
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 anfang – 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

10.5.10	Feststellen, ob ein Parameter übergeben wird	211
10.5.11	Das CmdletBinding-Attribut	212
10.5.12	Kommentarschlüsselwörter für die Hilfe	213
10.5.13	Der Scope bei Function-Variablen	214
10.5.14	Der Scope bei verschachtelten Functions	216
10.5.15	Die Sichtbarkeit von Variablen	217
10.5.16	Fehlerbehandlung in Functions	218
10.5.17	Functions einer Ps1-Datei als Methoden-Members eines Objekts ansprechen	219
10.6	Zusammenfassung	219
11	WMI	221
11.1	WMI in 10 min	221
11.2	Ws-Man als Alternative zu DCOM	224
11.3	Die CIM-Cmdlets im Überblick	224
11.4	Die CIM-Cmdlets in der Praxis	225
11.4.1	Die WQL-Abfragesyntax	226
11.4.2	WQL-Abfragen in der Praxis	227
11.4.3	Die Namen von CIM-Klassen herausbekommen	227
11.4.4	Beispiele für Konfigurationsdatenabfragen	228
11.4.5	Umgang mit Drucker und Druckjobs	228
11.4.6	Installierte Anwendungen auflisten	229
11.4.7	Eine Anwendung per WMI deinstallieren	229
11.4.8	WMI-Abfragen mit PowerShell-Abfragen kombinieren	230
11.5	WMI-Abfragen im Netzwerk	232
11.6	WMI-Spezialitäten	233
11.6.1	Direkte Aufrufe über Ws-Management	233
11.6.2	Einstellungen der Ws-Man-Konfiguration	236
11.6.3	Assoziationsabfragen	237
11.6.4	WMI-Ereignisse	238
11.6.5	Abfragen von Leistungsindikatoren	240
11.7	Zusammenfassung	241
12	Fehlerbehandlung	243
12.1	Ein erster Überblick	243
12.2	Nicht terminierende und terminierende Fehler	244
12.2.1	Der ErrorAction-Parameter	245
12.2.2	Fehlermeldungen umleiten	246
12.3	Fehlervariablen	247
12.3.1	Fehlernummern und das ErrorRecord-Objekt	247
12.3.2	Die Rolle der Exception	249
12.3.3	Die Zeilennummer zu einem Fehler abfragen	249

12.4	Der Return-Code von externen Anwendungen	250
12.5	Fehler in Skripten abfangen	250
12.5.1	Terminating Errors abfangen mit dem trap-Befehl	251
12.5.2	Terminating Errors abfangen mit try, catch und finally	253
12.5.3	Auf bestimmte Fehlertypen reagieren	254
12.6	Fehler weiterleiten und Skripte beenden per throw-Befehl	255
12.6.1	Per throw-Befehl Pflichtparameter deklarieren	256
12.7	Fehler protokollieren	257
12.7.1	Anlegen einer Quelle	258
12.7.2	Ereignisprotokolleinträge schreiben	258
12.8	Empfehlungen für bessere Skripte	260
12.9	Zusammenfassung	260
13	PowerShell-Remoting	261
13.1	PowerShell-Remoting im Überblick	261
13.2	Ein erstes Beispiel	262
13.3	Voraussetzungen für PowerShell-Remoting	263
13.3.1	Szenario: Client und Server befinden sich innerhalb derselben Domäne	263
13.3.2	Szenario: Der Client ist nicht Teil der Domäne	264
13.3.3	Szenario: Client und Server sind nicht Teil der Domäne	264
13.3.4	Wer darf PowerShell-Remoting benutzen?	264
13.3.5	Server-Computer auf dem Client zur Liste der TrustedHosts hinzufügen	265
13.4	Cmdlets für PowerShell-Remoting	266
13.5	PowerShell-Remoting auf dem Server einrichten	267
13.5.1	PowerShell-Remoting per Gruppenrichtlinie einrichten	269
13.6	Das Invoke-Command-Cmdlet	270
13.6.1	Herstellen einer Remoting-Session über das explizite Anlegen einer Session	271
13.6.2	Herstellen einer Remoting-Session über den ComputerName-Parameter	273
13.6.3	Erneutes Verbinden mit einer Remoting-Session	273
13.6.4	Weitere Beispiele für Invoke-Command	274
13.6.5	Parameterübergabe an eine Remoting-Session	276
13.7	Die Rolle der Sessions	277
13.7.1	Interaktives Betreten einer Session	279
13.7.2	Die Rolle der Sessionkonfiguration	279
13.7.3	Anlegen einer Sessionkonfiguration	280
13.7.4	Anlegen einer eingeschränkten Sessionkonfiguration	281
13.8	Weitere Spezialthemen	282
13.8.1	Die Rolle der Sessionoptionen	282

13.8.2	Einbeziehen eines Proxy	283
13.8.3	Die Ws-Man-Konfiguration im Überblick	285
13.8.4	Einrichten einer HTTPS-Verbindung	286
13.9	Wenn es Probleme gibt	287
13.9.1	„Access denied“, was nun?	289
13.9.2	Das WinRM-Ereignisprotokoll	289
13.9.3	Eine Ws-Man-Verbindung belauschen	289
13.9.4	SSH als Alternative	290
13.10	Der Umgang mit Credentials	291
13.10.1	Zusammensetzen eines PSCredential-Objekts – die Cmdlets ConvertTo-SecureString und ConvertFrom-SecureString	292
13.11	Zusammenfassung	293
14	Zugriff auf das Active Directory	295
14.1	Eingebaute Zugriffsmöglichkeiten	295
14.1.1	Zugriffe über [ADSI] und [ADSISeacher]	296
14.1.2	Anmelden an eine Domäne mit [ADSI]	299
14.1.3	Den RootDSE ansprechen	299
14.1.4	Suchen im Active Directory mit [ADSISeacher]	300
14.2	Das Active Directory-Modul	301
14.2.1	Voraussetzungen bei Windows Server 2003/2008	301
14.2.2	Das Active Directory-Modul bei Windows 7/8/8.1	301
14.2.3	Das Active Directory-Modul im Überblick	301
14.3	Active Directory-Abfragen	302
14.3.1	Die PowerShell-Filtersyntax	304
14.4	Spezielle Abfragen mit Benutzerkonten	305
14.5	Computerkonten auflisten	306
14.6	Benutzerkonten anlegen und ändern	308
14.6.1	Benutzerkonten mit einer Vorlage anlegen	309
14.6.2	Benutzerkonten nach dem Anlegen aktivieren	309
14.6.3	Attribute von Benutzerkonten ändern	310
14.6.4	Benutzerkonten entfernen	311
14.7	Umgang mit Gruppen	312
14.8	Spezialitäten beim Active Directory-Modul	313
14.8.1	Das AD-Laufwerk	313
14.8.2	Aktivieren des Papierkorbs	314
14.8.3	Suchen nach anderen AD-Objekten	315
14.8.4	Cmdlets für die Domänenverwaltung	315
14.8.5	Das Active Directory Verwaltungszentrum als Alternative zur PowerShell	315
14.9	Zusammenfassung	316

15 Module und Snapins	317
15.1 Die Idee der Module	317
15.1.1 Implizites Laden von Modulen	318
15.2 Die Anatomie eines Moduls	319
15.3 Ein Überblick über die vorhandenen Module	319
15.3.1 Die Umgebungsvariable PSModulePath	320
15.3.2 Cmdlets für den Umgang mit Modulen	321
15.4 Modultypen	321
15.4.1 Skriptmodule	321
15.4.2 Ein Beispiel für ein Skriptmodul für Zip-Funktionen	322
15.4.3 Manifestmodule	324
15.4.4 Die Umsetzung einer Moduldatei Schritt für Schritt	325
15.4.5 Binäre Module	327
15.4.6 Dynamische Module	328
15.5 Module aus dem Internet laden	329
15.5.1 Module direkt laden	330
15.6 Snapins	332
15.7 Zusammenfassung	332
16 Windows Server-Module und Web Access	333
16.1 Ein erster Überblick	333
16.1.1 Windows Server-Module unter älteren Windows-Versionen benutzen	334
16.2 Das Storage-Modul als Nachfolger von Diskpart	335
16.2.1 Initialisieren einer Vhd-Datei	336
16.3 Geplante Aufgaben	337
16.3.1 Das DNS-Client-Modul	340
16.3.2 DNS-Namensauflösung	341
16.4 Umgang mit Freigaben	342
16.5 Netzwerkkonfiguration	343
16.6 Features hinzufügen und entfernen	344
16.7 PowerShell Web Access	344
16.8 Zusammenfassung	348
17 Umgang mit Text	349
17.1 Zeichenketten (Strings)	349
17.2 String-Verarbeitung	350
17.3 Here-Strings	351
17.3.1 Zeilenumbrüche und andere Spezialitäten	351
17.4 Textdateien lesen und schreiben	351
17.4.1 Binärdateien lesen	352
17.4.2 Streams als Alternative	353

17.5	XML-Daten lesen und schreiben	354
17.5.1	XML-Daten auswerten per Select-Xml	359
17.6	Reguläre Ausdrücke	360
17.6.1	Reguläre Ausdrücke in 10 min	360
17.6.2	Reguläre Ausdrücke bei der PowerShell	365
17.6.3	Operatoren für reguläre Ausdrücke	365
17.6.4	Der Replace-Operator	366
17.6.5	Das Select-String-Cmdlet	367
17.6.6	Der [Regex]-Type Accelerator	368
17.7	Aus Texten Objekte machen	368
17.8	Zusammenfassung	370
18	Die PowerShell für etwas Fortgeschrittene	371
18.1	Erweiterungen für die PowerShell	371
18.2	Verbesserter Umgang mit Objekten innerhalb eines Array (oder einer Collection)	372
18.3	Ein Blick hinter die Kulissen mit den Metadaten	373
18.3.1	Informationen über die Parameter eines Cmdlets abfragen	374
18.4	Die Metadaten eines Objekts Teil 1 – das universelle RuntimeType-Objekt	377
18.5	Die Metadaten eines Objekts Teil 2 – die PsObject-Eigenschaft	379
18.5.1	Die Objektstruktur sichtbar machen (Teil 1)	380
18.5.2	Die Objektstruktur sichtbar machen (Teil 2)	380
18.6	Externe Prozesse starten	381
18.6.1	Die Rolle der Path-Umgebungsvariablen	381
18.6.2	Prozesse starten mit dem Start-Process-Cmdlet	382
18.6.3	Ausgaben für Konsolenprogramme umleiten	383
18.7	Webservice-Funktionen aufrufen	383
18.8	.NET-Laufzeitfunktionen aufrufen	384
18.8.1	Instanzen-Member einer Klasse aufrufen	385
18.8.2	Statische Member einer Klasse aufrufen	386
18.8.3	.NET-Assemblies laden	387
18.8.4	Zip-Funktionen benutzen	387
18.9	COM-Komponenten ansprechen	388
18.9.1	Auf die erweiterten Eigenschaften einer Datei zugreifen	388
18.9.2	Excel fernsteuern	390
18.10	Jobs	391
18.10.1	Auf die Beendigung eines Jobs reagieren	393
18.10.2	Job ist fertig – Benachrichtigung per Event	393
18.10.3	Ereignisse wieder deregistrieren	395
18.11	Ereignisse	395
18.11.1	Regelmäßige Wiederholungen über ein Timer-Ereignis	396

18.11.2	Auf das Beenden des PowerShell-Hosts reagieren	397
18.11.3	Die PowerShell-Sitzung per Tastenshortcut beenden	398
18.11.4	Event-Parameter	398
18.12	Internationale Skripts	399
18.13	Neue Objekte mit den New-Object-Cmdlet anlegen	401
18.14	Umgang mit Type Acceleratoren	401
18.15	Das PowerShell-Typensystem erweitern	403
18.16	Umgang mit Privilegien	404
18.17	Weitere Themen	404
18.17.1	Parameterübergabe per Splat-Operator	404
18.17.2	Aufsplitten von Eigenschaften	405
18.17.3	Members über Variablen ansprechen	406
18.17.4	Eine Invoke-Method-Function	407
18.17.5	E-Mails versenden	408
18.18	Skripte signieren	409
18.18.1	PowerShell und OpenSSL-Zertifikate	413
18.19	Zusammenfassung	414
19	Workflows	415
19.1	Ein erster Überblick	415
19.2	Ein erstes Beispiel	416
19.2.1	Workflow mit Parametern	417
19.2.2	Die Rückgabewerte eines Workflows festlegen	417
19.2.3	Eingabehilfen in der PowerShell ISE	418
19.2.4	Die allgemeinen Workflow-Parameter	418
19.2.5	Workflows im Netzwerk ausführen	419
19.2.6	Workflows debuggen	420
19.3	Workflow = Aktivitäten	420
19.3.1	Neue Befehlswörter und Variablen	422
19.4	Speziellere Themen	423
19.4.1	Unterschiede zwischen einem Workflow und einem PowerShell-Skript	423
19.4.2	„Problemfälle“ behandeln mit der InlineScript-Aktivität	423
19.4.3	Der Gültigkeitsbereich von Variablen – workflow und using ...	424
19.4.4	Verschachtelte Workflows	426
19.4.5	Parallele Aktivitäten	427
19.4.6	Workflows unterbrechen und fortsetzen	428
19.4.7	Den Zustand eines Workflows speichern („Persistenz“)	429
19.4.8	Workflows in Visual Studio erstellen	430
19.4.9	PowerShell-Workflows in Visual Studio anzeigen	432
19.5	Zusammenfassung	432

20	Desired State Configuration (DSC)	433
20.1	Desired State Configuration in 10 Min.	433
20.2	Ressourcen statt Cmdlets	435
20.3	Ein Hallo Welt-Beispiel für DSC	436
20.4	Zusammenfassung	437
21	Datenbankzugriffe mit der PowerShell	439
21.1	Ein erster Überblick	439
21.1.1	Erforderliche Komponenten	441
21.2	Läuft der SQL Server?	441
21.3	Das SQLPS-Modul	442
21.3.1	Das Invoke-SqlCmd-Cmdlet	443
21.3.2	Datenbanken anlegen	443
21.3.3	Datensätze einfügen	445
21.3.4	Datensätze aktualisieren	446
21.3.5	Datensätze löschen – Anzahl der Datensätze abfragen	446
21.4	Der PowerShell-Host Sqlps.exe	446
21.5	Das SQLServer-Laufwerk	447
21.5.1	Löschen von Tabellen	448
21.6	Datenbanken direkt ansprechen	449
21.6.1	Die Rolle der Verbindungszeichenfolge	449
21.6.2	SQL-Abfragen ausführen	449
21.6.3	Bilddaten einfügen	451
21.7	Andere DBMS-Typen ansprechen	452
21.7.1	Datensätze von einer MySQL-Datenbank abrufen	453
21.8	Datenbanken per SMO ansprechen	453
21.8.1	Die SMO kennenlernen	454
21.8.2	Anmelden an einen SQL Server	454
21.9	SQLPSX als Alternative zum direkten Zugriff per SMO und ADO.NET	455
21.10	Zusammenfassung	456
22	Windows Azure per PowerShell konfigurieren	457
22.1	Die Azure-Cmdlets	457
22.2	Voraussetzungen	459
22.3	Die ersten Schritte	460
22.4	Der Umgang mit virtuellen Maschinen	460
22.4.1	Anlegen einer neuen VM	460
22.4.2	Starten einer VM	462
22.4.3	Entfernen einer VM	462
22.4.4	Tipps für eine Fehlersuche	463
22.5	Herstellen einer PowerShell-Remoting-Session	463
22.6	Zusammenfassung	464

23 Benutzeroberflächen für PowerShell-Skripte	465
23.1 Qual der Wahl	465
23.2 Voraussetzungen	466
23.3 Die Rolle der Ereignisse	466
23.4 Die Umsetzung bei Windows Forms	467
23.5 Primal Forms Community Edition	470
23.6 WPF als Alternative zu Windows Forms	478
23.6.1 ShowUI – WPF ohne XAML	482
23.6.2 Ein einfaches Beispiel zur Einstimmung	482
23.6.3 Eine Dialogbox für die Eingabe von Benutzerkonten	483
23.6.4 Eine Benutzeroberfläche für Cmdlets	483
23.7 Zusammenfassung	485
24 Spaß mit der PowerShell	487
24.1 Töne und Melodien	487
24.1.1 PowerShell-Musik	488
24.1.2 Wave-Dateien abspielen	488
24.1.3 Systemsounds abspielen	489
24.1.4 Die sprechende PowerShell	490
24.1.5 Die singende PowerShell	490
24.2 Banner-Ausgaben	491
24.3 Wo ist die ISS?	491
24.4 Urlaubsfotos sortieren	492
24.5 Ein Spielhallen-Klassiker	494
24.6 Zusammenfassung	495
25 Die PowerShell für Entwickler	497
25.1 Erweiterungen für die PowerShell	497
25.2 Die Rolle der Assemblies	498
25.2.1 Aktuell geladene Assemblies anzeigen	498
25.2.2 Die Inhalte einer Assembly anzeigen	499
25.2.3 Ein Röntgenblick in eine Assembly	500
25.3 Assemblies laden mit dem Add-Type-Cmdlet	500
25.3.1 Externe Assemblies per Add-Type laden	500
25.4 C#-Programmcode per Add-Type-Cmdlet einbinden	502
25.4.1 Einbinden von Methoden-Definitionen	502
25.4.2 Einbinden einer vollständigen Typdefinition	504
25.5 Umgang mit Fenstern	506
25.5.1 Fenster per Tastendruck steuern	506
25.5.2 Das Icon des Konsolenfensters ändern	507
25.6 Cmdlets erstellen	508
25.6.1 Ein Hallo Welt-Cmdlet	508

25.7	PowerShell-Skripte in einem C#-Programm ausführen	510
25.8	Weitere Themen	511
25.8.1	VSVars32.bat einbinden	511
25.9	Generische Collections	513
25.9.1	Zugriff auf Visual Studio-Projektdateien	513
25.9.2	MsBuild-Aktionen durch PowerShell-Skripte erweitern	514
25.10	PowerShell-Unterstützung für Visual Studio	514
25.10.1	Der Paketmanager-PowerShell-Host	514
25.11	Zusammenfassung	516
Sachverzeichnis		517