO)	828
41.5	SQLPSX	831
41.6	Microsoft-SQL-Server-Administration mit der PowerShell in der Praxis	838
42	ODBC-Datenquellen	844
42.1	ODBC-Treiber und -Datenquellen auflisten	845
42.2	Anlegen einer ODBC-Datenquelle	846
42.3	Zugriff auf eine ODBC-Datenquelle	847
43	Registrierungsdatenbank (Registry)	849
43.1	Schlüssel auslesen	849
43.2	Schlüssel anlegen und löschen	850
43.3	Laufwerke definieren	850
43.4	Werte anlegen und löschen	851
43.5	Werte auslesen	852
43.6	Praxislösung: Windows-Explorer-Einstellungen	853
43.7	Praxislösung: Massenanlegen von Registry-Schlüsseln	853

44	Computer- und Betriebssystemverwaltung	855
44.1	Computerinformationen	855
44.2	Versionsnummer des Betriebssystems	857
44.3	Zeitdauer seit dem letzten Start des Betriebssystems	857
44.4	BIOS- und Startinformationen	858
44.5	Windows-Produktaktivierung	859
44.6	Umgebungsvariablen	859
44.7	Schriftarten	863
44.8	Computernamen und Domäne	863
44.9	Herunterfahren und Neustarten	864
44.10	Windows Updates installieren	865
44.11	Wiederherstellungspunkte verwalten	869
45	Windows Defender	870
46	Hardwareverwaltung	871
46.1	Hardwarebausteine	871
46.2	Plug-and-Play-Geräte	873
46.3	Druckerverwaltung (ältere Betriebssysteme)	873
46.4	Druckerverwaltung (seit Windows 8 und Windows Server 2012)	875
47	Softwareverwaltung	877
47.1	Softwareinventarisierung	877
47.2	Installation von Anwendungen	880
47.3	Deinstallation von Anwendungen	881
47.4	Praxislösung: Installationstest	882
47.5	Praxislösung: Installierte .NET SDKs aufräumen	883
47.6	Windows 10 Apps verwalten	887
47.7	Installationen mit PowerShell Package Management („OneGet“)	890
47.8	Versionsnummer ermitteln	893
47.9	Servermanager	894
47.10	Windows-Features installieren auf Windows-Clientbetriebssystemen	905
47.11	Praxislösung: IIS-Installation	907
47.12	Softwareeinschränkungen mit dem PowerShell-Modul „AppLocker“	909
48	Prozessverwaltung	915
48.1	Prozesse auflisten	915
48.2	Prozesse starten	916
48.3	Prozesse mit vollen Administratorrechten starten	917
48.4	Prozesse unter einem anderen Benutzerkonto starten	918
48.5	Prozesse beenden	919
48.6	Warten auf das Beenden einer Anwendung	920

49	Windows-Systemdienste	921
49.1	Dienste auflisten	921
49.2	Dienstzustand ändern	924
49.3	Diensteigenschaften ändern	924
49.4	Dienste hinzufügen	925
49.5	Dienste entfernen	926
50	Netzwerk	927
50.1	Netzwerkkonfiguration	927
50.2	DNS-Client-Konfiguration	932
50.3	DNS-Namensauflösung	936
50.4	Erreichbarkeit prüfen (Ping)	937
50.5	Windows Firewall	938
50.6	Remote Desktop (RDP) einrichten	945
50.7	E-Mails senden (SMTP)	946
50.8	Auseinandernehmen von E-Mail-Adressen	947
50.9	Abruf von Daten von einem HTTP-Server	947
50.10	Praxislösung: Linkprüfer für eine Website	954
50.11	Aufrufe von SOAP-Webdiensten	957
50.12	Aufruf von REST-Diensten	960
50.13	File Transfer Protocol (FTP)	962
50.14	Hintergrunddatentransfer mit BITS	963
51	Ereignisprotokolle (Event Log)	967
51.1	Protokolleinträge auslesen	967
51.2	Ereignisprotokolle erzeugen	969
51.3	Protokolleinträge erzeugen	969
51.4	Protokollgröße festlegen	969
51.5	Protokolleinträge löschen	969
52	Leistungsdaten (Performance Counter)	970
52.1	Zugriff auf Leistungsindikatoren über WMI	970
52.2	Get-Counter	971
53	Sicherheitseinstellungen	973
53.1	Aktueller Benutzer	973
53.2	Grundlagen	974
53.3	Zugriffsrechtelisten auslesen	979
53.4	Einzelne Rechteinträge auslesen	980
53.5	Besitzer auslesen	982
53.6	Benutzer und SID	982

53.7	Hinzufügen eines Rechteeintrags zu einer Zugriffsrechteliste	986
53.8	Entfernen eines Rechteeintrags aus einer Zugriffsrechteliste	988
53.9	Zugriffsrechteliste übertragen	990
53.10	Zugriffsrechteliste über SDDL setzen	991
53.11	Zertifikate verwalten	992
54	Optimierungen und Problemlösungen	995
54.1	PowerShell-Modul „TroubleshootingPack“	995
54.2	PowerShell-Modul „Best Practices“	999
55	Active Directory	1001
55.1	Benutzer- und Gruppenverwaltung mit WMI	1003
55.2	Einführung in System.DirectoryServices	1003
55.3	Basiseigenschaften	1015
55.4	Benutzer- und Gruppenverwaltung im Active Directory	1017
55.5	Verwaltung der Organisationseinheiten	1025
55.6	Suche im Active Directory	1026
55.7	Navigation im Active Directory mit den PowerShell Extensions	1033
55.8	Verwendung der Active-Directory-Erweiterungen von www.IT-Visions.de	1034
55.9	PowerShell-Modul „Active Directory“ (ADPowerShell)	1036
55.10	PowerShell-Modul „ADDSDeployment“	1065
55.11	Informationen über die Active Directory-Struktur	1068
56	Gruppenrichtlinien	1071
56.1	Verwaltung der Gruppenrichtlinien	1071
56.2	Verknüpfung der Gruppenrichtlinien	1073
56.3	Gruppenrichtlinienberichte	1075
56.4	Gruppenrichtlinienvererbung	1077
56.5	Weitere Möglichkeiten	1078
57	Lokale Benutzer und Gruppen	1079
57.1	Modul „Microsoft.PowerShell.LocalAccounts“	1079
57.2	Lokale Benutzerverwaltung in älteren PowerShell-Versionen	1080
58	Microsoft Exchange Server	1083
58.1	Daten abrufen	1083
58.2	Postfächer verwalten	1084
58.3	Öffentliche Ordner verwalten	1085
59	Internet Information Services (IIS)	1086
59.1	Überblick	1086
59.2	Navigationsprovider	1088

59.3	Anlegen von Websites	1090
59.4	Praxislösung: Massenanlegen von Websites	1091
59.5	Ändern von Website-Eigenschaften	1093
59.6	Anwendungspool anlegen	1094
59.7	Virtuelle Verzeichnisse und IIS-Anwendungen	1095
59.8	Website-Zustand ändern	1095
59.9	Anwendungspools starten und stoppen	1096
59.10	Löschen von Websites	1096
60	Virtuelle Systeme mit Hyper-V	1097
60.1	Das Hyper-V-Modul von Microsoft	1098
60.2	Die ersten Schritte mit dem Hyper-V-Modul	1100
60.3	Virtuelle Maschinen anlegen	1104
60.4	Umgang mit virtuellen Festplatten	1110
60.5	Konfiguration virtueller Maschinen	1113
60.6	Praxislösungen: Ressourcennutzung überwachen	1117
60.7	Dateien kopieren in virtuelle Systeme	1119
60.8	PowerShell Management Library for Hyper-V (für ältere Betriebssysteme)	1120
61	Windows Nano Server	1123
61.1	Das Konzept von Nano Server	1123
61.2	Einschränkungen von Nano Server	1125
61.3	Varianten des Nano Servers	1127
61.4	Installation eines Nano Servers	1127
61.5	Docker-Image	1128
61.6	Fernverwaltung mit PowerShell	1129
61.7	Windows Update auf einem Nano Server	1131
61.8	Nachträgliche Paketinstallation	1131
61.9	Abgespeckter IIS unter Nano Server	1133
61.10	Nano-Serververwaltung aus der Cloud heraus	1134
62	Docker-Container	1135
62.1	Container-Varianten für Windows	1135
62.2	Docker-Installation auf aktuellem Windows 10 und Windows 11	1139
62.3	Docker-Installation auf älteren Windows 10-Clients	1147
62.4	Docker-Installation auf Windows Server	1149
62.5	Docker PowerShell installieren	1151
62.6	Docker-Basiswissen	1152
62.7	Container mit modernem .NET	1155
62.8	Container mit IIS-Webserver und klassischem ASP.NET	1164
62.9	Container mit Linux und PowerShell 7	1173

62.10	Container mit Linux und Microsoft SQL Server	1175
62.11	Docker-Container mit Visual Studio	1177
62.12	Weitere Container-Befehle	1182
63	Microsoft Azure	1188
63.1	Azure-Konzepte	1188
63.2	Kommandozeilenwerkzeuge für die Azure-Verwaltung	1190
63.3	Benutzeranmeldung und Informationsabfrage	1193
63.4	Azure Ressourcen-Gruppen	1194
63.5	Azure Web-Apps	1194
63.6	Azure SQL Server	1196
63.7	Azure Kubernetes Services (AKS)	1197
63.8	Azure DevOps (ADO)	1221
64	Grafische Benutzeroberflächen (GUI)	1242
64.1	Einfache Nachfragedialoge	1242
64.2	Einfache Eingabe mit Inputbox	1243
64.3	Komplexere Eingabemasken	1244
64.4	Universelle Objektdarstellung	1246
64.5	WPF PowerShell Kit (WPK)	1247
64.6	Direkte Verwendung von WPF	1255
Teil D: Profiwissen – Erweitern der PowerShell		1257
65	Unit Tests mit Pester	1259
65.1	Einführung in das Konzept des Unit Testing	1259
65.2	Pester installieren	1260
65.3	Befehle in Pester	1260
65.4	Testen einer PowerShell-Funktion	1261
65.5	Testgenerierung	1262
65.6	Tests starten	1262
65.7	Prüf-Operationen	1264
65.8	Mock-Objekte	1264
65.9	Test von Dateisystemoperationen	1265
66	Entwicklung von Commandlets in der PowerShell-Skriptsprache	1267
66.1	Aufbau eines skriptbasierten Commandlets	1267
66.2	Verwendung per Dot Sourcing	1269
66.3	Parameterfestlegung	1270
66.4	Fortgeschrittene Funktion (Advanced Function)	1276

66.5	Mehrere Parameter und Parametersätze	1279
66.6	Unterstützung für Sicherheitsabfragen (-whatif und -confirm)	1281
66.7	Kaufmännisches Beispiel: Test-CustomerID	1283
66.8	Erweitern bestehender Commandlets durch Proxy-Commandlets	1286
66.9	Dokumentation	1292
67	Entwicklung eigener Commandlets mit C#	1296
67.1	Technische Voraussetzungen	1297
67.2	Grundkonzept der .NET-basierten Commandlets	1299
67.3	Schrittweise Erstellung eines minimalen Commandlets	1301
67.4	Erstellung eines Commandlets mit einem Rückgabeobjekt	1309
67.5	Erstellung eines Commandlets mit mehreren Rückgabeobjekten	1311
67.6	Erstellen eines Commandlets mit Parametern	1315
67.7	Verarbeiten von Pipeline-Eingaben	1317
67.8	Verkettung von Commandlets	1320
67.9	Fehlersuche in Commandlets	1324
67.10	Statusinformationen	1327
67.11	Unterstützung für Sicherheitsabfragen (-whatif und -confirm)	1332
67.12	Festlegung der Hilfeinformationen	1334
67.13	Erstellung von Commandlets für den Zugriff auf eine Geschäftsanwendung	1339
67.14	Konventionen für Commandlets	1340
67.15	Weitere Möglichkeiten	1342
68	PowerShell-Module erstellen	1343
68.1	Erstellen eines Skriptmoduls	1343
68.2	Praxislösung: Umwandlung einer Skriptdatei in ein Modul	1345
68.3	Erstellen eines Moduls mit Binärdateien	1345
68.4	Erstellen eines Moduls mit Manifest	1346
68.5	Erstellung eines Manifest-Moduls mit Visual Studio	1353
69	Hosting der PowerShell	1355
69.1	Voraussetzungen für das Hosting	1356
69.2	Hosting mit PSHost	1357
69.3	Vereinfachtes Hosting seit PowerShell 2.0	1360
Anhang A: Crashkurs Objektorientierung		1363
Anhang B: Crashkurs .NET		1371
B.1	Was ist das .NET Framework?	1374
B.2	Was ist .NET Core/.NET?	1375
B.3	Eigenschaften von .NET	1376

B.4	.NET-Klassen	1377
B.5	Namensgebung von .NET-Klassen (Namensräume)	1377
B.6	Namensräume und Softwarekomponenten	1379
B.7	Bestandteile einer .NET-Klasse	1380
B.8	Vererbung	1381
B.9	Schnittstellen	1381
Anhang C: Weitere Informationen im Internet		1382
Anhang D: Abkürzungsverzeichnis		1383
Stichwortverzeichnis		1407

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?	<input type="radio"/> Ja	<input type="radio"/> 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)	<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 alle Möglichkeiten der PowerShell 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 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?	<input type="radio"/> Ja	<input type="radio"/> 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 Einflüssen 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 Lösungswort **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) 11.5.2 / 22.03.2018



Hilfe

- Alle installierten Module
Get-Module -ListAvailable | # Name, ModulTyp, 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-Watch | Get-Member

Wichtige Navigations-Commandlets

Mit den Navigations-Commandlets kann man nicht nur in Dateisystem, sondern auch andere Flächen und hierarchischen Mengen arbeiten. z.B. Registry (HKLM, HKCU), Umgebungsvariablen (env), Zertifikate (cert), Active Directory (AD), usw. arbeiten. z.B.

Dir HKLM://Software/
New-Item HKLM://Software/ITVisions
RD HKLM://Software/ITVisions

Get-PSDrive	Laufwerkliste
Get-Location (pwd)	Abrufen des aktuellen Standorts
Set-Location (cd)	Festlegung des aktuellen Standorts
Get-Item (g)	Holt ein Element
Get-Childitem (dir, ls, gci)	Auflisten der Unter Elemente
Get-Content (type, cat, gc)	Abrufen eines Elementinhalts (z.B. Dateiinhalt)
Set-Content (li)	Elementinhalt festlegen
Add-Content (cat, add)	Elementinhalt ergänzen
New-Item (ni, mkdir)	Erstellen eines Elements (Art oder Date)
Get-ItemProperty (gpi)	Attribut abrufen
Set-ItemProperty (spi)	Attribut eines Elements festlegen
Remove-Item (del, rmdir, mv, erase)	Obj. löschen wenn nicht vorhanden
Move-Item (move, mv)	Element verschieben
Copy-Item (copy, cp, cpi)	Element kopieren
Rename-Item (mv, 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	Abrufen beliebiger Objekte aus dem AD
Get-User, 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 Kennworts
Get-ADGroupMember	Liste der Gruppenmitglieder
Add-ADGroupMember	Mitglied einer Gruppe hinzufügen
Remove-ADGroupMember	Mitglied aus einer Gruppe entfernen

Weitere wichtige Commandlets

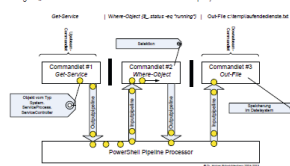
Get-Date / Set-Date	Datum und Zeit abrufen/festlegen
Get-Service	Windows-Systemdienste
Start-Stop/Suspend/Resume-Service	Dienststatus ändern
Get-Process	Laufende Prozesse
Start-Process/Stop-Process	Prozess starten/beenden
Wait-Process	Warten auf Ende eines Prozesses
Get-Counter	Leistungsindikatoren abrufen
Get-Eventlog	Ereignisprotokolleinträge
Write-Eventlog	Eintrag im Ereignisprotokoll
Get-Eventing	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 | verkett werden. Get-Service | where { \$_.status -eq 'running' } | Out-File c:\temp\laufendediene.txt

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

Die Pipeline befördert .NET-Objekte. Die Beförderung ist synchron (außer bei eigenen „blockierenden“ Commandlets wie Set-Objekt).



Wichtige Pipelining-Commandlets

Where-Object (where, fi)	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 (gmi)	Aufliste der Methoden (Reflection)
Measure-Object (measure)	Berechnung: min, max, sum, average
Compare-Object (compare, diff)	Vergleichen von zwei Objektinstanzen

Vergleichsoperatoren

Da die Zeichen < und > für Umkehrungen der Ausgabemenge verwendet werden, können PowerShell eher ungewöhnliche Operatoren zum Einsatz:

Vergleich unter Ignorierung der Groß-/Kleinschreibung	Vergleich unter Berücksichtigung der Groß-/Kleinschreibung	Bedeutung
-lt / -lti	-lti	Kleiner
-le / -le	-le	Kleiner oder gleich
-gt / -gt	-gt	Größer
-ge / -ge	-ge	Größer oder gleich
-eq / -eq	-eq	Gleich
-ne / -ne	-ne	Nicht gleich
-like / -like	-like	Ähnlichkeit zwischen Zeichenketten, Einsatz von Platzholdern (*) und ? möglich
-notlike / -notlike	-notlike	Keine Ähnlichkeit zwischen Zeichenketten, Einsatz von Platzholdern (*) und ? möglich
-match / -match	-match	Vergleich mit regulärem Ausdruck

Vorderseite der PowerShell-Kurzreferenz

Kurzreferenz ("Cheat Sheet") Windows PowerShell

Autor: Dr. Holger Schwichtenberg (www.IT-Visions.de) 11.5.2 / 22.03.2018



-notmatch / -notmatch	-notmatch	Stimmt nicht mit regulärem Ausdruck überein
-is		Typgleichheit, 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 (alles !) verwendet. Beispiel: ((MB > 150 + \$?) -and 100KB) -and (\$? -lt 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 weiteren Ausgabe
Out-GridView (ogv)	Grafische Tabelle mit Filter- und Sortieroptionen
Out-File	Speichern in Date
Out-Printer (lp)	Ausgabe an Drucker
Out-Clipboard	Ausgabe in Zwischenablage
Out-Speech	Sprachausgabe (TTS/CO)
Out-Null	Die Objekte der Pipeline werden nicht weitergegeben
Read-Host	Eingabe von Konsole einlesen
Import-Export-CSV	CSV-Datei importieren/exportieren
Import-Export-XML	XML-Datei importieren/exportieren

Benutzerdefinierte Tabellenausgabe
Get-Process | @('Label1', \$Expression = {\$_ .ID}; Width=5), @('Label2', \$Expression = {\$_ .ProcessName}; Width=20), @('Label3', \$Expression = {\$_ .WorkingSet4 / 1MB}; Width=1); Format-Table (0:0000:0:0)

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 Unterdruck in \$() geklammert werden "\$(\$Expression) Count: Objekte in der Ergebnismenge"
Einsatz des Formatoperators
Get-Process | % { (0..4) | (1..0,000,00)MB -f \$(\$_.Name), (\$_.WS / 1MB) }
Auflösen einer Zeichenkette als Befehl
\$Befehl = "Get-Service -q"
\$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

Einzige Eigenschaften der Pipeline-Objekte ausgeben (Get-Date).Dir | Out-GridView
(Get-Process) .Name
(Get-Process | sort ws -desc)[0].Name

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

PowerShell-Datentypen

[char], [string]	[byte], [int], [long]	[int?]
[bool]	[single], [double]	[array], [Hashtable]
[DateTime]		[VM], [ADSI]

PowerShell-Skriptsprache

Bedingung
if ((Get-Date).Year -le 2014) { 'AP' } else { 'New' }

Schleifen
for(\$i = 1; \$i -le 10; \$i++) { \$i }
while(\$i -le 10) { \$i; \$i++ }
do { \$i = 1; while (\$i -le 10) {
foreach (\$p in (Get-Process explore)) { \$p.Kill() }
}}
Unterfunktion mit Pflichtparameter und optionalen Parameter
function Get-DLL([Parameter(Mandatory=\$)]\$Name, [string]\$Dir = "") {
return Get-Childitem \$Dir -File "\$Name.dll"
}
Get-DLL -c:Windows\System32

Kommentar
Das 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::Read(300, 500))
Instanzierung und Zugriff auf Instanzmitglieder
\$b = New-Object System.DirectoryServices.DirectoryEntry("WinNT://Server/HT")
\$b.FullName
\$b.Description = "Autor des PowerShell Cheat Sheets"
\$b.SetInfo()

Zusätzliche Assembly laden und nutzen (System.Reflection.Assembly::LoadWithPartialName("Microsoft.VisualBasic"), \$env:obj = [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 Namespace von einem Computer Get- CimClass -Namespace root\cimv2 -Computer MyServer
Liste aller Instanzen einer WMI-Klasse auf einem Computer Get-CimInstance Win32_LogicalDisk -Namespace root\cimv2 -Computer MyServer

WQL-Abfrage auf einem Computer Get-CimInstance Query "Select * from Win32_NetAdapter where adapterType like '802.11' -Computer MyServer

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

Alternativ mit allen 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.powerhell.com
www.gotwired.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