

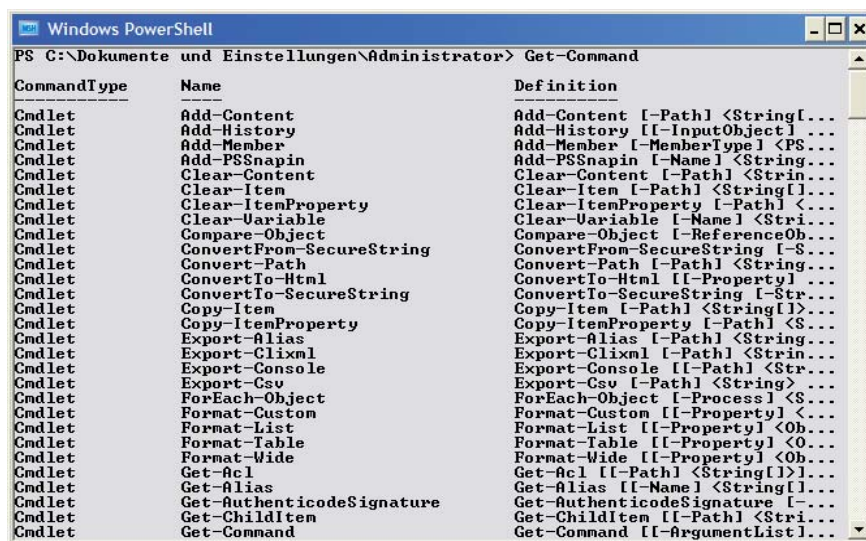
Neues Power-Tool für Windows

Windows Power Shell ist die von Grund auf neu gestaltete Kommandozeile von Windows XP und Vista. Mit dem Nachfolger der DOS-Box haben Sie die Registry, Windows-Prozesse und Ihre Daten voll im Griff.

Die für Windows Vista entwickelte Kommandozeilenumgebung Windows Power Shell arbeitet nach einem völlig neuen, objektorientierten Konzept. Registry-Inhalte und andere Systeminformationen ruft sie direkt ab und manipuliert sie auch. Wie die Power Shell die Daten weiterverarbeitet, legt der Nutzer mit wenigen Kommandos und stets nach demselben Schema fest.

Die Windows Power Shell lässt sich zudem sehr gut programmieren. Power-Shell-Skripts können viel mehr als die Batch-Programme der herkömmlichen Kommandozeilenumgebung. Der Kasten „Windows Power Shell: Das ist neu“ auf Seite 42 fasst die wesentlichen Veränderungen zusammen, die die Windows Power Shell mitbringt.

Im Kasten „DOS-Kommandos: neue Befehle unter alten Namen“ auf Seite 42 finden Sie die Power-Shell-Nachfolger der klassischen Befehle, die Sie aus der bisherigen DOS-Box kennen. Alle neuen Commandlets im Überblick listet der Befehl `Get-Command` innerhalb der Windows Power Shell (Bild A). Wie Sie Skriptprogramme schreiben, erklärt der Kasten „Skripts: Power-Shell-Programme“ auf Seite 43. Der unten ste-



| CommandType | Name | Definition |
|-------------|---------------------------|-----------------------------------|
| Cmdlet | Add-Content | Add-Content [-Path] <String>... |
| Cmdlet | Add-History | Add-History [[-InputObject] ... |
| Cmdlet | Add-Member | Add-Member [-MemberType] <PS... |
| Cmdlet | Add-PSSnapin | Add-PSSnapin [-Name] <String>... |
| Cmdlet | Clear-Content | Clear-Content [-Path] <String>... |
| Cmdlet | Clear-Item | Clear-Item [-Path] <String>... |
| Cmdlet | Clear-ItemProperty | Clear-ItemProperty [-Path] <... |
| Cmdlet | Clear-Variable | Clear-Variable [-Name] <Stri... |
| Cmdlet | Compare-Object | Compare-Object [-ReferenceOb... |
| Cmdlet | ConvertFrom-SecureString | ConvertFrom-SecureString [-S... |
| Cmdlet | Convert-Path | Convert-Path [-Path] <String>... |
| Cmdlet | ConvertTo-Html | ConvertTo-Html [[-Property] ... |
| Cmdlet | ConvertTo-SecureString | ConvertTo-SecureString [-Str... |
| Cmdlet | Copy-Item | Copy-Item [-Path] <String>... |
| Cmdlet | Copy-ItemProperty | Copy-ItemProperty [-Path] <S... |
| Cmdlet | Export-Alias | Export-Alias [-Path] <String>... |
| Cmdlet | Export-Clixml | Export-Clixml [-Path] <Strin... |
| Cmdlet | Export-Console | Export-Console [-Path] <Str... |
| Cmdlet | Export-Csv | Export-Csv [-Path] <String> ... |
| Cmdlet | ForEach-Object | ForEach-Object [-Process] <S... |
| Cmdlet | Format-Custom | Format-Custom [-Property] <... |
| Cmdlet | Format-List | Format-List [[-Property] <Ob... |
| Cmdlet | Format-Table | Format-Table [[-Property] <O... |
| Cmdlet | Format-Wide | Format-Wide [[-Property] <Ob... |
| Cmdlet | Get-Acl | Get-Acl [-Path] <String>... |
| Cmdlet | Get-Alias | Get-Alias [-Name] <String>... |
| Cmdlet | Get-AuthenticodeSignature | Get-AuthenticodeSignature [-... |
| Cmdlet | Get-Childitem | Get-Childitem [-Path] <Stri... |
| Cmdlet | Get-Command | Get-Command [[-ArgumentList]... |

Get-Command: So erhalten Sie eine Liste mit allen Befehlen der Windows Power Shell (Bild A)

hende Kasten „Bei Fragen: Die Online-Hilfe der Windows Power Shell“ zeigt, wie Sie die Online-Hilfe nutzen.

Windows Power Shell installieren

Die Windows Power Shell baut auf dem .NET Framework 2.0 Redistributable (x86) von Microsoft (<http://www.microsoft.com/germany/msdn/netframework>, kostenlos) auf und benötigt diese Soft-

ware, um zu funktionieren. Installieren Sie also zunächst das .NET Framework 2.0. Sie finden es auf Heft-CD und -DVD in der Rubrik „Computer, Windows Power Shell“. Spielen Sie im Anschluss daran die neueste Ausgabe der Windows Power Shell auf (www.microsoft.com/windowsserver2003/technologies/management/powershell, kostenlos). Wählen Sie beim ersten Start die Option **A**. Damit erteilen Sie der Shell erst die notwendige Betriebslaubnis. Die Eingabeaufforderung erscheint. Geben Sie das folgende Kommando ein:

```
1 Set-ExecutionPolicy RemoteSigned
```

Nur dann dürfen Sie eigene Skriptdateien mit der Endung SP1 ausführen.

Neue Tricks mit dem „dir“-Befehl

Wie die Windows-Kommandozeilenumgebung enthält auch die Windows Power Shell einen „dir“-Befehl, der die Elemente des aktuellen Verzeichnisses auflistet. Der entsprechende Befehlssteil – Commandlet genannt – heißt `Get-Childitem`. Mit anderen Befehlssteilen

Bei Fragen: Die Online-Hilfe der Windows Power Shell

Die wichtigsten Fragen zu einem Commandlet klärt das Hilfe-Commandlet.

Mit dem Hilfe-Commandlet `Get-Help` erfahren Sie zum Beispiel, welche Argumente und welche Optionen beim Aufruf des Prozess-Commandlets möglich sind.

Das Commandlet `Get-Command` gibt die Liste aller Commandlets aus. Mit `Get-Command | Format-List` erhalten Sie zusätzlich Informationen zum Gebrauch.

Mit `Get-Alias` sehen Sie, welche Abkürzungen für die Commandlets definiert sind.

Das Commandlet `Get-Member` zeigt die Objekteigenschaften von Commandlets an. Damit finden Sie zum Beispiel heraus, dass `(Get-Date).Minute` dem Minutenwert des aktuellen Datums entspricht. Geben Sie `Get-Date | Get-Member` ein. Die Ausgabe enthält die Liste von Methoden und Eigenschaften des Commandlets. Methoden stehen am Anfang der Liste. Ihr „MemberType“ heißt „Method“. Eigenschaften vom Typ „Property“ stehen am Schluss. Den Wert einer Eigenschaft erhalten Sie mit `(<Commandlet>.<Eigenschaftsname>.`

Steckbrief: Neues Power-Tool für Windows XP

Kompakt

Die neue Kommandozeilenumgebung für Windows XP und Vista heißt Windows Power Shell.

Die Windows Power Shell lässt sich mit Windows XP mit Service Pack 2 nutzen. Der Artikel bezieht sich auf den ersten Release Candidate von Windows Power Shell.

Eine Befehlszeile setzt sich aus mehreren Teilbefehlen zusammen. Sie heißen Commandlets. Die Windows Power Shell enthält 129 Commandlets.


Inhalt


- Windows Power Shell installieren S. 40
- Neue Tricks mit dem „dir“-Befehl S. 40
- Installierte Hardware auf einen Blick S. 41
- Geheimen Prozessen auf der Spur S. 42
- Suchen und Ersetzen im Handumdrehen S. 43
- Registry-Schlüssel kontrollieren S. 43
- Bei Fragen: Die Online-Hilfe der Windows Power Shell S. 40
- DOS-Kommandos: Neue Befehle unter alten Namen S. 42
- Windows Power Shell: Das ist neu S. 42
- Skripts: Ihre Programme S. 43

Weitere Infos

- www.microsoft.com/windowsserver2003/technologies/management/powershell Homepage zur Windows Power Shell
- <http://arstechnica.com/guides/other/msh.ars> Gute Einführung in die Windows Power Shell
- <http://channel9.msdn.com/wiki/default.aspx/Channel9.MSHWiki> Inoffizielle Infobörse zur Windows Power Shell

Software-Übersicht

| Programm | Beschreibung | Seite |
|--|--|-------|
|  .NET Framework 2.0 Redistributable (x86) | Die Betriebssystemerweiterung ist für die Windows Power Shell erforderlich | 40 |
| Windows Power Shell RC1 x86 | Neue Kommandozeilenumgebung für Windows Vista | 40 |

Das -Programm finden Sie auf Heft-CD und -DVD in der Rubrik „Computer, Windows Power Shell“.

```
1 Get-ChildItem -Recurse c:\ |
Where-Object {$_.LastWriteTime -gt (Get-Date).AddMinutes(-5)}
```

Dabei bedeutet **-Recurse c:**, dass das Laufwerk „C:“ mit allen Unterverzeichnissen durchsucht wird. **\$.LastWriteTime** steht für den Zeitpunkt des letzten Zugriffs auf eine Datei oder ein Verzeichnis. Der Wert **(Get-Date).AddMinutes(-5)** entspricht dem jeweils aktuellen Zeitpunkt minus fünf Minuten. Die Bedingung in geschweiften Klammern vergleicht die beiden Zeitpunkte mit dem Größer-Vergleichsoperator **-gt**.

Beispiel 2 – alte Dateien löschen: Mit folgendem Aufruf entfernen Sie im aktuellen Verzeichnis alle Dateien, die älter als ein Jahr sind:

```
1 Get-ChildItem | Where-Object
{$_.LastWriteTime -lt (Get-Date).AddYears(-1)} | Remove-Item
```

Das Kommando ähnelt dem Befehl im Beispiel 1 – nur dass auf das Commandlet **Where-Object** noch der Löschbefehl **Remove-Item** als weiteres Übergabekommando folgt. Das Kürzel **-lt** steht für den Vergleichsoperator kleiner. Vorsicht: Das Kommando löscht die alten Dateien im aktuellen

Verzeichnis ohne nochmalige Rückfrage und ohne Bestätigung.

Installierte Hardware auf einen Blick

Die Windows Power Shell greift direkt auf Systeminformationen zu. Auf der Kommandozeile rufen Sie Angaben zu Prozessor, BIOS, Speicher und Betriebssystem ab.

So geht's: Alle Informationen der Verwaltungsbibliothek Windows Management Instrumentation – kurz WMI – stehen auf der Windows Power Shell mit dem Commandlet **Get-WmiObject** zur Verfügung. WMI sammelt Daten zum Betriebssystem, zum BIOS und zu anderen Themen. Jedem Thema ist eine eigene Abteilung der WMI-Bibliothek gewidmet.

Die Abteilungen heißen Klassen. Ihre Bezeichnungen sind englisch, zum Beispiel **Win32_OperatingSystem**, **Win32_BIOS**, **Win32_Processor** oder **Win32_PhysicalMemory**.

Beispiel 1 – Systeminformationen anzeigen: Das folgende Kommando bewirkt, dass alle WMI-Angaben zum Prozessor angezeigt werden:

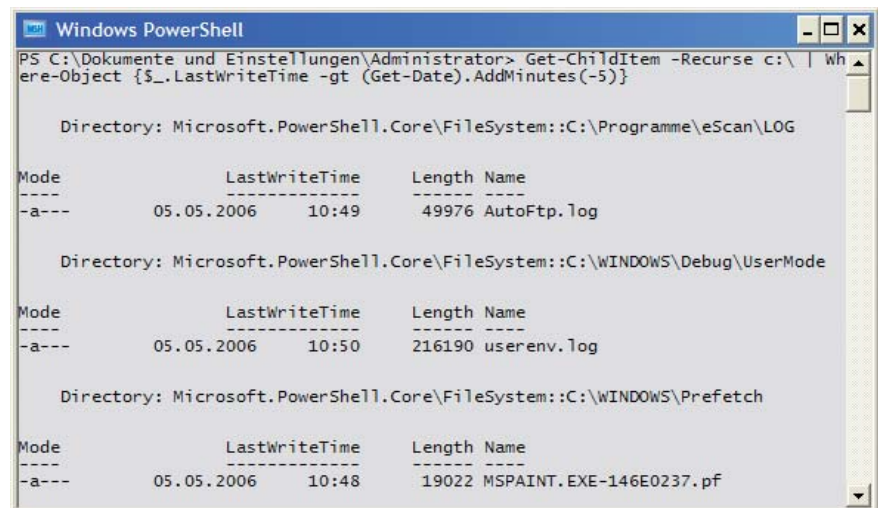
```
1 Get-WmiObject Win32_Processor
```

Auf dem Bildschirm erscheint eine Liste von Eigenschaften. Dabei stehen links die Namen der Eigenschaften. Auf ▶

kombiniert zeigt das Commandlet **Get-ChildItem** kürzlich geänderte Dateien an oder löscht alte Dateien.

So geht's: Mit dem Platzhalter ***** legen Sie genau fest, welche Dateien angezeigt werden. So zählt zum Beispiel das Kommando **Get-ChildItem *test*** nur Dateien und Verzeichnisse auf, deren Namen „test“ enthält. Neu ist, dass man die aufgelisteten Dateien gleich nach der Ausgabe einem weiteren Commandlet übergeben kann. Die Übergabe funktioniert mit dem Pipe-Zeichen **|**. Das wichtigste Übergabekommando heißt **Where-Object**. Es verknüpft den Befehl **dir** oder ein anderes Kommando mit einer Bedingung und steuert so die Ausgabe. Die Bedingung steht in geschweiften Klammern.

Beispiel 1 – vermisste Dateien finden: Folgendes Kommando zeigt alle Dateien an, die in den letzten fünf Minuten gespeichert wurden. Damit finden Sie zum Beispiel eine vor kurzem heruntergeladene Datei, ohne deren Pfad und Namen zu wissen (Bild B):



Get-ChildItem: Mit **Where-Object** listet das Commandlet kürzlich geänderte Dateien auf (Bild B)

der rechten Seite finden Sie dazu gehörenden Werte. Mit dem Kommando

```
1 (Get-Wmiobject Win32_Process).<Eigenschaftsname>
```

rufen Sie jeweils genau eine der Eigenschaften ab.

Um Daten zum BIOS zu bekommen, setzen Sie statt `Win32_Processor` den Klassennamen `Win32_BIOS` ein. Die Klassen `Win32_OperatingSystem` und `Win32_PhysicalMemory` liefern Informationen zum Betriebssystem beziehungsweise zum Arbeitsspeicher.

Beispiel 2 – Speicherbelegung anzeigen:

Mit dem folgenden Kommando erfahren Sie, welche Speicherbänke Ihr PC enthält, und wie sie belegt sind:

```
1 Get-WmiObject Win32_PhysicalMemory | Format-Table Tag, BankLabel, DeviceLocator, Capacity
```

Das Format-Commandlet `Format-Table` bewirkt die Ausgabe als Tabelle. Danach sind die Eigenschaften genannt, die die Tabelle zeigen soll.

Geheimen Prozessen auf der Spur

Das Prozess-Commandlet der Windows Power Shell gibt die Programmpfade der laufenden Prozesse aus – was der Task-Manager nicht tut. Damit eignet

DOS-Kommandos: Neue Befehle unter alten Namen

Einige Commandlets der Windows Power Shell gibt es in ähnlicher Form bereits in der herkömmlichen Kommandozeile. Sie lassen sich mit den alten Befehlsnamen aufrufen.

| Commandlet | Alias | Funktion |
|---------------|------------|---|
| Clear-Host | cls | Bildschirm löschen |
| Copy-Item | copy | Datei kopieren |
| Get-ChildItem | dir | Dateinamen eines Verzeichnisses auflisten |
| Get-Content | type | Dateinhalt ausgeben |
| Move-Item | move | Datei umbewegen |
| Remove-Item | del, rmdir | Datei oder Ordner löschen |
| Set-Location | cd | Verzeichnis wechseln |
| Set-Variable | set | Variable festlegen |
| Sort-Object | sort | Sortieren |
| Write-Output | echo | Variable oder Text ausgeben |

sich das Commandlet, um Trojanerprozesse aufzuspüren, die man oft am Programmpfad erkennt.

So geht's: Das Commandlet `Get-Process` ruft alle Prozessdaten ab. Mit `Format-Commandlets` wie `Format-Table` und `Where-Object` filtern Sie genau die Informationen heraus, die Sie benötigen.

Beispiel 1 – laufende Prozesse anzeigen: Folgendes Kommando listet die laufenden Prozesse, ihre CPU-Auslastung

und eine Kurzbeschreibung auf:

```
1 Get-Process | Sort-Object -Descending CPU | Format-Table -wrap ProcessName, CPU, Description
```

Das Commandlet `Sort-Object` mit der Option `-Descending CPU` bewirkt, dass die Ausgabe nach der CPU-Auslastung geordnet wird, und zwar in absteigender Reihenfolge. Mit `Format-Table` und den folgenden Eigenschaftsnamen lassen Sie sich den Prozessnamen, die CPU-Auslastung und die Beschreibung in einer Tabelle anzeigen. Die Option `-wrap` schaltet den Zeilenumbruch für lange Beschreibungen ein.

Beispiel 2 – Schädliche Prozesse aufspüren: Mit folgendem Kommando lassen Sie sich die Namen aller mehrfach gestarteten Prozesse und die Pfade ausgeben. Wenn die Liste Mehrfachprozesse mit verschiedenen Pfadnamen enthält, sollten Sie aufmerksam werden. Trojaner laufen nämlich oft unter dem Namen eines Systemprozesses. Lediglich der Pfad stimmt nicht mit dem Systemverzeichnis überein:

```
1 Get-Process | Group-Object {$_.Name} | Where-Object {$_.Count -gt 1} | Foreach-Object {Get-Process $_.Name} | Format-List Name, Path
```

Das Beispiel zeigt, wie komplex ein einzelner Befehl werden kann. Der Aufruf enthält fünf Commandlets, die jeweils mit einem Pipe-Zeichen getrennt sind. Ein Commandlet übergibt sein Ergebnis an das jeweils nächste. Das Commandlet `Group-Object` mit dem Argument `$_Name` gruppiert die Prozesse von `Get-Process` ihrem Namen nach. Das Commandlet `Where-Object` filtert

Windows Power Shell: Das ist neu

Mit der Power Shell erhält Windows eine völlig neue Kommandozeilenumgebung – mit Commandlets, Objekten und virtuellen Laufwerken.

Ein Befehl der Windows Power Shell besteht – ähnlich einem Satz – aus mehreren Teilen, den so genannten Commandlets.

Das Pipe-Zeichen: Zwischen den Commandlets eines Befehls steht meistens das Pipe-Zeichen `|`. Die Zeichenfolge

```
1 Commandlet 1 | Commandlet 2
```

bedeutet: `Commandlet 2` erhält das Ergebnis von `Commandlet 1` als Eingabe.

Beispiel: Das Commandlet `Get-ChildItem` listet die Dateinamen des aktuellen Verzeichnisses auf. Das Commandlet `Out-Host` mit der Option `-Paging` bewirkt, dass die Ausgabe in mehreren Seiten erfolgt. Das ehemalige `dir /p` lautet auf der Windows Power Shell `Get-ChildItem | Out-Host -Paging`.

Werte direkt abrufen: Die Power Shell ist objektorientiert. Falls ein Commandlet mehrere Werte beziehungsweise Eigenschaften zurückgibt, lässt sich jeder Wert direkt abrufen.

Beispiel: Das Commandlet `Get-Date` gibt das Datum und die Uhrzeit aus. Das Kommando `(Get-Date).Minute` zeigt nur die Minuten an.

So genannte Methoden ändern die ausgegebenen Werte bei Bedarf: `(Get-Date).AddMinutes(5)` zeigt das Datum plus fünf Minuten an. Auch Kombinationen der Methoden sind möglich: `((Get-Date).AddHours(-1)).Hour` gibt die aktuelle Stunde in der Greenwich-Zeit aus, so wie sie auf Uhren in Paris und London abzulesen ist.

Virtuelle Laufwerke Die Windows Power Shell arbeitet mit virtuellen Laufwerken – so genannten Providern. Einer dieser Provider ist die Windows-Registry. Im virtuellen Laufwerk der Registry erscheinen die Registry-Schlüssel als Verzeichnisse.

Beispiel: Der Befehl

```
1 Set-Location HKCU:\Software\Microsoft\Windows\CurrentVersion\Run
```

wechselt in den Teilschlüssel „Run“ von „HKEY_CURRENT_USER“. Das Kommando `Get-Property` zeigt die Parameter des Schlüssels an.

die Prozesse heraus, die mehr als einmal vorkommen. Dabei gibt das Argument `$_ .Count` an, wie oft der Prozess eines Namens in der Liste enthalten ist. Das Commandlet `Foreach-Object` ruft für jeden der ausgewählten Prozesse erneut das Prozesskommando `Get-Process` auf. Der Befehl `Format-List -Name, -Path` steuert schließlich die Ausgabe.

Suchen und Ersetzen im Handumdrehen

Beim Verwalten von Textdokumenten ist die neue Shell sehr nützlich. Ein Befehl genügt, um in allen Dateien eines Verzeichnisses einen alten Begriff durch einen neuen zu ersetzen.

So geht's: Mit den Commandlets `Get-Content` und `Set-Content` lässt sich der Inhalt einer Textdatei ausgeben und nach einer Änderung wieder zurückschreiben. Das eigentliche Suchen und Ersetzen erledigt die Methode `Replace` des Commandlets `Get-Content`.

Beispiel 1 – Begriff in einer Datei ersetzen: Mit folgendem Kommando tauschen Sie in der Datei „Test.txt“ alle „Photo“ gegen „Foto“ aus und speichern das Ergebnis als neue Datei mit dem Namen „Test_Ersetzt.txt“ ab:

```
1 Get-Content "Test.txt" | .
   Foreach-Object {$_.Replace
   ("Photo", "Foto")} | .
   Set-Content "Test_Ersetzt.txt"
```

Das Commandlet `Get-Content` übergibt den Text der Datei Zeile für Zeile an den folgenden Befehl `Foreach-Object`. Dieser leitet eine Befehlsschleife ein. Das Ersetzen-Kommando in den

geschweiften Klammern wird dadurch für jede Zeile der Datei `Test.txt` ausgeführt.

Bei Dokumenten im Word-Format DOC funktioniert die Methode nicht. Bei Dokumenten, die im Rich-Text-Format mit der Endung RTF abgespeichert sind, klappt sie meistens nur bei kurzen Suchbegriffen nicht, falls diese in einem der versteckten RTF-Steuerbefehle vorkommen.

Beispiel 2 – Begriff in mehreren Dateien ersetzen:

Sie haben auch die Möglichkeit, Textpassagen in mehreren Dokumenten auf einen Rutsch zu ersetzen. So ändern Sie etwa nach einem Umzug Ihre Adresse in allen Textdokumenten des aktuellen Verzeichnisses von Wiesenweg 10 in Parkallee 29:

```
1 Get-ChildItem *.txt | .Foreach-
   Object {Get-Content $_ | Out-
   String | .Foreach-Object {$_.
   Replace("Wiesenweg 10",
   "Parkallee 29")} | .
   Set-Content $_}
```

Das Kommando `Get-ChildItem *.txt` listet alle Textdateien im aktuellen Verzeichnis auf. Das Commandlet `Foreach-Object` führt für jede Datei der Liste den in geschweiften Klammern stehenden Aufruf für Suchen und Ersetzen aus. Das Commandlet `Out-String` bewirkt, dass die Textdateien des aktuellen Verzeichnisses jeweils komplett ausgelesen und geschlossen werden, bevor das Commandlet `Foreach-Object` den Inhalt weiterbearbeitet. Das ist nötig, damit das Commandlet `Set-Content` den geänderten Inhalt wieder mit demselben Dateinamen speichern kann.

Skripts: Ihre Programme

Wie bei der herkömmlichen Windows-Kommandozeilenumgebung lassen sich auch bei der Windows Power Shell mehrere Befehlszeilen zu einem Programm zusammenfügen.

Schreiben Sie die Befehlszeilen in eine Textdatei mit der Endung „PS1“ – zum Beispiel `script.ps1`. Starten Sie das Skript mit dem Kommando `& "<Pfad>\script.ps1"`.

Beispiel: Folgendes Programm addiert zwei Zahlen.

```
1 $Summe=$args[0]+$args[1]
2 Write-Output "Die Summe lautet:"
3 Write-Output $Summe
```

Wenn man etwa das Programm mit `& ". \script.ps1" 10 25` aufruft, erscheint das Ergebnis „35“ auf dem Bildschirm. Die als Standard definierte Variable `$args[0]` enthält das erste Argument, `$args[1]` das zweite Argument und so weiter.

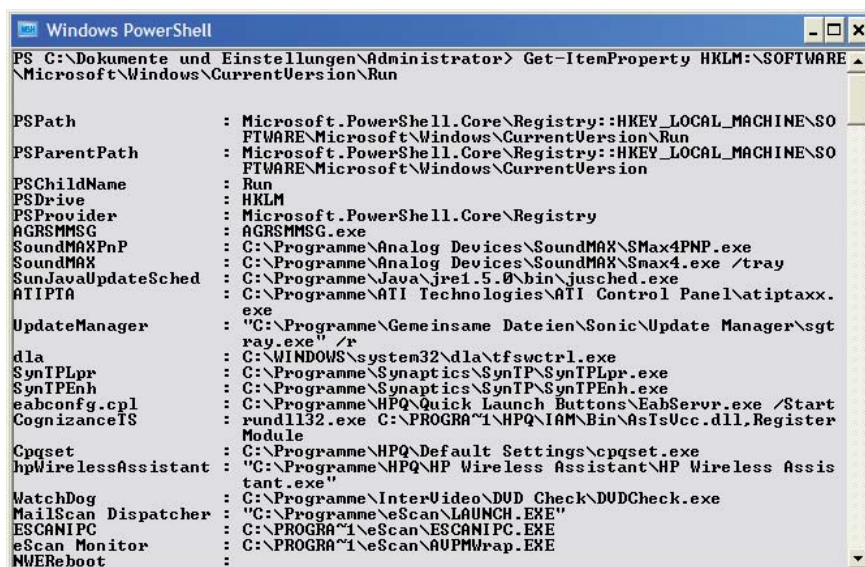
Registry-Schlüssel kontrollieren

Die Windows Power Shell enthält ein virtuelles Laufwerk für die Windows-Registry. Darin stehen alle Registry-Schlüssel samt ihren Parametern. Der Vorteil: Einzelne Werte der Windows-Registry lassen sich schnell abrufen, ohne dass Sie den Registrierungs-Editor öffnen müssten.

So geht's: Das CHDIR-Commandlet `Set-Location` führt Sie zum gewünschten Registry-Schlüssel. Die Teilschlüssel von „HKEY_LOCAL_MACHINE“ stehen in einem Unterverzeichnis des Laufwerks „HKLM:“, während die Teilschlüssel von „HKEY_CURRENT_USER“ als Unterverzeichnisse von „HKCU:“ zu finden sind. Die Parameter und ihre Werte ruft man mit dem Commandlet `Get-ItemProperty` ab.

Beispiel – Trojaner stoppen: Die folgenden beiden Befehle zeigen Autostart-Einträge an, die sich in der Registry verstecken. Trojaner nisten sich dort oft ein und werden bei jedem Systemstart aktiv (Bild C):

```
1 Get-ItemProperty HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
2 Get-ItemProperty HKCU:\Software\Microsoft\Windows\CurrentVersion\Run
```



Get-ItemProperty: Mit dem Commandlet lassen sich kritische Registry-Schlüssel direkt anzeigen (Bild C)

Klaus Plesser
computer@com-magazin.de