

Inhaltsverzeichnis

Einführung	19
Wer braucht eigentlich PowerShell?	20
Moderne Lernkurve	20
Computer – ich/ich – Computer: PowerShell als Fremdsprache	21
Eine strategische Plattform	22
Plattformübergreifende Technik	25
Persönliche Entwicklung	25
Wie Sie dieses Buch nutzen	25
Noch mehr Unterstützung	26

Teil A	
Einführung	27

1 PowerShell kennenlernen	29
PowerShell starten	30
PowerShell einrichten	31
Moderne ISE-Konsole starten	35
Erste Schritte mit PowerShell	38
Wichtige Vorsichtsmaßnahmen	38
Befehle eingeben	39
Farbcodierungen in ISE	41
Rechnen mit PowerShell	43
Unvollständige und mehrzeilige Eingaben	45
Tippfehler vermeiden und Eingaben erleichtern	46
Autovervollständigung	47
Pfadnamen vervollständigen	47
Befehlszeilen erneut verwenden	47
Befehlsnamen autovervollständigen	48
Parameter-Autovervollständigung	48
Argument-Autovervollständigung	49
PowerShell-Hilfe aus dem Internet nachladen	50
Hilfe in anderer Sprache installieren	50
Hilfe ohne Internetzugang installieren	51
Klassische Konsole oder moderner ISE-Editor?	51
Einschränkungen in ISE	53
Einschränkungen der klassischen Konsole	54
Testen Sie Ihr Wissen!	54
Zusammenfassung	55

2 Cmdlets – die PowerShell-Befehle	59
Alles, was Sie über Cmdlets wissen müssen	61
Cmdlets für eine Aufgabe finden	61
Suche nach Tätigkeit oder Tätigkeitsbereich	62
Mit ISE nach Cmdlets suchen	67
Mit der Hilfe nach Cmdlets suchen	70
Mit Parametern Wünsche formulieren	74
Parameter wecken das volle Potenzial der Cmdlets	75
Drei universelle Parametertypen	81
Common Parameter – allgemeine Parameter für alle Cmdlets	86
Fehlermeldungen unterdrücken	88
Unvollständige Parameternamen	90
Parameter-Binding sichtbar machen	92
Cmdlets und virtuelle Laufwerke	93
Typische Dateisystemaufgaben meistern	94
Weitere »virtuelle Laufwerke« nutzen	95
Registrierungsdatenbank-Werte lesen und schreiben	96
Virtuelle Laufwerke und Provider verstehen	98
Neue virtuelle Laufwerke	99
Laufwerks-Cmdlets zusammengefasst	101
In PowerShell-Laufwerken navigieren	102
Neue Cmdlets aus Modulen nachladen	103
Alle Cmdlets entstammen Modulen	104
Neue Module automatisch nachladen	106
Auslaufmodell: Snap-Ins	109
Alias: Zweitname für Cmdlets	110
Aliase sind keine neuen Befehle	110
Befehlstypen ermitteln	111
Klassische Interpreterbefehle sind Cmdlets	112
Eigene Aliase anlegen	113
Testen Sie Ihr Wissen!	114
Zusammenfassung	124
3 Anwendungen und Konsolenbefehle	127
Programme starten	129
Optionen für den Programmstart festlegen	131
Nicht unterstützte Konsolenbefehle im ISE-Editor	132
Argumente an Anwendungen übergeben	134
Welche Argumente unterstützt eine Anwendung?	135
Argumentübergabe kann scheitern	138
Ergebnisse von Anwendungen weiterverarbeiten	139
Error Level auswerten	139
Fragen an Benutzer stellen mit choice.exe	141
Rückgabertext empfangen	143
Laufende Programme steuern oder beenden	153
Auf laufende Prozesse zugreifen	154
Einstellungen laufender Prozesse ändern	155
Auf laufende Prozesse warten	156
Prozesse vorzeitig abbrechen	156

Texteingaben an Konsolenbefehle senden	157
Testen Sie Ihr Wissen!	158
Zusammenfassung	161
4 Eigene und fremde Skripts einsetzen	165
PowerShell-Skripts verfassen	166
Skriptcode eingeben	167
Eingabehilfen spüren Tippfehler auf	168
Skript ausführen	168
Sicherheitseinstellungen und Ausführungsrichtlinien	169
Skripts als Befehlserweiterung	171
Skripts wie Befehle ausführen	171
Skripts mit Parametern ausstatten	172
Rückgabewerte des Skripts festlegen	174
Profilskripts – die Autostartskripts	174
Vier verschiedene Profilskripts – pro Host	175
Profilskript anlegen und öffnen	176
Typische Profilskript-Aufgaben durchführen	178
Persönliche PowerShell-Anpassungen übertragen	179
Skripts außerhalb von PowerShell starten	179
PowerShell-Startoptionen	181
Skript als geplante Aufgabe ausführen	183
Nicht-PowerShell-Skripts öffnen	185
Stapel- oder Stapeldateien	185
VBScript-Dateien ausführen	187
Zusammenfassung	189

Teil B

Fortgeschrittene Anwender

191

5 Die PowerShell-Pipeline	193
Aufbau der PowerShell-Pipeline	194
Prinzipieller Aufbau der Pipeline	196
Die sechs wichtigsten Pipeline-Befehle	196
Spalten auswählen mit Select-Object	197
Exkurs: Formatierungssystem von PowerShell	198
Unsichtbare neue Spalten anzeigen	199
Wahre Spaltennamen ermitteln	201
Platzhalterzeichen verwenden	202
Reine Spalteninhalte weitergeben	205
Die ersten oder letzten Ergebnisse auswählen	208
Ergebnisse filtern mit Where-Object	211
Where-Object: Nicht sehr effizient... ..	211
...aber enorm praktisch!	211
Leere Elemente aussortieren	213
Fortgeschrittene Syntax bietet noch mehr Möglichkeiten	213
Dateiinhalte filtern	216
IP-Adressen bestimmen	216

Ergebnisse sortieren mit Sort-Object	217
Cmdlet-Ergebnisse sortieren	218
Nach mehreren Spalten sortieren	219
Sortierung mit anderen Cmdlets kombinieren	219
Frei programmierbare Aktion mit ForEach-Object	221
Grundprinzip: Eine Schleife	221
Ergebnisse analysieren mit Group-Object	228
Häufigkeit und Gesamtgrößen: Measure-Object	229
Statistische Berechnungen	230
Ordnergrößen berechnen	231
Frei programmierbare Eigenschaften	232
Datentyp der Sortierung ändern	232
Gruppierung nach bestimmten Textteilen	233
Umwandlung von Byte in MB	234
Hashtabelle: Mehrere Werte übergeben	235
Mehrere Spalten in umgekehrter Sortierung	238
Mehrspaltige Anzeigen	238
Neue Einträge hinzufügen	239
Spaltenbreite, Bündigkeit und Gruppenüberschriften	239
Frei wählbare Gruppierungskriterien	241
Pipeline und Performance: Optimierungen	242
Flaschenhals Pipeline	243
Klassische Schleifen sind wesentlich schneller	243
Pipeline ist wesentlich reaktionsfreudiger	244
Weniger Speicherbedarf oder höhere Geschwindigkeit?	244
Pipeline vorzeitig abbrechen	245
Testen Sie Ihr Wissen!	246
Zusammenfassung	253
 6 Ergebnisse ausgeben und formatieren	 255
Ergebnisse als Text ausgeben	257
Optimierte Spaltenbreiten	258
Listendarstellung erzwingen	261
Mehrspaltige Anzeige	262
Textausgabe an andere Ziele leiten	265
Out-Printer: Ergebnisse zu Papier bringen	268
Out-WinWord: Ergebnisse direkt an Microsoft Word senden	273
Out-PDF: Mit Microsoft Word PDF-Reports erstellen	275
Ergebnisse als Objekte exportieren	277
Out-GridView: Objekte im Spreadsheet anzeigen	277
Export-Csv: Export an Microsoft Excel und andere Programme	279
Ergebnisse als XML serialisieren	282
HTML-Reports erstellen	287
Objekteigenschaften in HTML-Spalten umwandeln	287
HTML im Webbrowser anzeigen	288
HTML-Reports ansprechend und farbig gestalten	290
Mehrere HTML-Tabellen kombinieren	293
Testen Sie Ihr Wissen!	295
Zusammenfassung	298

7 Operatoren	301
Operatoren – kurzer Überblick	302
Wie Operatornamen aufgebaut sind	303
Unäre Operatoren	303
Der Format-Operator -f	304
Bündige Spalten herstellen	306
Zahlen formatieren	307
Datums- und Zeitangaben formatieren	309
Vergleichsoperatoren	312
Unterscheidung zwischen Groß- und Kleinschreibung	315
Unterschiedliche Datentypen vergleichen	315
Vergleiche umkehren	317
Vergleiche kombinieren	318
Vergleiche auf Arrays anwenden	319
Zuweisungsoperatoren	320
Textoperatoren und Textmanipulationen	322
Texte splitten und zusammenfügen	323
Informationen in Texten finden	331
Textstellen ersetzen	343
Muster mit regulären Ausdrücken beschreiben	347
Testen Sie Ihr Wissen!	350
Zusammenfassung	351
 8 Einfache Funktionen	 353
Alles Wichtige: Ein Überblick	355
Warum Funktionen?	355
Grundsätzliches Handwerkszeug	355
Eigene Funktionen herstellen	357
Funktionen mit Parametern ausstatten	358
Funktionen überall verfügbar machen	360
Hilfe – Bedienungsanleitung hinzufügen	363
Kurz zusammengefasst...	365
Parameter und Eingaben für Funktionen	366
Parameter definieren	366
Unbenannte Argumente	368
Benannte Parameter	369
Positionale Parameter	369
Switch-Parameter	372
Optionale Parameter	374
Zwingend erforderliche Parameter	374
Streng typisierte Parameter und Datentypen	377
Mehrere Argumente für einen Parameter	378
Rückgabewerte festlegen	379
Ein Rückgabewert oder mehrere?	380
Return-Anweisung	381
Unerwünschte Rückgabewerte unterdrücken	382
Auf die Rückgabewerte zugreifen	383

Statusmeldungen ausgeben	383
Festlegen, welche Meldungen sichtbar sind	385
Farben der PowerShell-Meldungen festlegen	386
Zusammenfassung	387

9 Fehlerhandling und Debugging	389
Syntaxfehler erkennen und beheben	390
Folgefehler und der Blick auf das Wesentliche	391
Formale Regeln missachtet	394
Typische Logikfehler aufspüren und entschärfen	396
Falsche Verwendung von Operatoren	396
Tippfehler ändern den Code	398
Nicht initialisierte Variable	398
Versehentliche Verwendung von Arrays	399
Fehlendes Verständnis für Objektreferenzen	400
Falsche Verwendung von Klammern	404
Falsche Zuordnung von Skriptblöcken	404
Mit dem Debugger Logikfehler nachvollziehen	406
Haltepunkte setzen und Code schrittweise ausführen	407
Codeausführung fortsetzen	409
Intelligente Haltepunkte setzen	409
Anhalten, wenn Cmdlets oder Funktionen aufgerufen werden	411
Anhalten, wenn Variablen bestimmte Werte enthalten	411
Debugging-Meldungen ausgeben	414
Tracing: Ausgeführte Anweisungen anzeigen lassen	415
Laufzeitfehler abfangen	417
Spezialfall: Terminierende Fehler	418
Fehler ereignisgesteuert mit try...catch abfangen	419
Globaler Fehlerhandler: Trap	423
ErrorRecords und Exceptions – Details zum Fehler	426
Exception-Typen kennenlernen	427
Spezifische Fehlerhandler einsetzen	428
Aussagekräftige Fehlermeldungen generieren	432
Eigene Fehler auslösen	437
Testen Sie Ihr Wissen!	440
Zusammenfassung	444

Teil C

Low-Level-Zugang	445
-------------------------------	------------

10 Mit Objekten arbeiten	447
Einführung: Objekte und Typen	448
Eigenschaften... ..	449
...und Methoden	451
Alle Eigenschaften und Methoden anzeigen	453
Hilfe zu Objekten und ihren Mitgliedern	455
Ergebnisse eines Befehls untersuchen	456

Achtung Falle: Objekte und Arrays	457
Primitive Datentypen	461
Objekteigenschaften lesen	462
Verschachtelte Objekteigenschaften	463
Objekteigenschaften ändern	465
Eigenschaften in verschachtelten Objekten	467
Änderungen erfordern Write-Back	470
Objektmethoden aufrufen	475
Eine Methode mit mehreren Signaturen	476
Testen Sie Ihr Wissen!	479
Zusammenfassung	484
11 Wie PowerShell Objekte erweitert	485
PowerShell-Objekte verstehen	486
Erweiterte PowerShell-Objekte	487
AliasProperty: Eigenschaften umbenennen	490
NoteProperty: Taggen von Objekten	491
ScriptProperty: »Berechnete« Eigenschaften	492
Lesbare Eigenschaften	492
Lesbare und schreibbare Eigenschaften	494
ScriptMethod und ParameterizedProperty	499
Membertypen für den internen Gebrauch	504
PropertySet: Gruppen von Eigenschaften	504
MemberSet: Wie soll PowerShell das Objekt behandeln?	505
CodeProperty: Statische Methoden aufrufen	507
CodeMethod: Abkürzung für statische Methoden	514
Objekte permanent erweitern	515
Vorlagen für Typ-Erweiterungsdateien	521
Testen Sie Ihr Wissen!	523
Zusammenfassung	524
12 Datentypen umwandeln	525
Den Typ eines Objekts bestimmen	526
Objekte in andere Typen umwandeln	529
Explizite Typumwandlung	530
Nützliche Typen finden	534
Implizite Umwandlungen	536
Daten überprüfen durch Testumwandlung	538
Prüffunktionen entwickeln	539
Typkonvertierungen und Landessitten	541
Kombinierte Prüfungen	541
Testen Sie Ihr Wissen!	543
Zusammenfassung	545
13 Neue Objekte anlegen	547
Neue Objekte durch Umwandlung erstellen	548
Datumsoperationen	548
Direktzugriff auf WMI-Objekte	549

Direktzugriff auf Benutzerkonten	550
Multiple Treffer mit regulären Ausdrücken	554
Ein besseres Array durch Typumwandlung	555
Neue .NET-Objekte mit New-Object	556
Webseiteninhalte abrufen	556
Mit XML arbeiten	561
Konstruktoren verwenden	570
Neue Objekte mit Konstruktoren erstellen	571
Neue COM-Objekte herstellen	574
Sprachausgabe	577
Weitere Beispiele für häufig genutzte COM-Objekte	578
Welche COM-Objekte gibt es sonst noch?	586
Zusammenfassung	586
14 Typen verwenden	589
Typen enthalten weitere Befehle	590
Objekte und Typen – wo liegen die Unterschiede?	590
Eigenschaften und Methoden eines Typs auflisten	595
Statische und dynamische Member	596
Häufig eingesetzte nützliche Typen	597
Mathematische Funktionen verwenden	597
Zahlenformate konvertieren	599
DNS-Auflösung	599
Umgebungsvariablen	600
Pfade zu Systemordnern finden	602
Konsoleneinstellungen	602
Verfügbare Typen suchen und finden	604
Type Accelerators untersuchen	604
Bestehende Objekttypen erforschen	606
In .NET Framework suchen	607
Typen nach Stichwort suchen	611
Typen mit bestimmten Befehlen finden	613
Testen Sie Ihr Wissen!	615
Zusammenfassung	617
15 Typen nachladen und .NET-Code kompilieren	619
Systemassemblies nachladen	620
HTML-Encoding und -Decoding	620
Visual Basic-Befehle in PowerShell	620
Bilderformate konvertieren	624
Dialogfelder, WPF und der STA-Modus	625
Neue Typen mit Membern definieren	627
Vollkommen neue Typen definieren	628
Demotyp mit statischer und dynamischer Methode	628
Eigene Auflistungen für IntelliSense-Unterstützung	629
Neue Typen aus DLL-Dateien laden	631
Zusammenfassung	635

Teil D**PowerShell-Entwickler 637**

16	Pipelinefähige Funktionen	639
	Pipelinefähige Funktionen erstellen	642
	Parameter an Pipelineeingaben binden	643
	Alte Dateien finden	645
	Unterschiedliche Parameter über die Pipeline ansprechen	646
	Mehrere Parameter gleichzeitig empfangen	648
	CSV-Dateien direkt an Funktionen übergeben	649
	Aliasnamen für Parameter	652
	Alle Pipelineergebnisse auf einmal verarbeiten	660
	Pipelineergebnisse sammeln	660
	Fallstricke beim Einsatz von \$input	661
	Zusammenfassung	666
17	Objekte als Rückgabewerte	667
	Vollwertige Objekte zurückliefern	668
	Robust: Leere neue Objekte mit Select-Object	669
	Schnell: Vorinitialisierte Objekte	670
	Langwierig: Member mit Add-Member hinzufügen	671
	Neu: Objekte mit JSON anlegen	672
	ETS-Formatierung festlegen	673
	Zusammenfassung	675
18	Fortgeschrittene Parameter	677
	Unterstützung für IntelliSense	678
	Parameter mit Vorgabewerten ausrüsten	678
	Parameter mit Enumerationen typisieren	679
	Dynamische Vorschlagslisten	684
	Unterstützung für IntelliSense bei Rückgabewerten	685
	Kompatibilität zu älteren PowerShell-Versionen	689
	Mehrere Parametersets definieren	690
	Sich gegenseitig ausschließende Parameter	691
	Standard-Parameterset festlegen	693
	Allgemeine Parameter aktivieren	695
	Parameter -Verbose und -WarningAction	697
	Parameter -ErrorAction und Fehlerhandling	698
	Performance-Optimierung	699
	Simulationsmodus (-WhatIf) und Sicherheitsabfrage (-Confirm) implementieren	699
	Festlegen, welche Codeteile übersprungen werden sollen	700
	Weiterleitung verhindern	701
	Praxisbeispiel: Automatische Auslagerungsdateien aktivieren	702
	Gefährlichkeit einer Funktion festlegen	703
	Dynamische Parameter einsetzen	705
	Dynamische Parameter in Cmdlets	705
	Dynamische Parameter selbst definieren	707

Objektgenerator mit dynamischen Parametern	708
Dynamische Parameter mit dynamischen ValidateSets	710
Argumente »by Reference« übergeben	713
Splatting: Parameter weitergeben	715
Splatting im Alltag einsetzen	715
Übergebene Parameter als Hashtabelle empfangen	716
Mit Splatting Parameter weiterreichen	717
Zusammenfassung	720
19 Proxyfunktionen verstehen und einsetzen	721
Eine Proxyfunktion erstellen	722
Proxyfunktion automatisch erzeugen	722
Bestehende Cmdlets erweitern	723
Automatische Protokollfunktion	724
Get-ChildItem mit neuen Parametern	726
Proxyfunktion anlegen	727
Logik implementieren	731
Projekt: Start-Transcript für PowerShell ISE	739
Proxyfunktionen für Remoting	745
Eine Remotesitzung erstellen	746
Einen Remotebefehl in lokale Sitzung importieren	746
Zusammenfassung	747
20 Eigene Module erstellen	749
Skript in Modul verwandeln	750
Neues Modul anlegen	751
Was beim Modulimport geschieht... ..	752
Allgemeiner Modulaufbau	754
Manifestdatei für ein Modul	754
Neue Manifestdatei anlegen	755
Wirkung einer Manifestdatei	757
ETS-Anweisungen zu Modul hinzufügen	758
Objekte mit eindeutigem Typnamen versehen	760
Aufbau von FormatData-Definitionen	761
Formatdefinition in Modul integrieren	761
Volldynamische Skriptmodule	762
Universeller Modul-Loader als .psm1-Datei	762
Modulkopierfunktion	764
Universelles Manifest als .psd1-Datei	765
Einsatzbereites Universalmodul herstellen	765
Weitere neue Module herstellen	767
Module rückstandslos entfernen	769
Zusammenfassung	770

21	Gültigkeitsbereiche	771
	Skripts im Aufruferkontext	773
	Dotsourcing verstehen	773
	Unterschied zwischen Lesen und Schreiben	775
	Globale Variablen verwenden	776
	Globale Variablen	777
	Skriptglobale Variablen	778
	Aufruftyp eines Skripts testen	779
	Aufpassen bei Objekten und Referenzen	781
	Auf beliebige Gültigkeitsbereiche zugreifen	783
	Gültigkeitsbereiche in Modulen	787
	Auf den Aufruferkontext zugreifen	787
	Auf den Modulkontext zugreifen	788
	Zusammenfassung	789
22	Sicherheit und Signaturen	791
	Ein Zertifikat auswählen	792
	Installiertes Zertifikat auswählen	792
	PFX-Dateien für die Signierung laden	796
	Selbstsigniertes Zertifikat herstellen	799
	PowerShell-Skripts signieren	802
	Mehrere Skripts und ganze Ordnerstrukturen signieren	804
	Signaturen mit einem Zeitstempel versehen	805
	Signaturen überprüfen	805
	Zusammenfassung	808
	Teil E	
	Spezielle Techniken	809
23	Windows PowerShell-Remoting	811
	Klassisches Remoting	812
	Remotefähige Befehle finden	813
	Klassisches Remoting einsetzen	815
	Troubleshooting für klassisches Remoting	819
	WMI-Remoting einrichten	823
	Remote-Registrierungszugriff erlauben	824
	Universelles PowerShell-Remoting	825
	PowerShell-Remoting einschalten	826
	Enter-PSSession: Interaktive Remotekonsole	830
	Enter-PSSession führt nur interaktive Eingaben remote aus	831
	Kontrolle: Wer besucht meinen Computer?	832
	Remoting-Unterstützung im ISE-Editor	832
	Invoke-Command: Remoteausführung von PowerShell-Code	834
	Lokale Variablen und Abhängigkeiten	834
	Double-Hop und CredSSP: Anmeldeinfos weiterreichen	840

New-PSSession: Dauerhafte Remotesitzungen	843
Eigene Sitzungen anlegen	843
Sitzungen zu mehreren Computern: Fan-Out	844
Sitzungen vorübergehend trennen	845
Getrennte Sitzungen verwenden	845
Import-PSSession: Implizites Remoting	846
Remotesitzungen als Modul exportieren	848
Sitzungskonfigurationen anlegen und verwalten	848
Neue Konfiguration anlegen	850
Berechtigungen auf Sitzungskonfigurationen setzen	852
Datentransfer und Performance-Optimierung	853
Serialisierung	853
Optimierungsansätze	855
WSMan: Ports, Timeouts und andere Remoting-Einstellungen	855
Clientseitige Optionen	855
Serverseitige Optionen	856
Fehler finden und beheben	857
RPC-Server nicht verfügbar	857
Zugriff verweigert	859
Kerberos-Fehlermeldung	860
Öffentliche Netzwerke entdeckt	861
Andere Fehler	862
Zusammenfassung	862
 24 Hintergrundjobs	 863
Hintergrundjobs verwenden	864
Hintergrundjob anlegen	865
Laufende Hintergrundjobs kontrollieren	865
Ergebnisse eines Hintergrundjobs abrufen	865
Hintergrundjobs abschließen	866
Parallelverarbeitung für mehr Geschwindigkeit	866
Der Parameter -AsJob	867
Hintergrundjobs auf Remotecomputern starten	868
Weitere Remotejobverfahren	869
Was Sie bei Hintergrundjobs bedenken sollten... ..	871
Lightweight-Threads für Hintergrundjobs	872
Einen separaten Thread erzeugen	873
Hintergrundüberwachungen einrichten	875
InProgress-Jobs einsetzen	878
Testen Sie Ihr Wissen!	884
Zusammenfassung	886
 25 Ereignisverarbeitung	 887
Ereignisse verwenden	888
Ein Ereignis überwachen	888
Ereignisüberwachung wieder abschalten	889
Auf Events warten	890

Hintergrundjobs überwachen	891
Manuelle Überwachung	891
Automatische Überwachung	892
Ordner überwachen	893
Aufgaben regelmäßig durchführen	894
WMI-Ereignisse empfangen	895
Details zum Event erfahren	896
Systemänderungen erkennen	896
Eigene Ereignisse auslösen	897
Automatische Variablenüberwachung einrichten	898
Zusammenfassung	899
26 Workflows	901
Parallelverarbeitung mit Workflows	902
Das Schlüsselwort »parallel«	903
Die foreach-Schleife im Parallelmodus	904
Flexibles Remoting	906
Persistenz	907
Workflows unterbrechen sich selbst	907
Persistierte Workflows	907
Einschränkungen in Workflows	908
Verbotene Sprachelemente	909
Nicht unterstützte Cmdlets	909
Zusammenfassung	909
27 Benutzeroberflächen gestalten	911
Windows Presentation Foundation	912
Eigene Fenster öffnen	912
Spezialfenster ohne Rahmen	913
Transparenz und Bildschirmsperre	914
Warum WPF so relativ einfach ist	916
Ereignishandler	917
Elemente im Fenster anordnen	918
StackPanels	919
Grids	919
DockPanels	920
Zusammenfassung	924
28 Erweiterungen für den ISE-Editor	925
Das ISE-Objektmodell	926
\$psISE – der Zugang zum Objektmodell	926
ISE-Optionen automatisiert setzen	927
Standardkonfigurationen und Profilskripts	928
Set-TokenHighlight: Vorübergehende Hervorhebungen	929
PowerShell-Parser und abstrakte Syntaxstruktur	931
Skript in Token verwandeln	931
Tools, die auf dem Parser aufsetzen	933
Variablendokumentation	933

Kommentare entfernen	935
Variablen umbenennen	936
Aliase auflösen	937
Add-Ons-Menü verwenden	938
Komplexe Menüs erstellen	938
Zusammenfassung	940
 Stichwortverzeichnis	 941
 Über den Autor	 959