

holger SCHWICHTENBERG

Für alle
PowerShell
Versionen

WINDOWS PowerShell und PowerShell Core

Der schnelle Einstieg



Skriptbasierte Windows-
Systemadministration für
Windows, Linux und macOS

HANSER



Im Internet: Codebeispiele, Feedback-
möglichkeiten und Forum

www.IT-Visions.de
Dr. Holger Schwichtenberg

Schwichtenberg

Windows PowerShell und PowerShell Core Der schnelle Einstieg

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



Hanser Update ist der IT-Blog des Hanser Verlags mit Beiträgen und Praxistipps von unseren Autoren rund um die Themen Online Marketing, Webentwicklung, Programmierung, Softwareentwicklung sowie IT- und Projektmanagement. Lesen Sie mit und abonnieren Sie unsere News unter



www.hanser-fachbuch.de/update



Holger Schwichtenberg

Windows PowerShell und PowerShell Core Der schnelle Einstieg

Skriptbasierte Systemadministration
für Windows, Linux und macOS

HANSER

Der Autor:

Dr. Holger Schwichtenberg, Essen

www.IT-Visions.de

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. Autor 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 Autor 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.

© 2018 Carl Hanser Verlag München, www.hanser-fachbuch.de

Lektorat: Sylvia Hasselbach

Copy editing: Petra Kienle, Fürstenfeldbruck

Umschlagdesign: Marc Müller-Bremer, München, www.rebranding.de

Umschlagrealisation: Stephan Rönigk

Layout: Kösel Media GmbH, Krugzell

Druck und Bindung: Hubert & Co. GmbH & Co. KG BuchPartner, Göttingen

Printed in Germany

Print-ISBN: 978-3-446-45214-5

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

Inhalt

Vorwort	XV
Über den Autor Dr. Holger Schwichtenberg	XXI
1 Erste Schritte mit der PowerShell	1
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
2 Fakten zur PowerShell	23
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

3	Einzelbefehle der PowerShell	35
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
4	Hilfefunktionen	61
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
5	Objektorientiertes Pipelining	79
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
6	PowerShell-Skripte	121
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
7	Die PowerShell-Skriptsprache	135
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
8 Ausgaben	203
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
9 Benutzereingaben	229
9.1 Read-Host	229
9.2 Benutzerauswahl	230
9.3 Grafischer Eingabedialog	231
9.4 Dialogfenster	232
9.5 Authentifizierungsdialog	232
9.6 Zwischenablage (Clipboard)	234
9.7 Aufgaben zu diesem Kapitel	235

10 Das PowerShell-Navigationsmodell (PowerShell Provider)	237
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
11 Fernausführung (Remoting)	249
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 Fernausführungs-Commandlets	255
11.6 Interaktive Fernverbindungen im Telnet-Stil	256
11.7 Fernausführung von Befehlen	257
11.8 Parameterübergabe an die Fernausführung	261
11.9 Fernausführung 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
12 Verwendung von .NET-Klassen	281
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
13	Verwendung von COM-Klassen	301
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
14	Zugriff auf die Windows Management Instrumentation (WMI) ..	307
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
15	Fehlersuche	333
15.1	Detailinformationen	333
15.2	Einzel schrittmodus	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
16	Standardeinstellungen ändern mit Profilskripten	343
16.1	Profilpfade	343
16.2	Ausführungsreihenfolge	345
16.3	Beispiel für eine Profildatei	345
16.4	Starten der PowerShell ohne Profilskripte	346
17	PowerShell-Module	347
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
18	Praxislösungen	365
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
19	PowerShell Core 6.0 für Windows, Linux und macOS	409
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 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) 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 alle 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 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)


Dr. Holger Schwichtenberg



- 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, www.entwickler-lexikon.de, www.windows-scripting.de, 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 sowie Telefon 02 01-64 95 90-0

1

Erste Schritte mit der PowerShell

Das DOS-ähnliche Kommandozeilenfenster hat viele Windows-Versionen in beinahe unveränderter Form überlebt. Mit der Windows PowerShell (WPS) besitzt Microsoft seit dem Jahr 2006 einen Nachfolger, der es mit den Unix-Shells aufnehmen kann und diese in Hinblick auf Eleganz und Robustheit in einigen Punkten auch überbieten kann. Die PowerShell ist eine Adaption des Konzepts von Unix-Shells auf Windows unter Verwendung des .NET Frameworks und mit Anbindung an die Windows Management Instrumentation (WMI). Seit dem Jahr 2017 gibt es die PowerShell auch für Linux und macOS als „PowerShell Core“.

■ 1.1 Was ist die PowerShell?

In einem Satz: Die **Windows PowerShell (WPS)** ist eine .NET-basierte Umgebung für interaktive Systemadministration und Skripting auf der Windows-Plattform. **PowerShell Core (PS Core)** ist eine .NET Core-basierte Umgebung für interaktive Systemadministration und Skripting auf Windows, Linux und MacOS.

Die Kernfunktionen der PowerShell sind:

- Zahlreiche eingebaute Befehle, die „Commandlets“ genannt werden
- Zugang zu allen Systemobjekten, die durch COM-Bibliotheken, das .NET Framework und die Windows Management Instrumentation (WMI) bereitgestellt werden
- Robuster Datenaustausch zwischen Commandlets durch Pipelines basierend auf typisierten Objekten
- Ein einheitliches Navigationsparadigma für verschiedene Speicher (z. B. Dateisystem, Registrierungsdatenbank, Zertifikatsspeicher, Active Directory und Umgebungsvariablen)
- Eine einfach zu erlernende, aber mächtige Skriptsprache mit wahlweise schwacher oder starker Typisierung
- Ein Sicherheitsmodell, das die Ausführung unerwünschter Skripte unterbindet
- Integrierte Funktionen für Ablaufverfolgung und Debugging
- Die PowerShell kann um eigene Befehle erweitert werden.
- Die PowerShell kann in eigene Anwendungen integriert werden (Hosting).

■ 1.2 Windows PowerShell versus PowerShell Core

Die Windows PowerShell 5.1 ist weit mächtiger als die PowerShell Core 6.0, weil die PowerShell Core einen Neustart des PowerShell-Entwicklungsprojekts in Hinblick auf Plattformunabhängigkeit darstellt. In PowerShell Core fehlen viele Commandlets der Grundausstattung der Windows PowerShell und viele der verfügbaren PowerShell-Erweiterungsmodule laufen bisher nicht in der PowerShell Core.

Details zu den Funktionseinschränkungen der PowerShell Core lesen Sie in Kapitel 19 „PowerShell Core 6.0 für Windows, Linux und MacOS“.



TIPP: Wenn Sie unter Windows arbeiten, sollten Sie daher vorerst noch die Windows PowerShell (nach Möglichkeit in der aktuellen Version 5.1) verwenden.

Unter Linux und MacOS gibt es keine Windows PowerShell. Hier können Sie die PowerShell Core 6.0 verwenden. Der Wert der PowerShell Core unter Linux und MacOS liegt in den mächtigen Pipelining- sowie Ein- und Ausgabe-Commandlets. Für konkrete Zugriffe auf das Betriebssystem gibt es hingegen für die PowerShell Core unter MacOS und Linux noch fast keine Commandlets. Man wird also hier immer klassische Linux- und MacOS-Kommandozeilenbefehle mit zeichenkettenbasierter Verarbeitung in die PowerShell einbinden. Wie dies geht, wird im Kapitel 19 erklärt.

■ 1.3 Windows PowerShell herunterladen und auf anderen Windows-Betriebssystemen installieren

Die Windows PowerShell 5.1 ist in Windows 10 (ab Anniversary Update) und Windows Server 2016 bereits im Standard installiert.

Wenn Sie nicht Windows 10 oder Windows Server 2016 benutzen, müssen Sie die PowerShell 5.1 erst installieren.

Die nachträgliche Installation der Windows PowerShell 5.1 ist auf folgenden Betriebssystemen möglich:

- Windows Server 2012 R2
- Windows Server 2012
- Windows 2008 R2
- Windows 8.1
- Windows 7

Die Windows PowerShell 5.1 wird auf diesen Betriebssystemen als Teil des Windows Management Framework 5.1 (WMF) installiert – <https://www.microsoft.com/en-us/download/details.aspx?id=54616>.

Bei der Installation ist zu beachten, dass jeweils das .NET Framework 4.5.2 oder höher vorhanden sein muss. Auch mit .NET Framework 4.6.x und 4.7 funktioniert die PowerShell 5.1.

Das WMF-5.1-Installationspaket betrachtet sich als Update für Windows (KB3191566 für Windows 7 und Windows Server 2008 R2 bzw. KB3191564 für Windows 8.1 und Windows Server 2012 R2 sowie KB3191565 für Server 2012).

<input checked="" type="checkbox"/>	W2K12-KB3191565-x64.msu	20.6 MB
<input checked="" type="checkbox"/>	Win7AndW2K8R2-KB3191566-x64.zip	64.9 MB
<input checked="" type="checkbox"/>	Win7-KB3191566-x86.zip	42.7 MB
<input checked="" type="checkbox"/>	Win8.1AndW2K12R2-KB3191564-x64.msu	19.0 MB
<input checked="" type="checkbox"/>	Win8.1-KB3191564-x86.msu	14.5 MB

Bild 1.1 Installationspaket für PowerShell 5.1 als Erweiterung

Installationsordner

Die Windows PowerShell installiert sich in folgendes Verzeichnis: `%systemroot%\system32\WindowsPowerShell\V1.0` (für 32-Bit-Systeme).



ACHTUNG: Dabei ist das „V1.0“ im Pfad tatsächlich richtig: Microsoft hat dies seit Version 1.0 nicht verändert. Geplant war wohl eine „Side-by-Side“-Installationsoption wie beim .NET Framework. Doch später hat sich Microsoft dann entschieden, dass eine neue PowerShell immer die alte überschreibt.

Auf 64-Bit-Systemen gibt es die PowerShell zweimal, einmal als 64-Bit-Version in `%systemroot%\system32\WindowsPowerShell\V1.0` und einmal als 32-Bit-Version. Letztere findet man unter `%systemroot%\Syswow64\WindowsPowerShell\V1.0`. Die 32-Bit-Version braucht man, wenn man eine Bibliothek nutzen will, für die es keine 64-Bit-Version gibt, z. B. den Zugriff auf Microsoft-Access-Datenbanken.

Es handelt sich auch dabei nicht um einen Tippfehler: Die 64-Bit-Version befindet sich in einem Verzeichnis, das „32“ im Namen trägt, und die 32-Bit-Version in einem Verzeichnis mit „64“ im Namen!

Die 32-Bit-Version der PowerShell und die 64-Bit-Version der PowerShell sieht man im Startmenü: Die 32-Bit-Version hat den Zusatz „(x86)“. Die 64-Bit-Version hat keinen Zusatz. Auch den Editor „ISE“ gibt es in einer 32- und einer 64-Bit-Version.

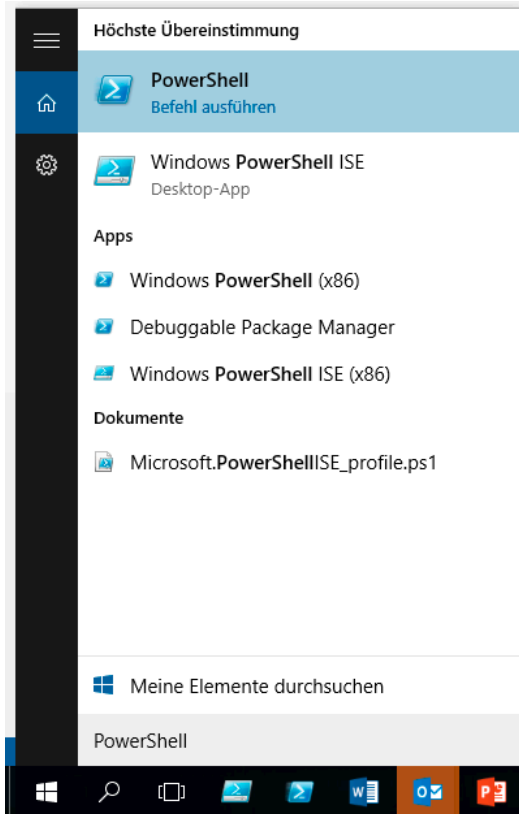


Bild 1.2 PowerShell-Einträge im Windows-10-Startmenü



TIPP: Unter Windows 8.x empfiehlt sich der Einsatz der Erweiterung <http://www.classicshell.net>, die das klassische Startmenü in Windows 8.x zurückbringt. Der Rückgriff auf ein Startmenü hat nicht nur mit Nostalgie zu tun, sondern auch ganz handfeste praktische Gründe: Der kachelbasierte Startbildschirm von Windows 8.x findet leider zum Suchbegriff „PowerShell“ weder die PowerShell ISE noch die 32-Bit-Variante der PowerShell.

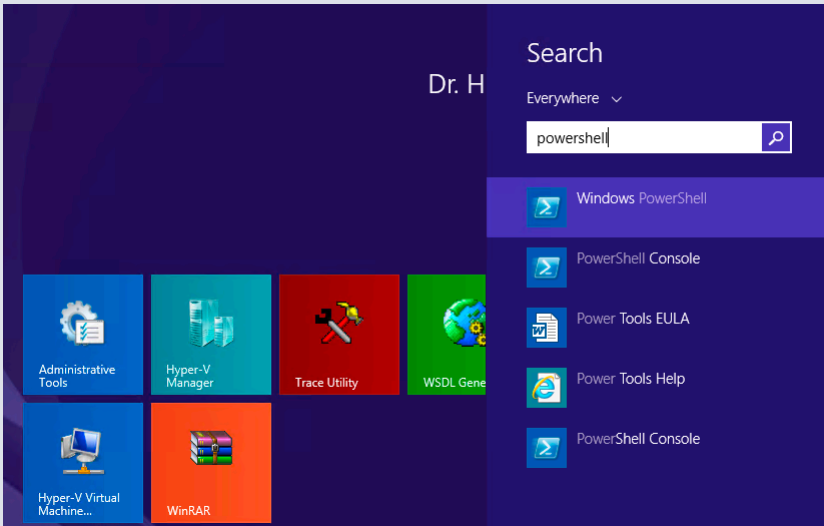


Bild 1.3 Versagen auf ganzer Linie: Der kachelbasierte Startbildschirm von Windows 8.x findet leider zum Suchbegriff „PowerShell“ weder die ISE noch die 32-Bit-Variante der PowerShell. In Windows 10 ist das Problem behoben.

Unter Windows 8.x geht es mit Classic Shell:

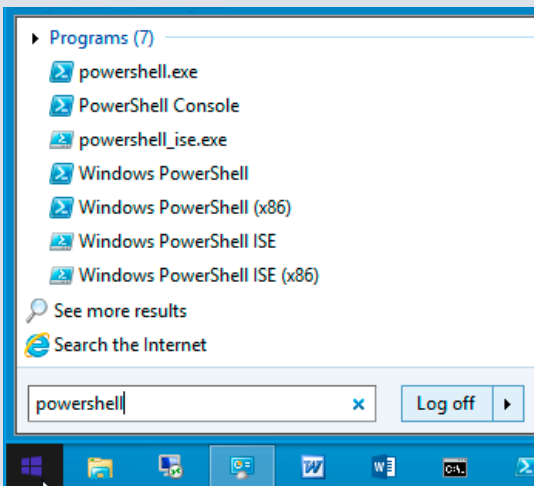


Bild 1.4 Die Classic Shell findet alle Einträge zur Windows PowerShell.

Ereignisprotokoll „PowerShell“

Durch die Installation der PowerShell wird in Windows auch ein neues Ereignisprotokoll „PowerShell“ angelegt, in dem die PowerShell wichtige Zustandsänderungen der PowerShell protokolliert.

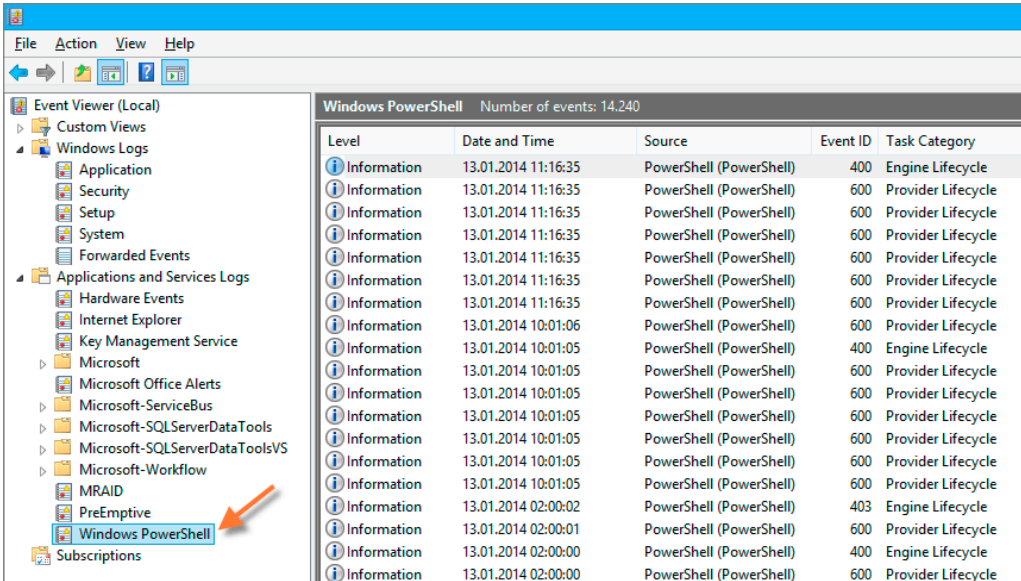


Bild 1.5 Ereignisprotokoll „Windows PowerShell“

Deinstallation

Falls man die PowerShell deinstallieren möchte, muss man dies in der Systemsteuerung unter „Programme und Funktionen/Installierte Updates“ tun und dort das „Microsoft Windows Management Framework“ deinstallieren.

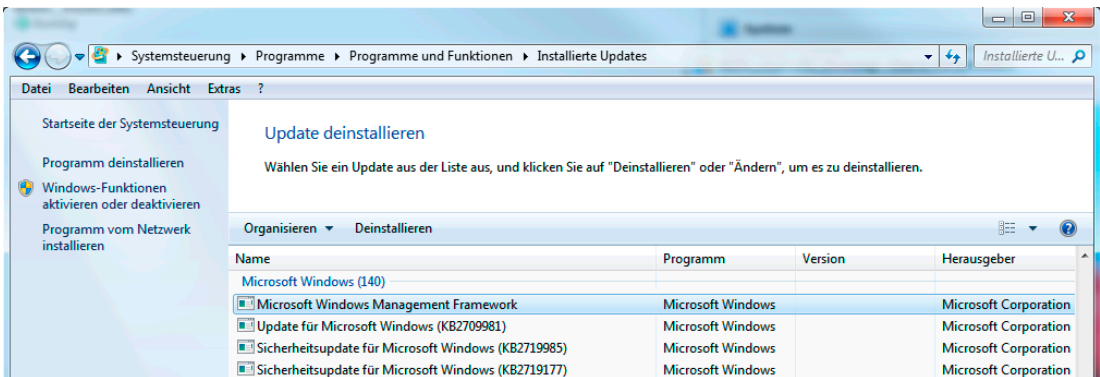


Bild 1.6 Deinstallation der PowerShell durch Deinstallation des WMF