O'REILLY®

6. Auflage
Inklusive PowerShell-Alternativen

Windows-Befehle für Server 2016 und Windows 10

kurz & gut

O'REILLYS TASCHENBIBLIOTHEK

Olaf Engelke

6. AUFLAGE

Windows-Befehle für Server 2016 und Windows 10

kurz & gut

Olaf Engelke

O'REILLY®

Olaf Engelke

Lektorat: Alexandra Follenius
Korrektorat: Sibylle Feldmann, www.richtiger-text.de
Herstellung: Susanne Bröckelmann
Umschlaggestaltung: Michael Oréal, www.oreal.de
Satz: III-satz, www.drei-satz.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

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.

ISBN:
Print 978-3-96009-069-4
PDF 978-3-96010-165-9
ePub 978-3-96010-166-6
mobi 978-3-96010-167-3

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem Imprint »O'REILLY«. O'REILLY ist ein Markenzeichen und eine eingetragene Marke von O'Reilly Media, Inc. und wird mit Einwilligung des Eigentümers verwendet.

6. Auflage

Wieblinger Weg 17
69123 Heidelberg

5 4 3 2 1 0

Inhalt

Einführung 7
Die Eingabeaufforderung 10
Allgemeine Befehle 21
Dateien und Verzeichnisse 37
Dateisysteme, Volumen und Festplatten 70
Drucker und Warteschlangen 101
Registrierung 110
Prozesse 118
Dienste 124
Berechtigungen und Rechte 131
Systemdiagnose und -information 140
Systemkonfiguration 161
Netzwerk 176
Internet Information Server 207
Benutzer und Gruppen 210
Active-Directory-Verzeichnisdienst 213
Cluster 245
Remotedesktopdienste 254
Installation und Deployment 259

Skripte und Batchdateien 262
Zertifikate 272
Die Wiederherstellungsumgebung 277
Konstrukte in Batchdateien 277
Windows PowerShell – Grundlagen 279
LDAP-Suchfilter 282
Windows-GUI – Tipps und Tricks 284
Windows im WWW 288

Index 291

Windows-Befehle für Server 2016 und Windows 10

Einführung

Am Anfang war die Dunkelheit. Ein meist schwarzer Bildschirm mit weiß, bernsteingelb oder grün leuchtender Befehlszeile, in der man dem Computer mit mehr oder weniger mühsam erlernten Befehlen und Tastenkombinationen sagen konnte, was er zu tun hatte, war die dominierende Schnittstelle für Administratoren und Anwender. Es sollte einige Zeit dauern, bis grafische Benutzeroberflächen, geeignet auch für eine Mausbedienung, die Eingabeaufforderung ablösten.

Bei modernen Windows-Versionen wie Windows 10/Server 2016 tritt diese Eingabeaufforderung kaum noch in Erscheinung, stattdessen wird sie mehr denn je vor dem Auge des Anwenders verborgen. Das liegt nicht daran, dass sie plötzlich alle Bedeutung verloren hätte. Vielmehr geht Microsoft wohl davon aus, dass die meisten Anwender ohnehin zu den in der grafischen Benutzeroberfläche bereitgestellten Werkzeugen greifen und Profis sich mit der PowerShell arrangieren, die seit Windows 7 integraler Bestandteil des Betriebssystems geworden ist und sich in den aktuellen Versionen zunehmend anschickt, die alte Eingabeaufforderung zu ersetzen. Deutlich zu sehen ist das daran, wie in den aktuellen Versionen von Windows als veraltet deklarierte Befehle durch PowerShell-Cmdlets ersetzt wurden und neuere Features wie Virtualisierung maßgeblich oder ausschließlich über PowerShell verwaltbar sind. Gerade unter Windows Server misst Microsoft der Administration mit Windows PowerShell eine erheblich gestiegene Bedeutung bei, während neue Befehle für die klassische Eingabeaufforderung mit der Lupe zu suchen sind. Dieser Tatsache versucht das Buch Rechnung zu tra-

gen und die Leser bei der Transformation zu PowerShell durch den einen oder anderen Fingerzeig zu unterstützen.

Für wen dieses Buch gedacht ist

Diese Referenz beschreibt die meisten Befehle der Windows-Eingabeaufforderung (die oft auch als Befehlszeile, Kommandozeile, Konsole oder DOS-Prompt bezeichnet wird) in der aktuellen Version von Windows und benennt, sofern es sich anbietet, deren Äquivalente in Windows PowerShell. Sie ist nicht nur für Systemadministratoren gedacht, sondern auch für normale Windows-Anwender. Enthalten ist die Mehrzahl der Befehle von Windows in den zum Zeitpunkt der Entstehung dieser Auflage aktuellen Versionen von Client und Server. Etliche der Befehle haben eine weiter zurückreichende Geschichte und sind in derselben Form oder mit abgewandelter Syntax auch in früheren Fassungen des Windows-Betriebssystems vorhanden.

Was dieses Buch nicht enthält

Aufgrund des kompakten Formats der Reihe »kurz & gut« wurden selten genutzte und sehr spezielle Befehle teilweise nicht in diese Referenz aufgenommen. Einige weitere Befehle werden nicht behandelt, weil entweder ihre Funktion bereits komplett von einem anderen Befehl übernommen wurde oder sie schlicht veraltet sind. Daher sind Befehle, die es ausschließlich für frühere Versionen gab, nur noch in Ausnahmen in dieser Ausgabe enthalten. Zudem wurden Informationen zu einigen Befehlen gekürzt, um Raum vor allem für Hinweise zu PowerShell-Alternativen zu schaffen und das Buch dabei weiterhin möglichst kompakt zu halten. Zum Teil fehlen auch solche Befehle, die nur nach der Installation spezieller Rollen oder Funktionen des Betriebssystems verfügbar sind.

Ebenfalls nicht enthalten sind Programme mit grafischer Benutzeroberfläche, da sich dieses Buch auf Befehle beschränkt, die in der Eingabeaufforderung bzw. in Skripten nutzbar sind. Ausnahmen bilden lediglich solche Programme, die sowohl über eine grafische Oberfläche verfügen als auch komplett oder in wesentlichen Teilen aus der Eingabeaufforderung heraus gesteuert werden können.

Auch auf Linux-Befehle wird nicht eingegangen, wenngleich diese mit dem als Feature installierbaren Windows-Subsystem für Linux ebenfalls die Windows-Arena betreten.

Aufbau

Die Befehle sind nach Funktionsgruppen geordnet und innerhalb dieser Gruppen weitgehend alphabetisch sortiert. Einen bestimmten Befehl finden Sie am einfachsten über den Index. Die Optionen der Befehle sind zunächst nach ihren Funktionen und dann weitestgehend nach Wichtigkeit angeordnet. Weniger wichtige Optionen werden nicht immer aufgeführt. Einige Befehle bieten derart zahlreiche und umfassende Optionen, dass es aufgrund des kompakten Buchformats nicht möglich ist, alle aufzuführen und zu erläutern.

Viele der in diesem Buch beschriebenen Befehle lassen sich unter allen gegenwärtig eingesetzten Windows-Versionen verwenden, auch wenn sie nur mit einem Vorgänger oder Nachfolger des von Ihnen eingesetzten Betriebssystems mitgeliefert werden. Einige Tools benötigen jedoch zwingend ein bestimmtes Serverbetriebssystem oder eine Mindestversion des Betriebssystems, weil sie API-Funktionen verwenden, die von Microsoft nicht für alle Versionen des Betriebssystems bereitgestellt wurden. Andere Befehle wurden von Microsoft immer weiter verbessert und verändert, sodass z. B. die Syntax und der Funktionsumfang des mit Windows 10 mitgelieferten Tools anders sind als jene des gleichnamigen Befehls in einer Vorgängerversion. In solchen Fällen wird die aktuellste Version beschrieben.

Konventionen

`Fett`
: Kennzeichnet Windows-Befehle und -Optionen.

`GROSSBUCHSTABEN UND FETT`
: Kennzeichnet interne Befehle des Kommandozeileninterpreters *cmd.exe*. Diese Befehle sind bei Nutzung einer alternativen Shell, z. B. in PowerShell, nicht unmittelbar verfügbar.

Kursiv
: Kennzeichnet Parameter, die Sie selbst eingeben müssen.

[...]
: Kennzeichnet optionale Befehlsteile.

a | *b*
: Bedeutet, dass entweder *a* oder *b* eingesetzt werden kann.

{a | b}
: Bedeutet, dass entweder *a* oder *b* eingesetzt werden muss.

HKLM
HKCU
: Kennzeichnen die Registrierungsbäume (Hives) *HKEY_LOCAL_MACHINE* und *HKEY_CURRENT_USER*.

Menüname → Menüname
: Der Pfeil (→) in Verbindung mit kursiver Schrift beschreibt die Navigation innerhalb eines Menüs.

PoSh:
: Hinweise zu PowerShell-Alternativen

Die Eingabeaufforderung

Starten unter Windows 10/Server 2016

Nachdem unter Windows 8/Server 2012 die seinerzeit eingeführte Benutzeroberfläche Anwender begeistert oder abgeschreckt hat, ist Microsoft mit den Nachfolgern einen Schritt zurück in Richtung Startmenü gerudert. In den aktuellen Versionen von Windows-Client und -Server ist die Eingabeaufforderung in ihrer vollen Funktionalität enthalten, allerdings versteckt im Startmenü unter *Alle Programme → Windows-System*. Am schnellsten erreichen Sie sie, indem Sie im Startmenü von Windows 10 oder Server 2016 einfach drauflostippen und entweder `cmd` oder `Eingabeaufforderung` eingeben. Sobald Windows fündig geworden ist, können Sie das Symbol zur unmittelbaren Ausführung anklicken. Diese Variante hat einen in vielen Situationen entscheidenden Nachteil, denn die Eingabe-

aufforderung wird in diesem Fall mit den eingeschränkten Berechtigungen eines Standardbenutzers aufgerufen. Da diverse in diesem Buch beschriebene Befehle Administratorrechte benötigen (und seit Windows Vista/Server 2008 zudem bei standardmäßig aktivierter Benutzerkontenkontrolle die hohe Verbindlichkeitsstufe erforderlich ist), reicht der Benutzerkontext oft nicht.

Klicken Sie also stattdessen das gefundene Symbol der Eingabeaufforderung mit der rechten Maustaste an und dann mit der linken Maustaste auf *Mehr → Als Administrator ausführen*, um ein Fenster der Eingabeaufforderung mit erhöhten Rechten zu öffnen. Alternativ funktioniert nach wie vor die Tastenkombination *Strg+Umschalt+Enter* bei ausgewähltem Symbol sowie *Alt+J* zum Bestätigen der Sicherheitsabfrage.

Soll häufig auf die Eingabeaufforderung zurückgegriffen werden, ist diese Methode nicht sonderlich praktikabel. Es empfiehlt sich das Anlegen einer Verknüpfung. Und wo sollte diese liegen? Windows 10 bietet bei einem Rechtsklick die Erstellung einer Verknüpfung im Startmenü (*An Start anheften*) oder in der Taskleiste des Desktops (*An Taskleiste anheften*) an, wobei Letzteres erfahrungsgemäß die praktischere Variante ist. Durch einen Rechtsklick auf das angeheftete Symbol, den nochmaligen Rechtsklick auf den Menüeintrag *Eingabeaufforderung* und im folgenden Kontextmenü per Klick *Eigenschaften* können Sie einige Einstellungen für verbesserten Komfort anpassen:

Ausführen in:
: Hier können Sie einen alternativen Pfad eingeben, in dem sich die Eingabeaufforderung öffnen soll, beispielsweise *%WINDIR%\system32*, oder einen anderen Pfad, in dem sich zusätzliche Skriptdateien oder zu bearbeitende Dateien befinden.

Tastenkombination:
: Hier können Sie eine Tastenkombination festlegen, mit der die Eingabeaufforderung gestartet werden kann. Verwenden Sie dafür keine schon anderweitig im System oder in ständig laufenden Anwendungsprogrammen verwendete Kombination, um unerwartete Auswirkungen zu verhindern.

Durch einen Klick auf die Schaltfläche *Erweitert* öffnet sich das Dialogfeld *Erweiterte Eigenschaften*, in dem Sie über das Kontrollkästchen *Als Administrator ausführen* festlegen können, dass bei jeder Ausführung der Verknüpfung die Eingabeaufforderung mit erhöhten Rechten geöffnet wird.

Weitere Einstellungsmöglichkeiten betreffen unter anderem Schriftart und -größe, den verfügbaren Puffer zur Zwischenspeicherung von Befehlen und die Fenstergröße. Eine Eingabeaufforderung im Vollbildmodus steht schon seit mehreren Windows-Versionen nicht mehr zur Verfügung, allerdings kommen Sie mit der Tastenkombination *Alt+Enter* dieser inzwischen wieder recht nahe.

Ganz klassisch können Sie eine Verknüpfung auf dem Desktop erzeugen, indem Sie mit der rechten Maustaste auf eine beliebige freie Stelle des Desktops klicken, aus dem Kontextmenü den Eintrag *Neu* → *Verknüpfung* wählen, als *Speicherort des Elements* `cmd` eingeben und einen beliebigen Namen festlegen. Nach Beendigung des Assistenten können Sie die Eigenschaften der Verknüpfung bearbeiten, wie es oben für das an die Taskleiste angeheftete Symbol beschrieben wurde.

Schließlich können Sie unter Windows 10/Server 2016 die Tastenkombination *Win+X* verwenden und aus dem dann auf dem Desktop links unten erscheinenden Menü den Eintrag *Eingabeaufforderung (Administrator)* mit der Maus auswählen. Gegebenenfalls wird dort *Windows PowerShell* angezeigt. Um das zu ändern, klicken Sie mit der rechten Maustaste auf einen freien Bereich der Taskleiste, wählen den Eintrag *Taskleisteneinstellungen* und setzen dort den Schieberegler für den Eintrag *Beim Rechtsklick auf die Schaltfläche „Start" oder beim Drücken von Windows-Taste+X „Eingabeaufforderung" im Menü durch „Windows PowerShell" ersetzen* auf den gewünschten Wert.

Eine aus Sicherheitsgründen wenig empfehlenswerte Alternative zur generellen Ausführung der Eingabeaufforderung und aller anderen Anwendungen mit erhöhten Rechten ist die Deaktivierung der Benutzerkontensteuerung des Betriebssystems bei gleichzeitiger Verwendung eines Benutzerkontos der Administratorengruppe.

Sie können jederzeit nachsehen, ob ein Befehlszeilenfenster mit erhöhten Rechten ausgeführt wird: In der Regel beginnt die Titelzeile des Fensters in diesem Fall mit *Administrator:*.

Eingabe von Befehlen

Befehle gibt man ein, indem man sie in die Befehlszeile tippt und mit der *Enter*-Taste die Ausführung auslöst. Dabei kann man in der Regel nicht viel falsch machen, abgesehen von Syntaxfehlern, Tippfehlern und der unbeabsichtigten Verwendung von nicht zur Aufgabenstellung passenden Befehlen. Dennoch sind Befehlseingaben in ihren Grundanforderungen nicht immer konsistent, sodass es sich lohnen kann, im Fehlerfall die Hilfe zum jeweiligen Befehl etwas ausführlicher zu lesen:

- Bei der Eingabe von Befehlen ist es Ihnen freigestellt, ob Sie Groß- oder Kleinbuchstaben verwenden.
- Befehle und Befehlsoptionen können normalerweise in Groß- oder Kleinbuchstaben eingegeben werden. Ist das der Fall, werden sie in diesem Buch kleingeschrieben. Nur Optionen, die großgeschrieben werden *müssen*, werden in Großbuchstaben geschrieben.
- Befehlsoptionen werden normalerweise durch einen Schrägstrich eingeleitet: `/x`. In vielen Fällen kann der Schrägstrich durch ein Minuszeichen ersetzt werden. Einige Befehle akzeptieren nur das Minuszeichen.
- Die Reihenfolge der Optionen ist nicht einheitlich und nicht immer beliebig. Bitte entnehmen Sie die korrekte Reihenfolge der Syntax des jeweiligen Befehls.
- Einzelne Parameter werden abhängig vom Befehl durch Leerzeichen, Kommata oder Semikola voneinander getrennt.
- Parameter mit Leerzeichen, beispielsweise Verzeichnispfade und Dateinamen, müssen in der Regel von Anführungszeichen umschlossen werden.
- Regions- und Spracheinstellungen sowie die Sprachversion des Betriebssystems können die Syntax beeinflussen. In diesem

Buch wird die deutschsprachige Version des Betriebssystems mit den Einstellungen für Deutschland verwendet.

- Befehle können in der nachfolgenden Zeile fortgesetzt werden, wenn die vorherige Zeile mit dem ^-Zeichen beendet wurde.
- Durch ein vorangestelltes ^-Zeichen wird zudem verhindert, dass der Befehlsinterpreter das folgende Zeichen interpretiert. Solche Zeichen werden auch »Escapezeichen« genannt.
- Mehrere Befehle können mit dem &-Zeichen verknüpft werden:

  ```
  Befehl1 & Befehl2
  ```

  Die Befehle werden der Reihe nach ausgeführt.
- Die Ausführung eines Befehls kann davon abhängig gemacht werden, ob der vorangegangene Befehl erfolgreich ausgeführt wurde. Dazu werden die Befehle mit && bzw. || verknüpft:

  ```
  Befehl1 && Befehl2
  ```

  Befehl2 wird nur ausgeführt, wenn *Befehl1* erfolgreich ausgeführt werden konnte.

  ```
  Befehl1 || Befehl2
  ```

  Befehl2 wird nur ausgeführt, wenn *Befehl1* nicht erfolgreich ausgeführt werden konnte.

Umleitung der Ein- und Ausgabe

<Datei
: Liest Standardeingabe aus einer Datei statt von der Tastatur.

>Datei
1>Datei
: Schreibt Standardausgabe in eine Datei statt auf den Bildschirm.

>>Datei
1>>Datei
: Hängt Standardausgabe an eine Datei an.

2>Datei
: Schreibt Standardfehlerausgabe in eine Datei.

2>>Datei
: Hängt Standardfehlerausgabe an eine Datei an.

>Datei 2>&1
: Schreibt Standardausgabe in eine Datei und leitet Standardfehlerausgabe zur Standardausgabe um. Damit werden Standardausgabe und Standardfehlerausgabe in dieselbe Datei geschrieben.

Befehl1 |Befehl2
Befehl1 0>Befehl2
: Stellt eine Verknüpfung zwischen der Standardausgabe des ersten Befehls (*Befehl1*) und der Standardeingabe des zweiten Befehls (*Befehl2*) her.

  In Einzelfällen, beispielsweise für die Wiederverwendung von komplexen Werten wie GUIDs zur Vermeidung von Tippfehlern, ist es hilfreich, den gewünschten Ausschnitt der Eingabeaufforderung in die Zwischenablage zu kopieren und von dort entweder direkt oder nach Bearbeitung im Editor wieder einzufügen. Dafür können Sie auf althergebrachte Art in der Titelzeile des Fensters der Eingabeaufforderung das Systemmenü oben links nutzen, also auf *Bearbeiten* → *Markieren* klicken, und dann mit der Maus oder mit der Tastatur unter Zuhilfenahme der Pfeiltasten und der gedrückten *Umschalt*-Taste den zu kopierenden Bereich in Rechteckform auswählen. Sobald Sie die Auswahl abgeschlossen haben, drücken Sie die *Enter*-Taste, um den Text des ausgewählten Bereichs in die Zwischenablage zu kopieren.

  Zum Einfügen aus der Zwischenablage verwenden Sie wiederum den Menüeintrag *Bearbeiten* → *Einfügen* aus dem Systemmenü.

  Oder aber Sie nutzen die aus vielen anderen Windows-Anwendungen vertrauten Methoden mit Tastatur oder Maus, die in den aktuellsten Windows-Versionen auch endlich in der Eingabeaufforderung verfügbar sind, zum Beispiel:

  Umschalt+Pfeil links/Pfeil rechts: Markiert Zeichen links/rechts von der Cursorposition (jedes erneute Betätigen der Pfeiltaste markiert ein weiteres Zeichen).

Umschalt+Strg+Pfeil links/Pfeil rechts: Markiert nach Markierung eines Zeichens Wörter links/rechts von der Cursorposition (jedes erneute Betätigen der Pfeiltaste markiert ein weiteres Wort).

Strg+C: Kopiert die markierten Zeichen in die Zwischenablage.

Strg+V: Fügt Inhalte aus der Zwischenablage an der Cursorposition ein.

Ziehen mit gedrückter linker Maustaste: Markiert den ausgewählten Bereich.

Rechtsklick: Kopiert einen markierten Bereich als Text in die Zwischenablage oder fügt den Inhalt der Zwischenablage an der Cursorposition ein.

Umgebungsvariablen

Dieser Abschnitt erklärt einige wichtige Windows-Umgebungsvariablen. Variablen werden in Eingabeaufforderung und Batchdateien mit dem Prozentzeichen (%) ausgelesen, zum Beispiel zeigt der Befehl `echo %SystemRoot%` den Pfad des Windows-Ordners. Damit können Anwendungen und Skripte auf unterschiedlich eingerichteten Systemen ausgeführt werden, ohne dass eine abweichende Ordnerstruktur für die Aufrufe manuell berücksichtigt werden müsste. Die aktiven System- und Benutzervariablen können mit `set` angezeigt und temporär angepasst sowie mit `setx` dauerhaft geändert werden.

PoSh: In PowerShell befinden sich die Umgebungsvariablen im Container `env:`. `gci env:systemroot` zeigt die obige Variable.

COMPUTERNAME
: Enthält den Computernamen der Windows-Installation.

HOMEDRIVE und HOMEPATH
: Das Laufwerk und der Pfad auf diesem Laufwerk zum Benutzerprofilordner des angemeldeten Benutzers.

PATH
: Mehrere durch Semikola getrennte Verzeichnisse, die in dieser Reihenfolge nach Befehlen durchsucht werden, wenn diese ohne vorangestellten Pfad aufgerufen werden.

PATHEXT
: Mehrere durch Semikola getrennte Dateierweiterungen, die in dieser Reihenfolge an einen Befehl ohne Erweiterung angehängt werden, um die ausführbare Datei für den Befehl zu finden. Diese Erweiterungen müssen den voranstehenden Punkt beinhalten, also beispielsweise *.exe* oder *.cmd*.

ProgramFiles
: Der Verzeichnisname des Programmordners (normalerweise *C:\Program Files*). Bei 64-Bit-Versionen des Betriebssystems handelt es sich um den Programmordner für 64-Bit-Software.

ProgramFiles(x86)
: Der Verzeichnisname des Programmordners für 32-Bit-Anwendungen in einem 64-Bit-Betriebssystem.

SystemRoot und windir
: Der Verzeichnisname des Windows-Systemverzeichnisses (normalerweise *C:\WINDOWS*).

TEMP und TMP
: Der komplette Pfad eines Verzeichnisses, das von Anwendungen und vom Betriebssystem zur Ablage temporärer Dateien verwendet wird.

USERNAME
: Der Name des angemeldeten Benutzers.

USERPROFILE
: Der Pfad zum Profilverzeichnis des angemeldeten Benutzers.

PoSh: Einer der Vorteil der PowerShell ist, dass mit Platzhaltern gearbeitet werden kann. So zeigt `gci env:*path*` alle Variablen an, die das Wort `path` im Namen enthalten.

Installation zusätzlicher Administrationstools

Bereits seit Vista enthalten die Installationsmedien von Windows keine Support-Tools mehr, da sie eingestellt wurden.

Microsoft hat zudem die Praxis aufgegeben, Zusatztools über Resource Kits zugänglich zu machen. Stattdessen werden nützliche

Programme direkt veröffentlicht, beispielsweise die Werkzeuge der Sysinternals-Suite. Auf einige dieser Programme wird im Buch eingegangen.

Im Unternehmensumfeld sind zudem die »Remoteserver-Verwaltungstools« (*Remote Server Administration Tools*, RSAT) zur Verwaltung von Servern von Bedeutung. Sie dienen der Verwaltung von Servern. Unter Server 2016 sind sie als Feature im Dashboard oder mittels PowerShell installierbar, für die Nutzung unter Windows 10 müssen sie zunächst von Microsoft heruntergeladen und auf herkömmliche Art und Weise installiert werden. Berücksichtigen Sie, dass die Werkzeuge wählerisch sind – so lassen sich die RSAT für Server 2016 beispielsweise nicht unter Windows 7 installieren – auch neuere Builds von Windows Server erfordern teilweise angepasste RSAT-Tools.

Für die Unterstützung der automatisierten Installation von aktuellen Windows-Versionen in Unternehmen stellt Microsoft das *Windows Assessment and Deployment Kit* zum Herunterladen bereit, das ebenfalls einige nützliche Werkzeuge für die Eingabeaufforderung enthält.

Hilfebefehle und -dateien

`help` ***`Befehl`***
: Zeigt die Hilfe für viele Windows-Standardbefehle an.

`Befehl` `/?`
: Zeigt bei den meisten ausführbaren Dateien einen Hilfetext an.

`net help` ***`Befehl`***
: Zeigt die Hilfe für einen der `net`-Befehle an. Beispielsweise erklärt `net help user` die Optionen des Befehls `net user`.

`net helpmsg` ***`nnnn`***
: Zeigt den Fehlertext zum Windows-Fehler mit der Nummer `nnnn` an.

Wie schon in den Vorgängerversionen von Windows fehlt eine GUI-basierte deutschsprachige Hilfe zur Eingabeaufforderung bei-

spielsweise über *Hilfe und Support*. Auch diesbezügliche Onlineangebote durch Microsoft werden kaum noch lokalisiert.

Windows Assessment and Deployment Kit

Zum Teil sind Informationen zur Verwendung der zusätzlichen Werkzeuge nur in englischer Sprache verfügbar und müssen eventuell online recherchiert werden. Auch hier kann zumindest zu den Befehlszeilenwerkzeugen oftmals eine rudimentäre Hilfe mittels `/?` hinter dem Befehl aufgerufen werden.

Windows PowerShell

Seit PowerShell im Jahr 2007 erstmals veröffentlicht wurde, ist ihre Integration in Microsoft-Betriebssysteme und -anwendungen weit vorangeschritten. In Windows Server 2016 und Windows 10 ist aktuell Version 5.1 der PowerShell integriert. Bei PowerShell handelt es sich um eine Skriptumgebung, die sowohl die klassische Eingabeaufforderung als auch früher beliebte Skriptsprachen wie VBScript beerben sollte und dies mittlerweile gerade im Serverumfeld auch recht erfolgreich tut.

PowerShell arbeitet nicht textbasiert wie andere eingabezeilenorientierte Benutzeroberflächen, sondern gibt Objekte über Pipelines (`|`) von einem Befehl zum anderen weiter. Die Benennung der als »Cmdlets« bezeichneten Befehle folgt dem Schema Verb-Nomen, z.B. `get-command` (Auflistung aller Befehle). PowerShell verwendet ein erweiterbares Providermodell, um neben dem Dateisystem folgende Datenspeicher als Laufwerk anzusprechen: Registrierung, Zertifikatsspeicher, Umgebungsvariablen, Aliase, Variablen und Funktionen.

Jede neue Version von PowerShell brachte eine erhebliche Steigerung des Funktionsumfangs mit sich. Intellisense und Tab-Vervollständigung helfen bei der Erkundung und Eingabe von Befehlen. Eine Hilfe, die online aktualisiert werden kann, steuert zur Funktionalität bei.

Bitte beachten Sie in gemischten Betriebssystemumgebungen, dass erst mit PowerShell 5.1 eingeführte Befehle nur unter Windows 10/Server 2016 zur Verfügung stehen.

Um zu ermitteln, welche Version der PowerShell aktuell auf Ihrem Rechner installiert ist, führen Sie innerhalb von PowerShell den folgenden Befehl aus:

```
if (test-path variable:psversiontable) {$psversiontable.psversion}
else {[version]"1.0.0.0"}
```

Die Ausgabe zeigt in der Spalte *Major* die Versionsnummer an.

Auch wenn es nach wie vor Anwendungsfälle gibt, in denen PowerShell im Vergleich zu den spezialisierten Befehlen der Eingabeaufforderung zu komplex ist, um mal schnell ein bestimmtes Ziel zu erreichen, gibt es nur wenige bislang in den Bereich der Eingabeaufforderung fallende Aufgaben, die sich mit PowerShell nicht lösen ließen.

Ergänzend bietet die PowerShell zahllose Möglichkeiten, die mit den Standardwerkzeugen der Eingabeaufforderung nicht und auch in anderen Skriptsprachen häufig nicht trivial umzusetzen sind. Da zudem fast alle Befehle der Eingabeaufforderung – mit Ausnahme der Parameterübergabe für interne Befehle – unmittelbar in der PowerShell-Konsole funktionieren, besteht eigentlich kaum mehr ein Grund, noch aus der Eingabeaufforderung heraus zu arbeiten. Lediglich die ISE (*Integrated Scripting Environment*, integrierte Skriptumgebung) weist bei der Unterstützung von einzelnen Befehlen der Eingabeaufforderung noch Mankos auf. Die dafür dort integrierte umfassende Hilfe unterstützt den Anwender dabei, die anfangs steil erscheinende Lernkurve beim Erlernen der PowerShell-Befehls- und Skriptsprache zu bewältigen.

Im weiteren Verlauf dieses Buchs werden zu den Befehlen der Eingabeaufforderung teilweise Alternativen in Form des Einsatzes der PowerShell oder einer Kombination aus PowerShell und Befehlszeile gezeigt. Praxisnahe Beispiele vermitteln, dass PowerShell ihre Stärken auch in Alltagssituationen voll zur Geltung bringen kann. Auf PowerShell bezogene Abschnitte werden mit dem Kürzel **PoSh** gekennzeichnet. Die aufgeführten Beispiele und Hinweise dienen weniger dazu, perfekt zu sein, stattdessen sollen sie die vielfältigen Möglichkeiten der Problemlösung mit PowerShell nahebringen und

beim selbstständigen Finden und Ausprobieren dafür relevanter Cmdlets helfen. Zudem gibt es bei der Verwendung von PowerShell meist nicht nur den einen Weg, ein Ziel zu erreichen.

Allgemeine Befehle

clip

```
Befehl | clip
clip < Datei
```

Der Befehl `clip` ermöglicht die Umlenkung der Bildschirmausgabe eines Befehls in die Zwischenablage, um deren Inhalt in eine Windows-Anwendung einzufügen.

PoSh: Mit `clip` lassen sich auch PowerShell-Bildschirmausgaben einfach als Text in die Zwischenablage kopieren.

cmd

```
cmd [Optionen] [[/c | /k] [/s] Befehl]
```

Startet eine neue Instanz des Windows-Befehlsinterpreters. Wurde ein Befehl angegeben, wird er ausgeführt. Verwenden Sie die mit `cmd /?` aufrufbare ausführliche Hilfe, um Informationen zu weiteren Features zu erhalten (z.B. zur automatischen Vervollständigung von Pfaden und Befehlen oder zur verzögerten Expansion von Variablen). Über die Eigenschaften eines Kommandozeilenfensters (Klick auf das Systemmenü in der linken Ecke des Titelfensters und dann auf *Eigenschaften*) können Sie sein Aussehen und Verhalten in weiten Bereichen beeinflussen, wobei diese Anpassung im Gegensatz zu den Eigenschaften der Verknüpfung nur für das aktive Fenster gilt. Insbesondere die Vergrößerung des Fensterpuffers, der das Scrollen in den nicht mehr am Bildschirm sichtbaren Bereich ermöglicht, ist oft sinnvoll.

Der Befehl `exit` beendet den Kommandozeileninterpreter.

Optionen

`[/c | /k] [/s]`
: Der Interpreter führt den angegebenen Befehl aus und bleibt nach Beendigung des Befehls aktiv (`/k`) oder beendet sich (`/c`). Die Option `/s` veranlasst den Befehlsinterpreter, den Befehl umschließende Anführungszeichen vor Ausführung des Befehls zu entfernen (normalerweise werden diese beibehalten). Das gilt jeweils in Verbindung mit `/c` oder `/k`.

`/q`
: Schaltet die Befehlsausgabe ab (siehe `echo off`).

`/e:{`*`on`*` | `*`off`*`}`
: Aktiviert oder deaktiviert die Erweiterungen des Befehlsinterpreters. Der Standardwert wird durch den Registrierungswert *\Software\Microsoft\Command Processor\Enable Extensions* unter *HKCU* oder *HKLM* bestimmt. Im Auslieferungszustand sind die Erweiterungen aktiviert.

`/a | /u`
: Die Ausgabe von internen Befehlen erfolgt im ANSI-(Standard-) bzw. Unicode-Format.

`/d`
: Deaktiviert die Autorun-Einträge in der Registrierung unter *\Software\Microsoft\Command Processor\Autorun* in *HKLM* und *HKCU*.

`/f:{`*`on`*` | `*`off`*`}`
: Aktiviert bzw. deaktiviert die Ergänzung von Datei- und Verzeichnisnamen mit der *Tab*-Taste.

Weitere Parameter entnehmen Sie dem Aufruf von `cmd /?` in einem offenen Befehlszeilenfenster.

```
cmd /u /? /c > c:\temp\help.txt
```

Leitet unter Verwendung des Unicode-Formats (womit deutsche Sonderzeichen in der erzeugten Datei in Windows-Anwendungen wie Notepad korrekt dargestellt werden) die Hilfeinformationen zum Befehlszeileninterpreter in die Datei *help.txt* im gegebenenfalls vorher anzulegenden Ordner *c:\temp* um.

PoSh: Da interne Befehle des Befehlszeileninterpreters wie beispielsweise `DIR` in der PowerShell nur nachgebildet sind und diese Nachbildungen gebräuchliche Parameter nicht unterstützen, kann durch expliziten Aufruf von `cmd` in der PowerShell-Konsole auf das Original des Befehls zurückgegriffen werden:

```
cmd /c dir /p
```

Gibt den Inhalt des aktuellen Verzeichnisses in einer PowerShell-Konsolensitzung mit dem klassischen `DIR`-Befehl seitenweise am Bildschirm aus.

command

(nur in 32-Bit-Betriebssystemen)

`command [[Laufwerk:]Pfad][Gerät][e:nnnnn] [[/p | /c *Befehl*] [/MSG]]`

Startet eine neue Instanz der MS-DOS-Eingabeaufforderung. Da es sich dabei um eine 16-Bit-Instanz handelt, ist dieser Befehl nicht Bestandteil der 64-Bit-Versionen von Windows. Der 16-Bit-Befehlsinterpreter ist in 32-Bit-Versionen von Windows nur aus Gründen der Kompatibilität zu sehr alten MS-DOS-Anwendungen noch enthalten.

COLOR

`color *Farbcode1 Farbcode2*`

Stellt die Hintergrundfarbe und die Schriftfarbe in der aktuell geöffneten Konsole um. Die Farbattribute sind als hexadezimale Werte anzugeben, die durch die Hilfe angezeigt werden. So stellt `color 1f` den Hintergrund auf Blau und die Schrift auf Weiß ein. Ohne Angabe von Farbcodes wird die Standardanzeige wiederhergestellt.

PoSh: Auch die Farbgebung der PowerShell-Konsole lässt sich mit dem Befehl `cmd /c color xy` anpassen. Alternativ lassen sich folgende PowerShell-Befehle verwenden:

```
$HOST.UI.RawUI.BackgroundColor = "DarkBlue"
$HOST.UI.RawUI.ForegroundColor = "White"
```

date

`date [tt.mm.jj]jj] [/t]`

Stellt das angegebene Datum ein oder fragt danach, wenn es nicht angegeben wurde. Mit der Option `/t` wird das Datum angezeigt, ohne es zu ändern, was sich mit Ausgabeumleitung für selbst erstellte Logdateien eignet.

PoSh: `get-date -displayhint date` zeigt das Systemdatum ohne Uhrzeit. Um das Datum in Kurzform angezeigt zu bekommen, verwenden Sie `get-date -format dd.MM.yyyy`.

Das Cmdlet `set-date` erlaubt hingegen, ein neues Systemdatum zu setzen. Beachten Sie, dass dabei die Uhrzeit auf 00:00 gesetzt wird, sofern sie nicht explizit angegeben wird.

doskey

`doskey [Optionen]`

Erlaubt den Zugriff auf bereits ausgeführte Befehle mit den Tasten *Pfeil hoch* und *Pfeil runter* und die Erstellung von Makros (Aliasdefinitionen).

Befehlshistorie und Editieroptionen

`/history`
: Zeigt die vollständige Befehlshistorie an. Durch Ausgabeumleitung mit > gefolgt von einem Dateinamen können Sie diese Liste für die künftige Verwendung als Makrodatei abspeichern.

`/listsize=n`
: Stellt die Größe der Befehlshistorie auf *n* Einträge ein.

`/insert | /overstrike`
: Stellt den Bearbeitungsmodus für aus der Historie abgerufene Befehle auf *Einfügen* bzw. auf *Überschreiben*. Die Standardeinstellung ist *Einfügen*. Durch Betätigen der Taste *Einfg* können Sie den Modus ebenfalls wechseln.

Makrooptionen

`Makroname=Befehl`
: Definiert ein Makro. Innerhalb des Befehls können die folgenden Variablen verwendet werden: `$T` fügt ein Trennzeichen ein, `$1` bis `$9` erlauben den Zugriff auf einzelne Parameter, `$*` fügt alle eingegebenen Parameter ein. `$G` ersetzt das Zeichen für die Ausgabeumleitung:

```
doskey liste=dir /b $1 $g dateien.txt $t notepad.exe
dateien.txt
```

`/macros`
: Zeigt alle vorhandenen Makros an.

`/macros:all`
: Zeigt zusätzlich die ausführbaren Dateien zugewiesenen doskey-Makros an.

`/macrofile=`*`Datei`*
: Aktiviert alle in der angegebenen Datei enthaltenen Makros.

`/exename=`*`Exe-Datei`*
: Erlaubt die Zuordnung einer ausführbaren Datei zu dem Makro, das soeben definiert wird.

`/macros:Exe-Datei`
: Zeigt alle vorhandenen Makros an, die der angegebenen ausführbaren Datei zugeordnet sind.

Die Tastenkombination *Alt+F10* löscht alle definierten Makros.

PoSh: Standardmäßig werden doskey-Makros dem Befehlszeileninterpreter *cmd.exe* zugeordnet. Mithilfe des Parameters `/exename=powershell.exe` können Sie doskey-Makrodateien für die Ausführung von PowerShell-Befehlen verwenden. Dabei beachten Sie bitte, dass die Syntax der Befehle den Anforderungen der PowerShell entsprechen muss – so wäre z. B. die Variable `%WINDIR%` durch den PowerShell-Ausdruck `$env:WINDIR` zu ersetzen.

find

`find [`*`Optionen`*`] "`*`Zeichenfolge`*`" [`*`Dateien`*`]`

Sucht in den angegebenen Dateien, in einem über die Tastatur eingegebenen Text oder über eine Pipeline in der Standardeingabe

nach der von Anführungszeichen umschlossenen Zeichenfolge und gibt Zeilen bzw. deren Anzahl aus, die die Zeichenfolge enthalten.

Eine beliebte Anwendung von `find` ist die Filterung der Ausgabe eines anderen Befehls. Das folgende Beispiel gibt unter Windows 10 diejenigen Verzeichnisse innerhalb des Benutzerprofils aus, die als Junction auf ein anderes Verzeichnis zeigen:

```
dir /ad %userprofile% | find /i "junction"
```

Optionen

`/v`
: Zeigt nur die Zeilen an, in denen die Zeichenfolge nicht vorkommt.

`/i`
: Ignoriert Groß-/Kleinschreibung beim Vergleich.

`/c`
: Zeigt nur die Anzahl der übereinstimmenden Zeilen an.

`/n`
: Zeigt vor jeder Zeile die Zeilennummer an.

`/offline`
: Schließt bei der Suche in Dateien auch Offline-Dateien ein.

findstr

`findstr [`*`Optionen`*`] [/c:`*`Zeichenfolge`*` | /g:`*`Datei`*` | `*`Zeichenfolgen`*`] [`*`Dateien`*`]`

Sucht in den angegebenen Dateien nach einer oder mehreren Zeichenfolgen oder regulären Ausdrücken und gibt übereinstimmende Zeilen aus. Wurden keine Dateien angegeben, wird die Standardeingabe durchsucht. Falls Sie mehrere Suchbegriffe verwenden wollen, müssen Sie sie gemeinsam in Anführungszeichen einschließen.

Optionen

`/r`
: Interpretiert die Zeichenfolge als regulären Ausdruck.

`/l`
: Interpretiert die Zeichenfolge buchstabengetreu.

/c:*Zeichenfolge*
: Kennzeichnet die angegebene Zeichenfolge als Suchbegriff. Diese Option ist insbesondere bei Leerzeichen im Suchbegriff hilfreich, da ohne sie jedes Wort der Zeichenfolge einzeln gesucht wird.

/g:*Datei*
: Liest die Suchausdrücke aus der angegebenen Datei. Ein Schrägstrich anstelle des Dateinamens bedeutet, dass der Dateiname in der Eingabeaufforderung abgefragt wird.

/b | /e
: Übereinstimmende Zeilen werden nur dann ausgegeben, wenn die Übereinstimmung am Anfang (`/b`) oder am Ende (`/e`) der Zeile auftritt. Es kann nur eine dieser beiden Optionen verwendet werden.

/i
: Vergleicht ohne Berücksichtigung von Groß-/Kleinschreibung.

/v
: Zeigt nicht übereinstimmende Zeilen an.

/x
: Zeigt nur exakt übereinstimmende Zeilen an.

/n | /o
: Zeigt die Zeilennummer (`/n`) oder die Anzahl der Zeichen vom Dateianfang bis zur Übereinstimmung (`/o`) für jede Fundstelle an.

/m
: Zeigt nur die Namen der Dateien mit gefundener Übereinstimmung.

/s
: Durchsucht die angegebenen Dateien auch in Unterverzeichnissen.

/f:*Datei*
: Liest die Dateiliste aus der angegebenen Datei. Ein Schrägstrich anstelle des Dateinamens bedeutet, dass der Dateiname in der Eingabeaufforderung abgefragt wird.

/d:*Verzeichnisliste*
: Durchsucht die Dateien in der angegebenen Verzeichnisliste. Verzeichnisse werden durch Semikola voneinander getrennt.

/p
: Dateien, die nicht druckbare Zeichen enthalten, werden übersprungen.

/offline
: Schließt bei der Suche in Dateien auch Offlinedateien ein.

Bestandteile von regulären Ausdrücken

.
: Ein beliebiges Zeichen.

^ | $
: Der Anfang bzw. das Ende einer Zeile.

\< | \>
: Der Anfang bzw. das Ende eines Worts.

\x
: Zeichen *x* verwenden, auch wenn es ein Metazeichen ist (z.B. bedeutet \$, dass das Dollarzeichen als Suchbegriff verwendet wird).

[Zeichenklasse]
: Ein beliebiges Zeichen aus einem Zeichensatz.

[^Zeichenklasse]
: Ein beliebiges Zeichen, das nicht im Zeichensatz enthalten ist.

[*a-z*]
: Ein beliebiges Zeichen aus dem angegebenen Bereich. Es können mehrere Bereiche und Listen von Zeichen in den Klammern angegeben werden.

*****
: Keines oder mehrere der Zeichen/Klassen aus der angegebenen Liste. Zum Beispiel bedeutet `[0-9]*` keine oder mehrere Zahlen, und `.*` bedeutet keines oder mehrere beliebige Zeichen.

PoSh: PowerShell verfügt über leistungsstarke Suchfunktionen. Folgendes Kommando durchsucht alle Dateien mit der Erweiterung *.log* in angegebenen Ordnern und Unterverzeichnissen und zeigt Dateinamen, Zeilennummer und Inhalt aller Zeilen mit der Zeichenkette »error«:

```
gci C:\Windows\system32\Logfiles\*.log -rec | select-string
-pattern "error"
```

Sollen lediglich die Namen derjenigen Dateien aufgeführt werden, in denen der Suchbegriff vorkommt, erweitern Sie den obigen Befehl mit:

```
-list | select-object filename
```

Dabei bewirkt der Parameter `-list` die Unterdrückung von Duplikaten; und da es sich bei den Ergebnissen von `select-string` nicht um bloßen Text, sondern um Objekte handelt, lässt sich aus diesen Objekten der Dateiname durch Weitergabe an `select-object` extrahieren.

---

more

***Befehl* | more [*Optionen*]**
more [*Optionen*] [*Dateien*]

Zeigt die Ausgabe eines Befehls oder die angegebene(n) Datei(en) seitenweise an. Wird häufig verwendet, um überlange Befehlsausgaben bequem lesen zu können.

Optionen

/e
: Aktiviert die erweiterten Features.

/c
: Löscht den Bildschirm, bevor die erste Seite angezeigt wird.

/s
: Zeigt statt mehrerer aufeinanderfolgender Leerzeilen nur eine an.

/t*n*
: Konvertiert Tabulatoren in *n* Leerzeichen. Der Standardwert für *n* sind acht Leerzeichen.

+*n*
: Beginnt mit der Anzeige in Zeile *n*.

`more` verwendet auch alle Optionen, die gegebenenfalls in der Umgebungsvariablen *MORE* gesetzt sind.

Die Hilfe beschreibt zusätzlich die an der Eingabeaufforderung »-- Fortsetzung --« akzeptierten Befehle zur Steuerung der Anzeige.

Mit der *Leertaste* blättern Sie eine Seite, mit der *Enter*-Taste eine Zeile vor. Mit der Tastenkombination *Strg+C* brechen Sie die Ausgabe ab. *P* gefolgt von einer Zahl *n* scrollt um *n* Zeichen weiter.

PoSh: PowerShell bietet als Äquivalent die Möglichkeit, die Ausgabe über den Pipelineoperator an den Befehl `out-host` weiterzureichen. Die Ausgabe des PowerShell-Befehls

```
get-childitem -rec | out-host -p
```

verhält sich grundsätzlich wie jene mit `more`: Durch Betätigung der *Leertaste* wird die Ausgabe um eine Bildschirmseite weitergeblättert, durch die *Enter*-Taste zeilenweise. Das funktioniert nicht in der ISE-Umgebung.

Durch den Befehl `more` werden umgekehrt auch PowerShell-Bildschirmausgaben akzeptiert.

PATH

`PATH` [*`Pfad`*]

Zeigt den Suchpfad (eine durch Semikola getrennte Verzeichnisliste, in der die Eingabeaufforderung nach aufgerufenen Programmen sucht) an oder verändert ihn. Die Umgebungsvariable `%path%` kann verwendet werden, um den aktuellen Suchpfad in einen veränderten Suchpfad einzufügen:

```
path %path%;C:\NeuerOrdner\
```

Ein mit dem Befehl `path` geänderter Suchpfad wird nicht gespeichert. Die Änderung gilt nur für die aktuelle Instanz des Befehlszeileninterpreters und von dieser aus aufgerufene Instanzen. Für dauerhafte Änderungen verwenden Sie den Befehl `setx`.

PoSh: `$env:path` zeigt den aktuellen Pfad an.

Hinweis: Das vorübergehende Ändern der Pfadvariablen mit PowerShell-eigenen Befehlen wird beim Befehl `set` erläutert.

set

set [/a] [/p] [Variable=[*Zeichenfolge*]]

Mit dem Befehl set lassen sich Variablen für das aktive Befehlszeilenfenster anzeigen, setzen und ändern. Wird das Fenster geschlossen, werden die Änderungen verworfen.

Variable
: Der Name der Variablen. Wird keine Variable angegeben, werden alle aktuellen Umgebungsvariablen angezeigt. Werden anstelle einer Variablen einzelne Zeichen oder Zeichenketten angegeben, werden diejenigen Variablen gezeigt, die mit diesen Zeichen beginnen.

Zeichenfolge
: Der Wert, den die Variable annehmen soll.

set currentpath="C:\Mein Programm;%PATH%"
: Setzt die Variable currentpath und verknüpft als deren Wert den Pfad *C:\Mein Programm* mit dem Inhalt der aktuellen PATH-Variablen.

/a
: Gibt an, dass die Zeichen rechts des Gleichheitszeichens einen numerischen Ausdruck darstellen. Sie müssen in Anführungszeichen eingeschlossen, numerisch und mit gültigen Operatoren verknüpft sein. Die verfügbaren Operatoren sind ausführlich in der Hilfe beschrieben, zum Beispiel:

```
set /a test="2*4"
```

/p
: Ermöglicht es, eine Variable durch eine Benutzereingabe zu definieren, zum Beispiel:

```
set /p keyword=
```

Mithilfe von set lässt sich so per Batchdatei eine speziell für eine Konsolenanwendung definierte Umgebung schaffen, die andere Bereiche von Windows nicht beeinflusst.

Tipp: Ausgabe einer Zeichenfolge mit vorangestelltem Zeitstempel:

Zunächst setzen Sie die Variable jetzt, die Datum, Uhrzeit und eine beliebige Zeichenfolge beinhaltet, mit folgendem Befehl: