



x-systems.press



Peter Monadjemi

PowerShell für die Windows- Administration

Ein kompakter und
praxisnaher Überblick

 Springer Vieweg



X-systems.press



Peter Monadjemi

PowerShell für die Windows- Administration

Ein kompakter und
praxisnaher Überblick

 Springer Vieweg

X.systems.press

Weitere Bände in dieser Reihe
<http://www.springer.com/series/5189>

X.systems.press ist eine praxisorientierte Reihe zur Entwicklung und Administration von Betriebssystemen, Netzwerken und Datenbanken.

Peter Monadjemi

PowerShell für die Windows-Administration

Ein kompakter und praxisnaher Überblick

Peter Monadjemi
ActiveTraining
Esslingen
Deutschland

ISSN 1611-8618

ISBN 978-3-658-02963-0

ISBN 978-3-658-02964-7 (eBook)

DOI 10.1007/978-3-658-02964-7

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

Springer Vieweg

© Springer Fachmedien Wiesbaden 2014

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsgesetz zugelassen ist, bedarf der vorherigen Zustimmung des Verlags. Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt 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.

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier

Springer Vieweg ist eine Marke von Springer DE. Springer DE ist Teil der Fachverlagsgruppe Springer Science+Business Media
www.springer-vieweg.de

Vorwort

Die Windows PowerShell ist Microsofts Antwort auf den Umstand, dass die Befehlszeilenshell Cmd.exe seit ihrer Einführung im Jahre 1993 mit der ersten Version von Windows 3.1 nur minimal überarbeitet wurde, und damit was Komfort und Befehlsumfang betrifft, nicht mehr den Anforderungen im administrativen Alltag gerecht werden kann. Doch die Windows PowerShell ist (natürlich) sehr viel mehr als der moderne Nachfolger von Cmd.exe, der auf dem .NET Framework basiert, und dessen Befehle Objekte anstelle von Text über die Pipeline weiterreichen. Die PowerShell ist ein Automatisierungswerkzeug, das sich in einer Vielzahl von Szenarien im modernen Windows Server-Umfeld einsetzen lässt. Umfasst ihr Netzwerk nur wenige Server und eine überschaubare Anzahl an Clients, lassen sich alle Einstellungen komfortabel in der GUI erledigen. Das Bild ändert sich schlagartig, wenn auf einmal mehrere Hundert oder mehrere Tausend (virtuelle) Server in einem Rechenzentrum konfiguriert werden müssen. In diesen Situationen präsentiert sich die PowerShell als ein Werkzeug, das schnell unverzichtbar werden dürfte. Es muss aber nicht unbedingt die große IT mit ihren Rechenzentren, die zunehmend in die Cloud verlagert werden, sein, auch in den kleinen Dingen spielt die PowerShell ihre Stärken aus. Möchten Sie ein Dutzend Pdf- oder Html-Dateien, die über einen Webbrowser heruntergeladen wurden, „entsperren“, geht dies noch per Maus. Sobald aber mehrere Hundert oder gar Tausende von Dateien im Spiel sind, geht es nicht mehr ohne ein Automatisierungswerkzeug. Auch für diese eher profanen Tätigkeiten ist die PowerShell ideal geeignet. Stichwort Cloud-Computing. Wer in diesem Bereich als Administrator bereits unterwegs ist, erhält mit der PowerShell ein Werkzeug, mit dem sich nahezu alle Dienstleistungen, die Microsoft über ihre Azure-Plattform anbietet, konfigurieren lassen. Das Bereitstellen vorkonfigurierter VMs („Virtuelle Maschinen“) wird damit sehr einfach möglich. Ob dabei 10 oder 10.000 VMs bereitgestellt werden, spielt keine Rolle. Übrigens ist der Zugriff auf die PowerShell nicht mehr auf Windows Clients beschränkt. Wurde auf einem Windows Server 2012 das Feature PowerShell Web Access hinzugefügt und konfiguriert, kann eine PowerShell-Session im Browser von jedem Endgerät aus gestartet werden.

In diesem Buch lernen Sie sowohl die Grundlagen der PowerShell als auch die verschiedenen Themengebiete, die im modernen „Admin-Alltag“ eine Rolle spielen, praxisnah und leicht verständlich beschrieben kennen. Mit der Desired State Configuration (DSC) wird in diesem Buch ein Thema vorgestellt, das in Zukunft im Microsoft-Umfeld

für die Server-Konfiguration eine sehr wichtige Rolle spielen wird. DSC soll nach den Plänen von Microsoft der künftige Standard für die Server-Konfiguration in einem Rechenzentrum sein.

Ich wünsche Ihnen viel Freude beim Lesen und Lernen,

Peter Monadjemi

Inhaltsverzeichnis

1	Einleitung	1
1.1	Und was geht es in diesem Buch?	2
1.2	Welche Version wird verwendet?	3
1.3	Ist die PowerShell schwer zu lernen?	3
1.4	Wenn ein Thema in diesem Buch nicht vorkommt	4
1.5	Ein Wort zu den Schreibkonventionen für Beispiele	4
1.6	Wo gibt es die Beispielskripte?	5
1.7	Kontakt zum Autor	5
1.8	Danksagungen	5
2	Ein erster Überblick	7
2.1	Wie alles anfing – ein kurzer Blick zurück	8
2.2	Die Stärken der PowerShell	9
2.2.1	Beispiele für die Konsistenz der PowerShell-Befehle	10
2.3	Ein technischer Überblick über die PowerShell	11
2.3.1	Die Rolle des .NET Frameworks (.NET-Laufzeit)	12
2.3.2	PowerShell als Interpreter	13
2.3.3	PowerShell für andere Plattformen?	13
2.3.4	Die Rolle des PowerShell-Hosts	14
2.4	Download und Installation	14
2.5	Zielgruppe und Anwendungsbereiche	17
2.6	PowerShell lernen	18
2.6.1	RT(F)M	19
2.6.2	Die Neuerungen der Version 4.0	19
2.7	Zusammenfassung	20
3	Die ersten Schritte mit der PowerShell	21
3.1	Die PowerShell starten	21
3.1.1	Schalter für Powershell.exe	22
3.1.2	Befehlsübergabe im Base64-Format	22
3.2	Einrichten des Konsolenfensters	23

3.2.1	Einstellen von Vorder- und Hintergrundfarbe	23
3.2.2	Copy und Paste in der Befehlszeile	24
3.2.3	Auswahl von Consolas als Schriftart	24
3.2.4	Einstellen des Fensterpuffers	25
3.2.5	Festlegen der Befehlspuffergröße	25
3.2.6	Per Tastatur scrollen	26
3.2.7	Die Rolle der Maus im Konsolenfenster	26
3.2.8	Alternativen zum Konsolenfenster	26
3.2.9	Konsoleneinstellungen per PowerShell-Skript vornehmen	27
3.3	Die PowerShell ISE als Alternative zur Konsole	28
3.4	Regeln für die Eingabe von Befehlen	29
3.4.1	Berechnungen in der Befehlszeile	29
3.4.2	Formatierte Ausgabe per f-Operator	30
3.5	Die Rolle des Prompts	31
3.6	Die ersten Schritte mit der Konsole	33
3.7	Befehlsauswahl per Show-Command	35
3.8	Der Umgang mit der Hilfe	36
3.8.1	Die PowerShell-Hilfe offline speichern	37
3.9	Die Rolle der Ausführungsrichtlinie	37
3.9.1	Auflisten aller Richtlinienebenen	38
3.9.2	Die Rolle der Einstellung RemoteSigned	38
3.10	PowerShell-Skripte aus dem Internet laden	39
3.11	Die Rolle der Profilskripte	40
3.11.1	Ein Profilskript laden	41
3.12	Wichtige Begriffe	42
3.12.1	Cmdlet	43
3.12.2	Function	43
3.12.3	Parameter	43
3.12.4	Alias	43
3.12.5	Command	43
3.12.6	Pipeline	44
3.12.7	Objekt	44
3.13	Zusammenfassung	44
4	PowerShell-Commands	45
4.1	Umgang mit Cmdlets	45
4.1.1	Überblick über die zur Auswahl stehenden Cmdlets	46
4.1.2	Die vier wichtigsten Cmdlets	47
4.2	Cmdlets und ihre Parameter	47
4.2.1	Die Common Parameters	48
4.2.2	Positionsparameter	49
4.2.3	Benannte Parameter	50
4.2.4	Parametersets	50

4.3	Hilfe zu Commands	51
4.3.1	Wenn sich die Hilfe-Ausgabe nicht beenden lässt	53
4.4	Cmdlets und Module	53
4.5	Alias	54
4.6	Zusammenfassung	55
5	Die Objektpipeline	57
5.1	Objekte	57
5.2	Typen	59
5.2.1	Wo kommen die Typen her?	59
5.2.2	Mehr über Typen erfahren	60
5.2.3	Umgang mit Typen	60
5.3	Members	61
5.3.1	Membertypen	61
5.3.2	Aufruf von Methoden-Members	63
5.4	Das Get-Member-Cmdlet	64
5.4.1	Abfragen statischer Members	66
5.5	Die Objektpipeline	67
5.5.1	Die Pipeline sichtbar machen	68
5.6	Pipeline-Operationen	69
5.6.1	Sortieren mit Sort-Object (Sort)	69
5.6.2	Filtern mit Where-Object (Where bzw. ?)	70
5.6.3	Verknüpfte Bedingungen	70
5.6.4	Objekte und Eigenschaften auswählen mit Select-Object (Select)	71
5.6.5	Gruppieren mit Group-Object (Group)	73
5.6.6	Wiederholungen mit ForEach-Object (%)	74
5.6.7	Vereinfachte Syntax ab der Version 3.0	76
5.6.8	ForEach-Object für etwas Fortgeschrittene	77
5.6.9	ForEach-Object abbrechen	78
5.6.10	„Messungen“ mit Measure-Object	78
5.6.11	Aufsplitten der Pipeline mit Tee-Object	79
5.6.12	Der (neue) PipelineVariable-Parameter	79
5.7	Die Objektpipeline ausgeben	80
5.7.1	Tabellarische Ausgaben per Format-Table (Alias Ft)	81
5.7.2	Eigene Spalten definieren	82
5.7.3	Listenausgabe per Format-List (Alias Fl)	84
5.7.4	Eine stark reduzierte Ausgabe per Format-Wide	84
5.7.5	Individuelle Formatierung per Format-Custom-Cmdlet (Alias Fc)	85
5.7.6	Direkte Ausgaben in den Host	85
5.8	Die Objektpipeline in andere Formate konvertieren	87
5.8.1	Export in das CSV-Format	88

5.8.2	Export in das HTML-Format	88
5.8.3	Export in das XML-Format	89
5.8.4	Export in das JSON-Format	89
5.9	Die Objektpipeline exportieren	90
5.10	Die Schattenseite der Pipeline	91
5.11	Objekte in 10 min	91
5.12	Zusammenfassung	95
6	Provider, Laufwerke und die Registry	97
6.1	Provider	97
6.1.1	Provider hinzufügen	99
6.1.2	Provider und ihre Fähigkeiten	100
6.2	PowerShell-Laufwerke	100
6.3	Die Rolle des Pfades	101
6.3.1	Der LiteralPath-Parameter	102
6.3.2	Platzhalter für Pfade	102
6.4	Die Item-Cmdlets	103
6.5	Die ItemProperty-Cmdlets	104
6.6	Verzeichnisse (Container) durchlaufen per Get-ChildItem-Cmdlet	105
6.6.1	Große Dateien finden	107
6.6.2	Dateien mit einem „Zone.Identifier“-Eintrag finden	107
6.7	Verzeichnisnavigation	108
6.7.1	Das aktuelle Verzeichnis merken – die Cmdlets Push-Location und Pop-Location	108
6.7.2	UNC-Pfade	109
6.8	Cmdlets für den Umgang mit Pfaden	109
6.9	Kopieren von Dateien und Verzeichnissen	110
6.9.1	Allgemeine „Elemente“ kopieren	110
6.9.2	Die Anzahl der kopierten Dateien erhalten	111
6.9.3	Dateien kopieren per Robocopy	112
6.9.4	Kopieren von Programmdateien von der Produktversion abhängig machen – die VersionInfo-Eigenschaft	112
6.10	Datei- und Verzeichnissicherheit	115
6.10.1	Berechtigungen für ein Verzeichnis abfragen	115
6.10.2	Gezielt Zugriffsregeln für ein Benutzerkonto abfragen	116
6.10.3	Berechtigungen für ein Verzeichnis per Set-ACL setzen	119
6.10.4	Das PowerShellAccessControl-Modul als Alternative zu Get-ACL und Set-ACL	121
6.11	Weitere Themen für den Umgang mit PowerShell-Laufwerken	123
6.11.1	Neue Laufwerke anlegen	123
6.11.2	Umgang mit Umgebungsvariablen	124
6.11.3	Den Wert der Path-Variablen ergänzen	125

6.11.4	Dauerhafte Umgebungsvariablen anlegen	126
6.12	Der Zugriff auf die Registry	126
6.12.1	Registry-Zugriffe per Item-Cmdlets	126
6.12.2	Auflisten der Run-Einträge	129
6.13	Laufwerksabfragen per WMI	131
6.14	Virtuelle Laufwerke und ISO-Dateien	131
6.15	Transaktionen	132
6.16	Zusammenfassung	133
7	Systemabfragen mit der PowerShell	135
7.1	Prozesse abfragen per Get-Process	135
7.1.1	Die Rolle der PropertySets	136
7.1.2	Den Besitzer eines Prozesses anzeigen	137
7.1.3	Prozesse und Module	137
7.1.4	Prozesse und Fenster	137
7.1.5	Datei- und Programmversionsnummern	137
7.1.6	Nicht reagierende Prozesse „abschießen“	138
7.2	Dienste abfragen per Get-Service	139
7.2.1	Den Startmodus eines Dienstes abfragen	140
7.3	Ereignisprotokolle abfragen per Get-EventLog und Get-WinEvent	140
7.4	Updates abfragen per Get-Hotfix	141
7.5	Weitere Abfragen per WMI	142
7.5.1	Konsolenprogramme als Alternative	142
7.6	Ausgaben in einem Fenster per Out-GridView	142
7.7	HTML-Reports	144
7.8	Zusammenfassung	145
8	Die PowerShell ISE	147
8.1	Ein erster Überblick	147
8.2	PowerShell-Skripte in der ISE ausführen	149
8.2.1	Unterschiede zwischen ISE und PowerShell-Konsole	149
8.2.2	STA oder MTA?	150
8.2.3	Interaktive Konsolenprogramme in der ISE-Konsole ausführen	151
8.3	Der PowerShell-Debugger	151
8.3.1	Wozu ist ein Debugger gut?	151
8.3.2	Der PowerShell-Debugger im Überblick	152
8.3.3	Die Rolle der Haltepunkte	152
8.3.4	Setzen von Haltepunkten	152
8.3.5	Die Breakpoint-Cmdlets im Überblick	153
8.3.6	Haltepunkte für Variablen und Commands	154
8.3.7	Haltepunkte über eine Aktion steuern	154
8.3.8	Haltepunkte entfernen	155

8.3.9	Automatische Variablen beim Debuggen	155
8.3.10	Remote-Debugging	155
8.4	Erweiterungen für die ISE (Add-Ons)	156
8.4.1	Ein Beispiel für ein ISE-Add-On	156
8.4.2	ISE-Ausschnitte	157
8.5	Alternativen zur PowerShell ISE	157
8.5.1	Aus Ps1-Dateien Exe-Dateien machen	159
8.5.2	PowerGUI Konsole	160
8.6	Zusammenfassung	161
9	PowerShell-Skripte	163
9.1	Skripte als eine moderne Variante der Stapeldateien	163
9.2	PowerShell-Skripte	164
9.2.1	Die Ausführungsrichtlinie	164
9.2.2	PowerShell-Skripte ausführen	165
9.2.3	Powershell.exe aus einem Skript heraus starten	165
9.2.4	Der dotsourced-Aufruf eines Skripts	165
9.2.5	Skripte über eine Verknüpfung aufrufen	166
9.2.6	PowerShell-Skripte von einer Netzwerkfreigabe ausführen	166
9.3	Die PowerShell-Skriptbefehle im Überblick	167
9.3.1	Kommentare	168
9.3.2	Variablen	168
9.3.3	Automatische Variablen	168
9.3.4	Variablen und ihre Datentypen	169
9.3.5	Mehr zu den Datentypen	171
9.3.6	Variablenverwendung erzwingen	171
9.3.7	Der Variablen-Scope	172
9.3.8	Zuweisungen	173
9.3.9	Umgang mit Datum und Zeit	173
9.3.10	Formatierte Ausgabe von DateTime-Werten	174
9.3.11	Die Rolle der Scriptblöcke	175
9.3.12	Einfache Entscheidungen mit dem if-Befehl	176
9.3.13	Entscheidungen mit dem else-Befehl	176
9.3.14	Entscheidungen mit dem elseif-Befehl	177
9.3.15	Mehrfachentscheidungen mit dem switch-Befehl	177
9.3.16	Wiederholungen	179
9.3.17	Der for-Befehl	179
9.3.18	Die Befehle do und while	180
9.3.19	Der while-Befehl	181
9.3.20	Der foreach-Befehl	181

9.3.21	Der continue-Befehl sorgt für einen Abbruch des Schleifendurchlaufs	182
9.3.22	Der break-Befehl sorgt für den Abbruch einer Schleife	183
9.3.23	Fehlerbehandlung mit dem trap-Befehl	183
9.4	Ein- und Ausgaben in einem Skript	184
9.4.1	Skripteingaben über Parameter	184
9.4.2	Skripteingaben über die Pipeline	186
9.4.3	Eingaben über das Read-Host-Cmdlet	187
9.4.4	Ausgaben über eine Meldungsbox	187
9.4.5	Eine Meldungsbox mit Optionen	188
9.5	Arrays (einfache Listen)	189
9.5.1	Arrays deklarieren	190
9.5.2	Die Länge eines Arrays	191
9.5.3	Mehrdimensionale Arrays	191
9.5.4	Ein Array mit einem Inhalt anlegen	192
9.6	Hashtables (Listen mit direktem Zugriff)	192
9.6.1	Eigenschaften für ein Objekt per Select-Object hinzufügen	193
9.6.2	Hashtables für etwas Fortgeschrittene	194
9.7	PowerShell-Skripte mit Parametern	196
9.8	Informationen über ein Skript abfragen	196
9.9	Zusammenfassung	197
10	Functions	199
10.1	Functions definieren	199
10.2	Die Rückgabewerte einer Function	200
10.2.1	Der return-Befehl	200
10.3	Functions mit Parametern	201
10.3.1	Die Variable \$Args	201
10.3.2	Der param-Befehl	202
10.3.3	Runde Klammern, die auf den Function-Namen folgen	204
10.3.4	Fehler beim Function-Aufruf mit Argumenten	204
10.4	Functions, die die Pipeline abarbeiten	204
10.5	Spezielle Themen beim Umgang mit Functions	206
10.5.1	Parameterübergabe als Referenz	206
10.5.2	Verschachtelte Functions	207
10.5.3	Die Rolle der Parameterattribute	207
10.5.4	Das Parameter-Attribut	208
10.5.5	Validierungsattribute	210
10.5.6	Validieren auf einen Zahlenbereich	210
10.5.7	Validieren auf eine Menge	210
10.5.8	Validierung mit Hilfe eines regulären Ausdrucks	211
10.5.9	Vorteile der Parametervalidierung	211