

6 Konfigurieren von Microsoft Exchange Server mit der Exchange-Verwaltungsshell

Microsoft Exchange Server 2007 führt als neues Hilfsmittel für das er weiteste Aufgabefeld der Exchange Server-Administratoren und -Entwickler die Exchange-Verwaltungsshell ein. Dabei handelt es sich um eine erweiterungsfähige Befehlszeilenumgebung für Exchange Server 2007, die auf dem von der Windows PowerShell bereitgestellten Framework aufbaut. Wenn Sie Exchange Server 2007 auf einem Servercomputer oder die Exchange Server-Verwaltungstools auf einer Arbeitsstation installieren, werden dabei die Windows PowerShell sowie die Exchange-Verwaltungsshell gleich mitinstalliert. Dieses Kapitel stellt die Windows PowerShell und ihre Funktionen vor und erläutert die verfügbaren Befehle und Optionen der Exchange-Verwaltungsshell.

Verwenden der Windows PowerShell

Wenn Sie bereits mit Unix gearbeitet haben, ist der Begriff *Befehlsshell* für Sie vermutlich kein Fremdwort. Für die meisten auf Unix basierenden Betriebssysteme stehen mehrere komplett ausgestattete Befehlsshells zur Verfügung, z.B. die Korn-Shell (KSH), die C-Shell (CSH) und die Bourne-Shell (SH). Obwohl zu jedem Microsoft Windows-Betriebssystem stets auch eine Befehlszeilenumgebung gehörte, war bisher keine voll ausgestattete Befehlsshell darunter. Die Windows PowerShell soll dieses Manko beheben.

Einführung in die Windows PowerShell

Die Unix-Befehlsshells führen – ähnlich wie die weniger ausgefeilte Windows-Befehlszeilenumgebung – vordefinierte sowie externe Befehle und Befehlszeilendienstprogramme aus und geben die Ergebnisse in Textform als Ausgabestream zurück. Der Ausgabestream kann auf unterschiedliche

Weise manipuliert werden, u.a. kann er umgeleitet werden, um einem weiteren Befehl als Eingabe zu dienen. Die Umleitung der Ausgabe eines Befehls zur Eingabe eines anderen wird als *Piping* bezeichnet und kommt in Shellskripts häufig zum Einsatz.

Bei der C-Shell handelt es sich um eine der raffinierteren Unix-Shells. In vielerlei Hinsicht vereint sie einige der besten Funktionen der Programmiersprache C mit einer voll ausgestatteten Unix-Shellumgebung. Die Windows PowerShell baut die Idee einer vollständigen, auf einer Programmiersprache basierenden Befehlsshell weiter aus, indem sie eine von C# abgeleitete Skriptsprache und ein auf dem .NET Framework basierendes Objektmodell implementiert.

Dadurch, dass die Skriptsprache der Windows PowerShell auf C# basiert, wird gewährleistet, dass die Skripts für C#-Entwickler leicht verständlich sind und neue Entwickler für C# gewonnen werden. Die Verwendung eines auf dem .NET Framework basierenden Objektmodells ermöglicht der Windows PowerShell, ganze Objekte mit allen ihren Eigenschaften als Ausgabe von einem Befehl zum nächsten weiterzugeben. Die Möglichkeit, Objekte umzuleiten, ist äußerst vielseitig und erlaubt eine weitaus dynamischere Manipulation eines Resultsets. Sie können beispielsweise nicht nur den Namen eines bestimmten Benutzers, sondern gleich das gesamte dazugehörige Benutzerobjekt erhalten. Die Eigenschaften dieses Benutzerobjekts können Sie dann je nach Notwendigkeit bearbeiten, indem Sie die jeweils benötigten Eigenschaften namentlich aufrufen.

Ausführen und Verwenden der Windows PowerShell

Um die Windows PowerShell aufzurufen, müssen Sie zunächst ein Eingabeaufforderungsfenster öffnen und an der Befehlszeile den Befehl **power-shell** eingeben. Sie können die Windows PowerShell beenden und zur Eingabeaufforderung zurückkehren, indem Sie **exit** eingeben.

Normalerweise wird Ihnen beim Start der Shell eine Nachricht wie die folgende angezeigt:

```
Windows Powershell  
Copyright (C) 2006 Microsoft Corporation. All rights reserved.
```

Sie können diese Nachricht unterdrücken, indem Sie die Shell wie folgt mit dem Parameter **-nologo** starten:

```
powershell -nologo
```

Unabhängig davon, wie Sie die Shell starten, sehen Sie daran, dass sich die Titelleiste der Eingabeaufforderung in **Eingabeaufforderung – power-shell** geändert hat, dass Sie sie verwenden.

Während des Starts der Shell werden *Benutzer- und Systemprofile* ausgeführt, um die Umgebung einzurichten. Bei der folgenden Liste handelt es sich um die Beschreibung der Profildateien in der Reihenfolge, in der sie ausgeführt werden:

1. **%AllUsersProfile%\Documents\PSConfiguration\profile.ps1** Ein für alle Benutzer ausgeführtes systemweites Profil. Der Systemadministrator konfiguriert über dieses Profil gebräuchliche Einstellungen für die Windows PowerShell.
2. **%AllUsersProfile%\Documents\PSH\Microsoft.PowerShell_profile.ps1** Ein für alle Benutzer ausgeführtes systemweites Profil. Der Systemadministrator konfiguriert über dieses Profil gebräuchliche Einstellungen für die Windows PowerShell.
3. **%UserProfile%\My Documents\PSConfiguration\profile.ps1** Ein nur für den aktuellen Benutzer ausgeführtes Profil. Einzelne Benutzer konfigurieren über dieses Profil gebräuchliche Einstellungen für die Windows PowerShell.
4. **%UserProfile%\My Documents\PSH\Microsoft.PowerShell_profile.ps1** Ein nur für den aktuellen Benutzer ausgeführtes Profil. Einzelne Benutzer konfigurieren über dieses Profil gebräuchliche Einstellungen für die Windows PowerShell.

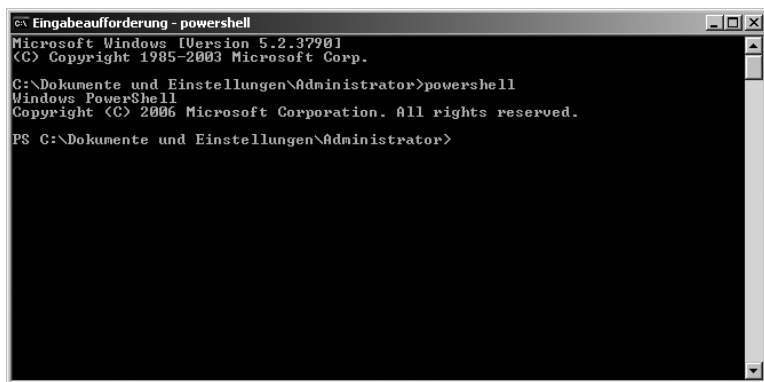


Abbildung 6.1: Starten der Windows PowerShell

Mithilfe des Parameters **-noprofile** können Sie die Windows PowerShell starten, ohne Profile zu laden:

```
msh-noprofile
```

Wenn Sie die Windows PowerShell zum ersten Mal aufrufen, wird Ihnen normalerweise die Nachricht angezeigt, dass Skripts deaktiviert sind und keines der aufgelisteten Profile ausgeführt wird. Dabei handelt es sich um die sichere Standardkonfiguration der Windows PowerShell. Um Skripts zur Ausführung zuzulassen, geben Sie an der Eingabeaufforderung der Shell folgenden Befehl ein:

```
set-executionpolicy allsigned
```

Dieser Befehl legt die Ausführungsrichtlinie fest, dass nur Skripts ausgeführt werden, die über vertrauenswürdige Signaturen verfügen. Um die Umgebung weniger restriktiv einzustellen, verwenden Sie folgenden Befehl:

```
set-executionpolicy remotesigned
```

Dieser Befehl legt die Ausführungsrichtlinie fest, dass aus dem Internet heruntergeladene Skripts nur dann ausgeführt werden, wenn sie von einer vertrauenswürdigen Quelle signiert wurden. Um alle Beschränkungen aufzuheben, können Sie folgenden Befehl eingeben:

```
set-executionpolicy unrestricted
```

Dieser Befehl legt die Ausführungsrichtlinie fest, dass alle Skripts ausgeführt werden, ganz gleich, ob sie über digitale Signaturen verfügen oder nicht.

Ausführen und Verwenden von Commandlets

Die Windows PowerShell führt das *Commandlet* als neuen Begriff ein. Ein Commandlet stellt die kleinste Funktionseinheit in der Windows PowerShell dar. Sie können es sich als einen integrierten Befehl vorstellen. Anstatt hochkomplex zu sein, fallen die meisten Commandlets recht einfach aus und verfügen nur über einen kleinen Satz zugeordneter Eigenschaften.

Sie verwenden Commandlets genauso wie andere Befehle und Dienstprogramme. Bei den Namen der Commandlets kommt es nicht auf Groß-/Kleinschreibung an. Sie können sie daher in beliebigen Kombinationen von Groß- und Kleinbuchstaben eingeben. Nach dem Start der Windows PowerShell können Sie an der Eingabeaufforderung den Namen des Commandlets eingeben, worauf es auf fast genau dieselbe Weise wie ein Befehlszeilenbefehl ausgeführt wird.

Damit die Commandlets leichter verständlich sind, werden sie jeweils mit einem Paar aus einem Verb und einem Substantiv benannt. Wie Tabelle 6.1 verdeutlicht, verrät das Verb Ihnen, welche Aufgabe das Commandlet allgemein erledigt. Das Substantiv teilt Ihnen mit, welches Objekt das Commandlet bearbeitet. Das Commandlet **get-variable** liest beispielsweise eine benannte Windows PowerShell-Umgebungsvariable und gibt ihren Wert zurück. Wenn Sie die zu lesende Variable nicht angeben, zeigt **get-variable** eine Liste aller Windows PowerShell-Umgebungsvariablen mit ihren jeweiligen Werten an.

Tabelle 6.1: In Commandlets häufig verwendete Verben und ihre Bedeutungen

Commandlet-Verb	Verwendung
New-	Erstellt eine neue Instanz eines Elements, z.B. ein neues Postfach.
Remove-	Entfernt eine Instanz eines Elements, z.B. ein Postfach.
Enable-	Aktiviert eine Einstellung oder E-Mail-aktiviert einen Empfänger.
Disable-	Deaktiviert eine aktivierte Einstellung oder E-Mail-deaktiviert einen Empfänger. ▶

Tabelle 6.1: In Commandlets häufig verwendete Verben und ihre Bedeutungen (Forts.)

Commandlet- Verb	Verwendung
Set-	Ändert einzelne Einstellungen eines Objekts.
Get-	Fragt ein Objekt oder eine Untermenge eines Objekttyps ab, z.B. ein angegebenes Postfach oder alle Postfachbenutzer.

Sie können mit Commandlets auf zwei unterschiedliche Weisen arbeiten:

- Ausführen von Befehlen direkt aus der Eingabeaufforderung der Shell heraus
- Ausführen von Befehlen innerhalb von Skripten

Alle Befehle und Commandlets, die Sie von der Eingabeaufforderung der Windows PowerShell aufrufen können, können Sie als Text in eine Datei kopieren, und diese dann mit der Erweiterung PS1 speichern. Das Skript können Sie daraufhin auf dieselbe Art ausführen wie alle anderen Befehle und Commandlets.

HINWEIS *Die Windows PowerShell umfasst auch eine umfangreiche Skriptsprache und erlaubt die Verwendung von standardmäßigen Sprachkonstrukten für Programmschleifen, bedingte Ausführung, Ablaufsteuerung und Variablenzuweisung. Eine Erläuterung dieser Funktionen geht über den Rahmen dieses Buchs hinaus.*

Aus der Windows-Befehlszeilenumgebung oder aus einer Batchdatei heraus können Sie Windows PowerShell-Commandlets mit dem Parameter **-command** aufrufen. Wenn Sie so verfahren, sollten Sie normalerweise auch die Ausgabe des Windows PowerShell-Logo und die Ausführung von Profilen unterdrücken. Danach können Sie folgenden Befehl an einer Eingabeaufforderung eingeben oder in eine BAT-Datei einfügen:

```
powershell -nologo -nopprofile -command get-service
```

Abschließend sollten Sie bei der Arbeit mit der Windows PowerShell daran denken, dass das aktuelle Verzeichnis möglicherweise nicht im Umgebungspfad enthalten ist. Aus diesem Grund müssen Sie gegebenenfalls wie im folgenden Beispiel »./« voranstellen, wenn Sie ein Skript im aktuellen Verzeichnis ausführen wollen:

```
./runtasks
```

Ausführen und Verwenden weiterer Befehle und Dienstprogramme

Da die Windows PowerShell im Kontext der Windows-Eingabeaufforderung ausgeführt wird, können Sie aus ihr heraus alle Windows-Befehlszeilenbefehle, Dienstprogramme und grafischen Anwendungen aufrufen. Dabei sollten Sie jedoch beachten, dass der Interpreter der Windows PowerShell alle Befehle analysiert, bevor er sie an die Eingabeaufforderungsum-

gebung weiterreicht. Gibt es in der Windows PowerShell einen gleichnamigen Befehl oder Alias, wird dieser anstelle des gewünschten Windows-Befehls ausgeführt. (Weitere Informationen zu Aliasen finden Sie im Abschnitt »**Verwenden von Commandlet-Aliasen**« weiter hinten in diesem Kapitel.)

Nicht zur Windows PowerShell gehörende Befehle und Programme müssen sich in einem Verzeichnis befinden, das in der Umgebungsvariablen PATH aufgeführt ist. Wenn das Element im Verzeichnispfad gefunden wird, wird es ausgeführt. Die PATH-Variable steuert auch, wo die Windows PowerShell nach Anwendungen, Dienstprogrammen und Skripts sucht. Auf Windows-Umgebungsvariablen können Sie innerhalb der Windows PowerShell über das Präfix **\$env** zugreifen. Wenn Sie sich die aktuellen Einstellungen der Umgebungsvariable PATH anzeigen lassen wollen, geben Sie **\$env:path** ein. Wollen Sie dieser Variablen ein Verzeichnis hinzufügen, können Sie dafür folgende Syntax verwenden:

```
$env:path += ";Hinzuzuf,genderVerzeichnispfad"
```

Bei *Hinzuzuf,genderVerzeichnispfad* handelt es sich um den Verzeichnispfad, den Sie hinzufügen möchten, also z.B.:

```
$env:path += ";C:\Scripts"
```

Um dieses Verzeichnis jedes Mal zu PATH hinzuzufügen, sobald Sie die Windows PowerShell starten, können Sie die Befehlszeile als Eintrag in Ihr Profil aufnehmen. Beachten Sie, dass es sich bei Commandlets um integrierte Befehle und nicht um eigenständige ausführbare Dateien handelt. Aus diesem Grund hat die Umgebungsvariable PATH auf sie keinerlei Auswirkung.

Arbeiten mit Commandlets

Commandlets bilden die Grundlage für die Arbeit mit einem Rechner über die Windows PowerShell. Obwohl es viele unterschiedliche Commandlets mit vielen verschiedenen Einsatzmöglichkeiten gibt, verfügen alle auch über einige gemeinsame Eigenschaften. Diese möchte ich im folgenden Abschnitt behandeln.

Verwenden von Windows PowerShell-Commandlets

Von der Eingabeaufforderung der Windows PowerShell aus können Sie eine komplette Liste der verfügbaren Commandlets anzeigen lassen, indem Sie **get-command** eingeben. Um die Hilfedokumentation eines einzelnen Commandlets zu erhalten, geben Sie **help** ein, gefolgt von der Commandlet-Bezeichnung, also z.B.:

```
help get-variable
```

Tabelle 6.2 enthält eine Liste der Commandlets, die häufig zu Administrationszwecken eingesetzt werden. Auch wenn viele weitere Commandlets

zur Verfügung stehen, handelt es sich bei den hier genannten um die, die Sie am häufigsten verwenden werden.

Tabelle 6.2: In Commandlets häufig verwendete Verben und ihre Bedeutungen

Commandlet-Name	Beschreibung
ConvertFrom-SecureString	Exportiert eine sichere Zeichenfolge in ein geschütztes Format
ConvertTo-SecureString	Generiert eine sichere Zeichenfolge aus einer normalen Zeichenfolge
Get-Alias	Gibt Aliasnamen für Commandlets zurück
Get-AuthenticodeSignature	Liest das mit einer Datei verknüpfte Signaturobjekt
Get-Credential	Liest ein Anmeldeinformationen-Objekt in Abhängigkeit von einem Kennwort
Get-Date	Liest die aktuelle Uhrzeit und das Datum aus
Get-EventLog	Liest die Protokolldaten aus den Windows-Protokolldateien
Get-ExecutionPolicy	Liest die für die aktuelle Shell wirksamen Ausführungsrichtlinien
Get-Host	Liest Hostinformationen
Get-Location	Zeigt den aktuellen Standort an
Get-MshDrive	Liest die Laufwerksinformationen des angegebenen Msh-Laufwerks
Get-Service	Liest eine Liste der Dienste
Import-Alias	Importiert eine Aliasliste aus einer Datei
New-Alias	Erstellt ein neues Commandlet-Alias-Paar
New-Service	Erstellt einen neuen Dienst
Push-Location	Schiebt einen Speicherort auf den Stack
Read-Host	Liest eine Eingabezeile aus der Hostkonsole
Restart-Service	Startet einen beendeten Dienst neu
Resume-Service	Setzt einen angehaltenen Dienst fort
Set-Alias	Ordnet einen Alias einem Commandlet zu
Set-AuthenticodeSignature	Platziert eine Authenticode-Signatur in einem Skript oder in einer anderen Datei
Set-Date	Legt das Systemdatum und die Systemzeit im Hostsystem fest
Set-ExecutionPolicy	Legt die Ausführungsrichtlinien für die aktuelle Shell fest
Set-Location	Legt einen angegebenen Standort als den aktuellen Arbeitsstandort fest
Set-Service	Nimmt Änderungen an den Eigenschaften eines Dienstes vor
Start-Service	Startet einen beendeten Dienst

Tabelle 6.2: In Commandlets häufig verwendete Verben und ihre Bedeutungen (Forts.)

Commandlet-Name	Beschreibung
Start-Sleep	Hält Shell- oder Skriptaktivitäten für den angegebenen Zeitraum an
Stop-Service	Beendet einen laufenden Dienst
Suspend-Service	Hält einen laufenden Dienst an
Write-Output	Gibt ein Objekt über die Pipeline aus

Verwenden von Commandlet-Parametern

Jedem Commandlet-Parameter wird ein Strich (-) vorangestellt. Um den Schreibaufwand bei der Eingabe zu verringern, kommt es bei einigen Parametern auf die Stellung in der Parameterliste an, sodass Sie in bestimmten Fällen Parameter nur in einer bestimmten Reihenfolge aufzuführen brauchen, ohne die einzelnen Parameternamen nennen zu müssen. Für **get-service** müssen Sie beispielsweise die Bezeichnung **-Name** nicht aufführen, sondern können einfach Folgendes eingeben:

```
Get-service Dienstname
```

wobei *Dienstname* den Namen des Dienstes angibt, den Sie untersuchen wollen, also z.B.:

```
Get-service MExchangeIS
```

Diese Befehlszeile gibt den Status des Microsoft Exchange-Informationsspeicherdienstes zurück. Da Sie für Namenswerte auch Platzhalter wie * einsetzen können, können Sie auch **get-service mse*** eingeben, um den Status aller Dienste mit Microsoft Exchange-Bezug zu erhalten.

Alle Commandlets unterstützen die in Tabelle 6.3 genannte Liste der gemeinsamen Parameter. Damit Sie sie jedoch nutzen können, müssen Sie das Commandlet so aufrufen, dass die Parameter als Bestandteil des Resultsets zurückgegeben werden.

Tabelle 6.3: Gemeinsame Commandlet-Parameter

Parametername	Beschreibung
-Confirm	Hält Prozesse an und wartet auf eine Bestätigung durch den Benutzer, bevor der Vorgang fortgesetzt wird. Die Remove- und Disable- Commandlets nutzen diesen Parameter.
-Debug	Stellt Debuginformationen auf Programmiererebene über den Vorgang zur Verfügung.
-ErrorAction	Steuert das Verhalten beim Auftreten eines Fehlers.
-ErrorVariable	Legt den Namen der Variablen fest, in der (zusätzlich zur Ausgabe der standardmäßigen Fehlermeldung) Objekte gespeichert werden, bei denen ein Fehler aufgetreten ist.
-OutBuffer	Legt den Ausgabepuffer für das Commandlet fest. ▶

Tabelle 6.3: Gemeinsame Commandlet-Parameter (Forts.)

Parametername	Beschreibung
-OutVariable	Legt den Namen der Variablen fest, in der auszugebende Objekte gespeichert werden sollen.
-Verbose	Stellt Detailinformationen zum Vorgang zur Verfügung
-WhatIf	Zeigt dem Benutzer an, was geschehen würde, falls ein Commandlet mit einem bestimmten Parametersatz aufgerufen würde. Die Remove- und Disable- Commandlets nutzen diesen Parameter.

Erläuterungen zu Commandlet-Fehlermeldungen

Bei der Arbeit mit Commandlets begegnen Ihnen standardmäßig zwei verschiedene Arten von Fehlern:

- **Fehler, die die Ausführung abbrechen**
- **Fehler, die die Ausführung nicht abbrechen** Hierbei wird eine Fehlermeldung ausgegeben, die Ausführung aber nicht unterbrochen.

Bei beiden Fehlerarten wird Ihnen normalerweise ein Fehlertext angezeigt, der Ihnen beim Beheben des Problems behilflich sein kann. Es kann beispielsweise sein, dass eine erwartete Datei fehlt oder Sie nur über unzureichende Berechtigungen verfügen, um eine bestimmte Aufgabe ausführen zu können.

Verwenden von Commandlet-Aliasen

Um die Verwendung zu erleichtern, erlaubt Ihnen die Windows PowerShell das Erstellen von Aliasen für Commandlets. Bei einem Alias handelt es sich um eine Kurzform für ein Commandlet. Sie können z.B. den Alias **gsv** anstelle des Commandlet-Namens **get-service** verwenden.

Tabelle 6.4 enthält eine Liste von oft eingesetzten Standardaliasen. Es gibt zwar noch viele weitere Aliase, aber diese werden am häufigsten verwendet.

Tabelle 6.4: Häufig verwendete Commandlet-Aliase

Alias	Commandlet
clear, cls	Clear-Host
Diff	Compare-Object
cp, copy	Copy-Item
Epal	Export-Alias
Epcsv	Export-Csv
ForEach	ForEach-Object
Fl	Format-List
Ft	Format-Table

Tabelle 6.4: Häufig verwendete Commandlet-Aliase (Forts.)

Alias	Commandlet
Fw	Format-Wide
Gal	Get-Alias
Is, dir	Get-ChildItem
Gcm	Get-Command
cat, type	Get-Content
h, history	Get-History
gl, pwd	Get-Location
gps, ps	Get-Process
Gsv	Get-Service
Gv	Get-Variable
Group	Group-Object
Ipal	Import-Alias
Ipcsv	Import-Csv
R	Invoke-History
Ni	New-Item
Mount	New-MshDrive
Nv	New-Variable
rd, rm, rmdir, del, erase	Remove-Item
Rv	Remove-Variable
Sal	Set-Alias
sl, cd, chdir	Set-Location
sv, set	Set-Variable
Sort	Sort-Object
Sasv	Start-Service
Sleep	Start-Sleep
spps, kill	Stop-Process
Spsv	Stop-Service
write, echo	Write-Output

Zusätzliche Aliase können Sie über das Commandlet **Set-Alias** definieren. Seine Syntax lautet:

```
Set-alias Aliasname cmdletName
```

Bei *Aliasname* handelt es sich um den Alias, den Sie verwenden wollen, und *Commandletn* ist das Commandlet, für das Sie den Alias festlegen. Das folgende Beispiel erstellt einen Alias namens **go** für das Commandlet **get-process**:

```
Set-alias go get-process
```

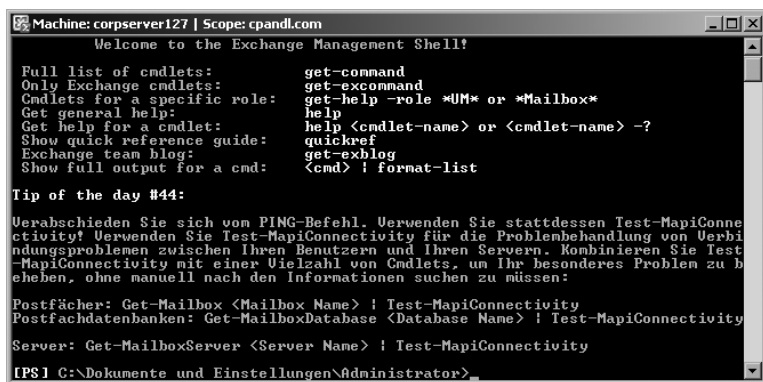
Um Ihre benutzerdefinierten Aliase in mit der Windows PowerShell jederzeit verwenden zu können, fügen Sie die entsprechenden Befehlszeilen Ihrer Profildatei hinzu.

Verwenden der Exchange-Verwaltungsshell

Bei der Exchange-Verwaltungsshell handelt es sich um eine befehlszeilen-gesteuerte Verwaltungsschnittstelle, die auf der Windows PowerShell basiert. Sie verwenden sie zur Verwaltung aller Aspekte der Exchange Server 2007-Konfiguration, die Sie auch über die Exchange-Verwaltungskonsole verändern können. Das bedeutet, dass Sie normalerweise jedes der beiden Werkzeuge zur Konfiguration von Exchange Server 2007 verwenden können. Es gibt jedoch auch einige Konfigurationseinstellungen, die Sie nur mithilfe der Exchange-Verwaltungsshell vornehmen können.

Ausführen und Verwenden der Exchange-Verwaltungsshell

Nachdem Sie die Exchange-Verwaltungstools auf einem Rechner installiert haben, können Sie die Exchange-Verwaltungsshell aufrufen, indem Sie auf **Start** klicken, **Alle Programme** wählen, danach **Microsoft Exchange Server 2007** und dann **Exchange-Verwaltungsshell**. Dass Sie mit der Exchange-Verwaltungsshell arbeiten, erkennen Sie daran, dass in der Titelzeile der Eingabeaufforderung **Machine:** zu lesen ist, gefolgt vom Servernamen und dem aktuellen Arbeitsverzeichnis. Dem aktuellen Verzeichnispfad geht die Zeichenfolge **[HPS]** voran, wie in Abbildung 6.2 zu sehen ist.



```
Machine: corpserver127 | Scope: cpandl.com
Welcome to the Exchange Management Shell!

Full list of cmdlets:      get-command
Only Exchange cmdlets:   get-excommand
Cmdlets for a specific role: get-help -role *UM* or *Mailbox*
Get general help:         help
Get help for a cmdlet:    help <cmdlet-name> or <cmdlet-name> -?
Show quick reference guide: quickref
Exchange team blog:       get-exblog
Show full output for a cmd: <cmd> ! format-list

Tip of the day #44:
Verabschieden Sie sich vom PING-Befehl. Verwenden Sie stattdessen Test-MapiConnectivity! Verwenden Sie Test-MapiConnectivity für die Problembehandlung von Verbindungsproblemen zwischen Ihren Benutzern und Ihren Servern. Kombinieren Sie Test-MapiConnectivity mit einer Vielzahl von Cmdlets, um Ihr besonderes Problem zu beheben, ohne manuell nach den Informationen suchen zu müssen:

Postfach: Get-Mailbox <Mailbox Name> | Test-MapiConnectivity
Postfachdatenbanken: Get-MailboxDatabase <Database Name> | Test-MapiConnectivity
Server: Get-MailboxServer <Server Name> | Test-MapiConnectivity

[HPS] C:\Dokumente und Einstellungen\Administrator>
```

Abbildung 6.2: Verwalten von Exchange Server mithilfe der Exchange-Verwaltungsshell an der Befehlszeile

Wenn Sie die Exchange-Verwaltungsshell starten, wird sie über eine Microsoft-Konfigurationsdatei (MCF1-Datei) initialisiert, die die Shell anweist,

den Administratormodus zu verwenden und bestimmte Sitzungsinformationen über die Active Directory-Umgebung einzuholen, in der Exchange Server zum Einsatz kommt. Da die Exchange-Verwaltungsshell eine Erweiterung der Windows PowerShell darstellt, werden auch für sie *Benutzer- und Systemprofile* ausgeführt, um die Umgebung einzurichten. Wollen Sie bestimmte Umgebungseinstellungen jedes Mal einsetzen, wenn Sie die Exchange-Verwaltungsshell verwenden, sollten Sie sie Ihrem Benutzerprofil hinzufügen – entweder in `%UserProfile%\Eigene Dateien\PSConfiguration\profile.ps1` oder in `%UserProfile%\Eigene Dateien\PSH\Microsoft.PowerShell_profile.ps1`.

Bei der Arbeit mit der Exchange-Verwaltungsshell stehen Ihnen alle Windows PowerShell-Commandlets und -Aliase zur Verfügung. Es kommen zwar weitere Commandlets hinzu, doch werden der Arbeitsumgebung keine zusätzlichen Aliase hinzugefügt. Um Ihre Sitzung zu beenden, können Sie die Exchange-Verwaltungsshell verlassen und zur Eingabeaufforderung zurückkehren, indem Sie **exit** eingeben. Alternativ können Sie das Shellfenster schließen, indem Sie auf **Schließen** klicken.

Arbeiten mit Exchange-Commandlets

Für die Arbeit mit der Exchange-Verwaltungsshell stehen Ihnen zusätzliche Exchange-spezifische Commandlets zur Verfügung. Wie für Windows PowerShell-Commandlets können Sie auch für Exchange-Commandlets Hilfeinformationen erhalten.

- Um eine Liste aller Exchange-Commandlets anzeigen zu lassen, geben Sie an der Eingabeaufforderung der Shell **get-excommand** ein.
- Um Exchange-Commandlets für eine bestimmte Serverfunktion anzeigen zu lassen, geben Sie **get-help -role Funktionsname** ein, wobei *Funktionsname* die Bezeichnung der gewünschten Serverfunktion darstellt. Sie können folgende Funktionsnamen verwenden:
 - ***UM*** für Commandlets, die für die Unified Messaging-Serverfunktion zuständig sind
 - ***Mailbox*** für Commandlets, die für die Postfachserverfunktion zuständig sind
 - ***ClientAccess*** für Commandlets, die für die zugriffserverfunktion zuständig sind

Beim Umgang mit der Exchange-Verwaltungsshell werden Sie häufig mit **Get-**, **Set-**, **Enable-**, **Disable-**, **New-** und **Remove-**Commandlets arbeiten. Alle diese Commandlets akzeptieren den Parameter **-Identity**, der das Objekt, mit dem Sie arbeiten, eindeutig bezeichnet.

Normalerweise führt ein solches Commandlet den Parameter **-Identity** an erster Stelle auf, sodass Sie die Identität mit oder ohne den Parameternamen angeben können. Wenn ein Objekt sowohl über einen Namen als auch über einen Alias verfügt, können Sie beide wahlweise für die Identität verwenden. Sie können beispielsweise jede der folgenden Methoden

benutzen, um das Postfachobjekt für den Benutzer **William Stanek** mit dem E-Mail-Alias **Williams** zu lesen:

```
get-mailbox -identity Williams
get-mailbox -identity 'William Stanek'
get-mailbox Williams
get-mailbox "William Stanek"
```

Bei **Get-Commandlets** können Sie üblicherweise eine Objektmenge mit allen in Frage kommenden Elementen zurückgeben lassen, indem Sie die Identität einfach weglassen. Wenn Sie in der Eingabeaufforderung der Shell z.B. **get-mailbox** eingeben, ohne eine Identität aufzuführen, erhalten Sie eine Liste aller Postfächer im Unternehmen (bis zur maximal in einer einzelnen Objektmenge erlaubten Anzahl).

Arbeiten mit Objektmengen und Umleiten der Ausgabe

Bei der Arbeit mit der Exchange-Verwaltungsshell müssen Sie häufig die Ausgabe eines Commandlets umleiten, damit sie einem anderen Commandlet als Eingabe dienen kann. Dies können Sie über das Pipe-Symbol (|) veranlassen. Wenn Sie beispielsweise die Postfächer einer bestimmten Postfachdatenbank statt aller Postfächer im Unternehmen anzeigen lassen wollen, können Sie die Ausgabe von **get-mailboxdatabase** wie im folgenden Beispiel zu **get-mailbox** umleiten:

```
get-mailboxdatabase -Identity iTechnik | get-mailbox
```

Hier greifen Sie über **get-mailboxdatabase** auf das Postfachdatenbankobjekt **Technik** zu. Dieses Objekt senden Sie dann als Eingabe an das Commandlet **get-mailbox**, das dann nacheinander alle Postfächer in dieser Datenbank abarbeitet. Wenn Sie keine weiteren Änderungen daran vornehmen, werden die Postfächer dieser Datenbank wie folgt in Listenform ausgegeben:

Name	Alias	Server	ProhibitSendQuota
Administrator	Administrator	corpsvr127	unbegrenzt
William S	williams	corpsvr127	unbegrenzt
Tom G	tomg	corpsvr127	unbegrenzt
David W	davidw	corpsvr127	unbegrenzt
Kari F	karif	corpsvr127	unbegrenzt
Connie V	conniev	corpsvr127	unbegrenzt
Mike D	miked	corpsvr127	unbegrenzt

Diese Ausgabe könnten Sie auch an ein anderes Commandlet weiterleiten, um an jedem einzelnen Postfach eine Aktion vorzunehmen.

Arbeiten mit Exchange-Commandlets

Exchange-Commandlets verwenden Sie zur Konfigurationsverwaltung in Ihrer Exchange-Organisation. Diese Commandlets arbeiten mit Objekten, die bestimmte Kriterien erfüllen. Die folgenden Abschnitte bieten einen

Überblick über die am häufigsten eingesetzten Commandlets und die jeweils gebräuchlichste Syntax.

Universelle Commandlets

Ihnen stehen einige universell verwendbare Commandlets zur Verfügung, die in Folgenden jeweils mit ihrer Syntax angegeben sind:

- **Get-ExchangeServer** Ruft eine Liste von allen oder von den angegebenen Exchange-Servern ab.

```
Get-ExchangeServer -Domain 'Domänenname'  
[-DomainController 'DC-Name']
```

- **Get-Recipient** Ruft eine Liste von allen oder von den angegebenen Empfängern ab.

```
Get-Recipient [-RecipientType 'Empfänger-ID']  
[-Identity 'ID'] [-Format-List'] [-DomainController 'DC-Name']  
[-OrganizationalUnit 'OE-Name'] [-Anr 'Zeichenkette']
```

HINWEIS *Über den Parameter -Anr wird eine Zeichenkette angegeben, die bei der Auflösung mehrdeutiger Namen behilflich ist. In den angegebenen Objekten wird nach jedem hier genannten Wert gesucht.*

Commandlets zur Kontaktverwaltung

Kontakte bearbeiten Sie mithilfe folgender Commandlets:

- **Enable-Mailcontact** E-Mail-aktiviert einen Kontakt.

```
Enable-Mailcontact -Identity 'ID'  
-externalEmailAddress 'E-Mail-Adresse'  
[-alias 'Alias']  
[-DomainController 'DC-Name']
```

- **Disable-MailContact** E-Mail-deaktiviert einen Kontakt.

```
Disable-MailContact -Identity 'Alias'  
[-DomainController 'DC-Name']
```

- **Get-MailContact** Ruft eine Liste von allen oder von den angegebenen E-Mail-aktivierten Kontakten ab.

```
Get-MailContact [-Identity 'ID'] [-Format-List']  
[-DomainController 'DC-Name'] [-OrganizationalUnit 'OE-Name']  
[-Anr 'Zeichenkette']
```

- **Set-MailContact** Ändert die angegebenen Eigenschaften des angegebenen E-Mail-aktivierten Kontakts.

```
Set-MailContact -Identity 'ID' [-Alias 'NeuerAlias']  
[-AcceptMessagesOnlyFrom 'Empfänger']  
[-DeliverToForwardingAddress <$false|$true>]  
[-DisplayName 'Name'] [-DomainController 'DC-Name']  
[-EmailAddresses 'Proxyadresse']  
[-EmailAddressPolicyEnabled: <$false|$true>]  
[-ExternalEmailAddress 'Proxyadresse']  
[-ForwardingAddress 'Empfänger']
```

```
[ -GrantSendOnBehalfTo 'Postfach' ]
[ -HiddenFromAddressListsEnabled <$false|$true> ]
[ -MaxReceiveSize 'Größe' ] [ -MaxRecipientPerMessage 'Größe' ]
[ -MaxSendSize 'Größe' ] [ -MessageBodyFormat 'Format' ]
[ -MessageFormat 'Format' ] [ -Name 'Name' ]
```

```
[ -PrimarySmtpAddress 'SMTP-Adresse' ]
[ -RejectMessagesFrom 'Empfänger' ]
[ -RejectMessagesFromDLMembers 'Empfänger' ]
[ -RequireSenderAuthenticationEnabled: <$false|$true> ]
[ -SimpleDisplayName 'Name' ]
```

- **Get-Contact** Ruft eine Liste von allen oder von den angegebenen Kontakten ab, unabhängig davon, ob sie E-Mail-aktiviert sind.

```
Get-Contact [ -DomainController 'DC-Name' ] [ -OrganizationalUnit 'OE-Name' ] [ -ResultSize 'Größe' ]
```

- **Set-Contact** Ändert die angegebenen Eigenschaften des angegebenen Kontakts oder legt sie fest.

```
Set-Contact -Identity 'ID' [ -AssistantName 'Name' ]
[ -City 'Name' ]
[ -Company 'Name' ] [ -CountryOrRegion 'Name' ]
[ -Department 'Name' ]
[ -DisplayName 'Name' ] [ -DomainController 'DC-Name' ]
[ -Fax 'Faxnummer' ]
[ -FirstName 'Name' ] [ -HomePhone 'Telefonnummer' ]
[ -Initials 'Wert' ]
[ -LastName 'Name' ] [ -Manager 'Empfänger-ID' ]
[ -MobilePhone 'Telefonnummer' ]
[ -Name 'Name' ] [ -Notes 'Wert' ] [ -Office 'Wert' ]
[ -Phone 'Telefonnummer' ]
[ -PostalCode 'PLZ' ] [ -SimpleDisplayName 'Name' ]
[ -StateOrProvince 'Wert' ]
[ -StreetAddress 'Wert' ] [ -TelephoneAssistant 'Wert' ]
[ -Title 'Wert' ]
[ -WebPage 'Wert' ]
```

Commandlets zur Benutzerverwaltung

Benutzer können Sie mithilfe folgender Commandlets verwalten:

- **Get-User** Ruft eine Liste von allen oder von den angegebenen Active Directory-Benutzern ab.

```
Get-User [ -Identity 'ID' ] [ -DomainController 'DC-Name' ]
[ -OrganizationalUnit 'OE-Name' ] [ -ResultSize 'Größe' ] [ -SortBy 'Wert' ]
```

- **Disable-MailUser** Hebt die E-Mail-Aktivierung des angegebenen Active Directory-Benutzers auf.

```
Disable-MailUser -Identity 'ID'
[ -DomainController 'DC-Name' ]
```

- **Get-MailUser** Ruft eine Liste von allen oder von näher spezifizierten E-Mail-aktivierten Benutzern ab.

```
Get-MailUser [-Identity 'ID'] [-DomainController 'DC-Name']
[-OrganizationalUnit 'OE-Name'] [-ResultSize 'Größe'] [-SortBy 'Wert']
```

- **Set-MailUser** Legt die angegebenen Eigenschaften für den angegebenen Benutzer fest.

```
Set-MailUser [-Identity 'ID'] [-AllowMerge <$false|$true>]
[-AttachmentFileNames 'Namen'] [-BadItemLimit 'Anzahl']
[-ContentKeywords 'Zeichenketten'] [-DomainController 'DC-Name']
[-EndDate 'DatumUhrzeit'] [-ExcludeFolders 'MapiVerzeichnispfad']
[-GlobalCatalog 'GCName'] [-IgnorePolicyMatch <$false|$true>]
[-IncludeFolders 'MAPI-Verzeichnispfad'] [-Locale 'Wert']
[-MaxThreads 'Anzahl']
[-NTAccountOU 'OE-ID'] [-PreserveMailboxSizeLimit <$false|$true>]
[-ReportFile 'LokalerPfad'] [-RetryInterval 'Zeitspanne']
[-RetryTimeout 'Zeitspanne'] [-StartDate 'DatumUhrzeit']
[-SubjectKeywords 'Werte'] [-ValidateOnly <$false|$true>]
```

Commandlets zur Verteilergruppenverwaltung

Verteilergruppen bearbeiten Sie mithilfe folgender:

- **Enable-DistributionGroup** E-Mail-aktiviert eine vorhandene universelle Verteilergruppe.

```
Enable-DistributionGroup -Identity 'ID' [-Alias 'Alias']
[-DisplayName 'Name'] [-DomainController 'DC-Name']
```

- **Disable-DistributionGroup** E-Mail-deaktiviert eine angegebene universelle Verteilergruppe.

```
Disable-DistributionGroup -Identity 'ID'
[-DomainController 'DC-Name']
```

- **Get-DistributionGroup** Ruft eine Liste von allen oder von den angegebenen E-Mail-aktivierten universellen Verteilergruppen ab.

```
Get-DistributionGroup [-Identity 'ID']
[-DomainController 'DC-Name']
[-ManagedBy 'Empfänger-ID'] [-OrganizationalUnit 'OE-Name']
[-ResultSize 'Größe'] [-SortBy 'Wert']
```

- **Set-DistributionGroup** Ändert die angegebenen Eigenschaften der spezifizierten Verteilergruppe.

```
Set-DistributionGroup -Identity 'ID' [-Alias 'NeuerAlias']
[-AcceptMessagesOnlyFrom 'Empfänger'] [-DisplayName 'Name']
[-DomainController 'DC-Name'] [-EmailAddresses 'Proxyadresse']
[-EmailAddressPolicyEnabled <$false|$true>] [-ExpansionServer 'Server']
[-GrantSendOnBehalfTo 'Postfach']
[-HiddenFromAddressListsEnabled <$false|$true>]
[-MaxReceiveSize 'Größe'] [-MaxSendSize 'Größe']
[-Name 'Name'] [-PrimarySmtpAddress 'SMTP-Adresse']
[-RejectMessagesFrom 'Empfänger']
[-RejectMessagesFromDLMembers 'Empfänger']
[-SimpleDisplayName 'Name']
```


- **Add-DistributionGroupMember** Fügt den angegebenen Benutzer der universellen Verteilergruppe hinzu.

```
Add-DistributionGroupMember -Identity 'ID' -Member 'Empfänger-ID'  
[-DomainController 'DC-Name']
```

- **Remove-DistributionGroupMember** Entfernt den angegebenen Benutzer aus der universellen Verteilergruppe.

```
Remove-DistributionGroupMember -Identity 'ID' -Member 'Empfänger-ID'  
[-DomainController 'DC-Name']
```

- **Get-DistributionGroupMember** Ruft eine Liste aller Mitglieder der angegebenen Verteilergruppe ab.

```
Get-DistributionGroupMember -Identity 'ID'  
[-DomainController 'DC-Name'] [-ResultSize 'Größe']
```

- **Get-Group** Ruft eine Liste aller Sicherheits- und Verteilergruppen ab.

```
Get-Group [-Identity 'ID'] [-DomainController 'DC-Name']  
[-OrganizationalUnit 'OE-Name'] [-ResultSize 'Größe'] [-SortBy 'Wert']
```

- **Set-Group** Legt die angegebenen Eigenschaften für die spezifizierte Windows-Gruppe fest.

```
Set-Group -Identity 'ID' [-DisplayName 'NeuerAnzeigenname']  
[-DomainController 'DC-Name'] [-ManagedBy 'Empfänger-ID']  
[-Name 'Name'] [-Notes 'Wert'] [-SimpleDisplayName 'Name']
```

Commandlets zur Postfachverwaltung

Postfächer bearbeiten Sie mithilfe folgender:

- **Enable-Mailbox** Postfachaktiviert ein vorhandenes Active Directory-Benutzerkonto.

```
Enable-Mailbox -Identity 'Domäne\Benutzername' -Database 'Postfachdaten-  
bank' [-Alias 'Alias'] [-DomainController 'DC-Name'] [-ManagedFolder-  
MailboxPolicy 'Richtlinien-Id'] [-MobileMailboxPolicy 'Richtlinien-Id']
```

- **Disable-Mailbox** Hebt die Postfachaktivierung des angegebenen Benutzerkontos auf.

```
Disable-Mailbox -Identity 'ID' [-DomainController 'DC-Name']
```

- **Set-Mailbox** Ändert die angegebenen Eigenschaften des spezifizierten Postfachs.

```
Set-Mailbox -Identity 'ID'  
[-AcceptMessagesOnlyFrom 'Empfänger-ID']  
[-AcceptMessagesOnlyFromDLMembers 'Empfänger-ID']  
[-Alias 'Alias']  
[-AntispamBypassEnabled: <$false|$true>]  
[-DeliverToMailboxAndForward: <$false|$true>]  
[-DisplayName 'Name'] [-DomainController 'DC-Name']  
[-EmailAddresses 'Proxyadressen'] [-EmailAddressPolicyEnabled  
<$false|$true>] [-ForwardingAddress 'Empfänger-ID']  
[-GrantSendOnBehalfTo 'Postfach-Id']  
[-HiddenFromAddressListsEnabled 'Status']
```

```
[-IssueWarningQuota 'Größe'] [-ManagedFolderMailboxPolicy 'Postfach-
richtlinien-Id']
[-MaxReceiveSize 'Größe'] [-MaxSendSize 'Größe'] [-Name 'Name']
[-Office 'Wert'] [-OfflineAddressBook 'OfflineadressbuchId']
[-PrimarySmtpAddress 'SMTP-Adresse'] [-ProhibitSendQuota 'Größe']
[-ProhibitSendReceiveQuota 'Größe'] [-RecipientLimits 'Größe']
[-RejectMessagesFrom 'Empfänger-ID']
[-RejectMessagesFromDLMembers 'Empfänger-ID']
[-RequireSenderAuthenticationEnabled <$false|$true>]
[-RetainDeletedItemsFor 'Zeit'] [-RetainDeletedItemsUntilBackup:
<$false|$true>]
[-RetentionHoldEnabled <$false|$true>]
[-SamAccountName 'Name'] [-UserPrincipalName 'Name']
[-WindowsEmailAddress 'SMTP-Adresse']
```

- **Get-Mailbox** Ruft eine Liste von allen oder von den angegebenen Postfächern ab.

```
Get-Mailbox [-Identity 'ID'] | [-Database 'Datenbankname'] [-DomainCont-
roller 'DC-Name']
[-OrganizationalUnit 'OE-Name'] [-ResultSize 'Größe'] [-SortBy 'Wert']
```

- **Get-MailboxStatistics** Ruft zusammenfassende Statistiken für alle oder für die angegebenen Postfächer ab, sofern für die Postfächer jeweils mindestens eine Anmeldung erfolgt ist.

```
Get-MailboxStatistics [-Identity 'ID'] [-Database 'Postfachdatenbank']
[-Server 'Server']
```

- **Move-Mailbox** Verschiebt das Postfach des angegebenen Benutzers zum angegebenen Server.

```
Move-Mailbox -Identity 'ID' -TargetDatabase 'Server\Postfachdatenbank'
[-AllowMerge <$false|$true>] [-AttachmentFileNames 'Werte']
[-ContentKeywords 'Werte'] [-DomainController 'DC-Name']
[-EndDate 'DatumUhrzeit'] [-ExcludeFolders 'MAPI-Verzeichnispfad']
[-GlobalCatalog 'GCName'] [-IgnorePolicyMatch <$false|$true>]
[-IncludeFolders 'MapiVerzeichnispfad'] [-Locale 'Wert']
[-MaxThreads 'Anzahl']
[-NTAccountOU 'OE-ID'] [-PreserveMailboxSizeLimit 'Schalter']
[-ReportFile 'LokalerPfad'] [-RetryInterval 'Zeitspanne']
[-RetryTimeout 'Zeitspanne'] [-StartDate 'DatumUhrzeit']
[-SubjectKeywords 'Werte'] [-ValidateOnly: <$false|$true>]
```

Commandlets zur Datenbankverwaltung

Exchange-Datenbanken können Sie mithilfe folgender Commandlets verwalten:

- **New-MailboxDatabase** Erstellt in der angegebenen Speichergruppe eine neue Postfachdatenbank.

```
New-MailboxDatabase -Name 'Postfachdatenbank' -StorageGroup 'Speicher-
gruppe'
[-CopyEdbFilePath 'EDB-Dateipfad'] [-DomainController 'DC-Name']
[-EdbFilePath 'EDB-Dateipfad'] [-HasLocalCopy <$false|$true>]
[-OfflineAddressBook 'OAB-ID'] [-PublicFolderDatabase 'Datenbank-ID']
```

- ◉ **Remove-MailboxDatabase** Entfernt die angegebene Postfachdatenbank.

```
Remove-MailboxDatabase -Identity 'Postfachdatenbank'
[-DomainController 'DC-Name']
```

- ◉ **Set-MailboxDatabase** Legt die angegebenen Eigenschaften der spezifizierten Postfachdatenbank fest.

```
Set-MailboxDatabase [-Identity 'Postfachdatenbank']
[-AllowFileRestore <$false|$true>] [-DomainController 'DC-Name']
[-EventHistoryRetentionPeriod 'Zeitspanne'] [-FixedFont <$false|$true>]
[-IndexEnabled <$false|$true>] [-IssueWarningQuota 'Größe']
[-ItemRetention 'Zeitspanne'] [-JournalRecipient 'Empfänger-ID']
[-MailboxRetention 'Zeitspanne'] [-MaintenanceSchedule 'Zeitplan']
[-MountAtStartup <$false|$true>] [-Name 'Name']
[-OfflineAddressBook 'OAB-ID'] [-ProhibitSendQuota 'Größe']
[-ProhibitSendReceiveQuota 'Größe'] [-PublicFolderDatabase 'Datenbank-ID']
[-QuotaNotificationSchedule 'Zeitplan'] [-RestoreInProgress <$false|$true>]
[-RetainDeletedItemsUntilBackup <$false|$true>]
[-SMimeSignatureEnabled <$false|$true>]
```

- ◉ **Get-MailboxDatabase** Ruft eine Liste aller oder den angegebenen Postfachdatenbanken ab.

```
Get-MailboxDatabase [-Identity 'Postfachdatenbank']
[-DomainController 'DC-Name']
```

```
Get-MailboxDatabase [-StorageGroup 'Speichergruppe']
[-DomainController 'DC-Name']
```

```
Get-MailboxDatabase [-Server 'Server'] [-DomainController 'DC-Name']
```

- ◉ **Mount-Database** Stellt die angegebene Postfachdatenbank bereit.

```
Mount-Database -Identity 'Postfachdatenbank' [-DomainController 'DC-Name']
[-Force <$false|$true>]
```

- ◉ **Dismount-Database** Hebt die Bereitstellung für die angegebene Postfachdatenbank auf.

```
Dismount-Database -Identity 'Postfachdatenbank'
[-DomainController 'DC-Name']
```

- ◉ **Enable-DatabaseCopy** Aktiviert die lokale kontinuierliche Sicherung für die angegebene Postfachdatenbank.

```
Enable-DatabaseCopy -Identity 'Postfachdatenbank'
[-CopyEdbFilePath 'EDB-Dateipfad'] [-DomainController 'DC-Name']
```

Commandlets zur Speichergruppenverwaltung

Exchange-Speichergruppen können Sie mithilfe folgender Commandlets verwalten:

- ◉ **New-StorageGroup** Erstellt die benannte Speichergruppe auf dem angegebenen Server.

```
New-StorageGroup -Name 'Speichergruppenname' -Server 'Server'  
[-CircularLoggingEnabled <$false|$true>] [-CopyLogFolderPath 'Lokaler-  
Pfad']  
[-CopySystemFolderPath 'LokalerPfad'] [-DomainController 'DC-Name']  
[-HasLocalCopy <$false|$true>] [-LogFolderPath 'LokalerPfad']  
[-SystemFolderPath 'LokalerPfad'] [-ZeroDatabasePages <$false|$true>]
```

- **Get-StorageGroup** Ruft eine Liste aller oder den angegebenen Speichergruppen ab.

```
Get-StorageGroup [-Identity 'Speichergruppe'] [-DomainController 'DC-  
Name']
```

- **Set-StorageGroup** Ändert den Namen der angegebenen Speichergruppe in den genannten Wert.

```
Set-StorageGroup -Identity 'Speichergruppe'  
[-CircularLoggingEnabled <$false|$true>] [-DomainController 'DC-Name']  
[-Name 'Name']  
[-ZeroDatabasePages <$false|$true>]
```

- **Remove-StorageGroup** Löscht die angegebene Speichergruppe.

```
Remove-StorageGroup -Identity 'Speichergruppe'  
[-DomainController 'DC-Name']
```

- **Enable-StorageGroupCopy** Aktiviert die lokale kontinuierliche Sicherung für die angegebene Speichergruppe, vorausgesetzt, dass diese Funktion bereits für alle Datenbanken innerhalb der Speichergruppe aktiviert ist.

```
Enable-StorageGroupCopy -Identity 'Speichergruppe'  
[-CopyLogFolderPath 'LokalerPfad'] [-CopySystemFolderPath  
'LokalerPfad'] [-DomainController 'DC-Name']  
[-SeedingPostponed: <$false|$true>]
```

- **Disable-StorageGroupCopy** Deaktiviert die lokale kontinuierliche Sicherung der angegebenen Speichergruppe.

```
Disable-StorageGroupCopy -Identity 'Speichergruppe'  
[-DomainController 'DC-Name']
```