Bild 1.2: Über *Get-Mailbox* erhalten Sie Infos zu den vorhandenen Postfächern.

1.1.1 Hilfe in der PowerShell aufrufen

Möchten Sie lieber alle notwendigen Informationen und Optionen in einer Befehlszeile angeben, anstatt diese nach und nach einzugeben, bietet die PowerShell eine ausführliche Hilfe an. Mit dem Befehl *Help Cmdlet* erhalten Sie Informationen zum entsprechenden Cmdlet, zum Beispiel *Help New-Mailbox*. Für viele Cmdlets gibt es noch die Option "-Detailed". Mit dieser erhalten Sie noch ausführlichere Informationen. Mit *Help Cmdlet* -Examples lassen sich Beispiele für den Befehl anzeigen. Auch das funktioniert für alle Kommandos in der PowerShell.

In der PowerShell hat Microsoft die Hilfefunktion deutlich erweitert. Rufen Sie eine Hilfe zu Cmdlets auf, kann sich die PowerShell selbstständig aus dem Internet mit *update-help* aktualisieren. Dazu muss der Server natürlich über eine Internet-Verbindung verfügen. Alternativ laden Sie die Hilfedateien auf einem anderen Computer herunter und speichern diese auf Servern und Computern ohne Internet-Verbindung mittels *save-help* ab.

Ebenfalls hilfreich ist das Cmdlet *Show-Command*. Es blendet ein neues Fenster mit allen Befehlen ein, die in der PowerShell verfügbar sind. Sie können im Fenster nach Befehlen suchen und sich eine Hilfe zum Befehl sowie Beispiele anzeigen lassen. Außerdem stellen Sie mit dem Fenster auch eigene Befehle zusammen, ohne diese direkt in der PowerShell eingeben zu müssen.

1.1.2 Daten abfragen

Mit *Get-Cmdlets* lassen Sie sich Informationen zu Objekten anzeigen. Diese fallen in der Exchange Management Shell wesentlich umfangreicher aus als in der Exchange-Verwaltungskonsole. Mit der Option "|fl" formatieren Sie dabei die Ausgabe. Auch hier sehen Sie, wie viele Informationen die Exchange Management Shell zur Verfügung stellt. Wünschen Sie nicht alle Informationen, sondern nur einzelne Parameter, können Sie diese nach der Option "|fl" anordnen. Um etwa für das Postfach "thomas.joos@contoso.int" nur den Displaynamen, die Datenbank, den Alias und die Organisationseinheit (Organizational Unit, OU) anzuzeigen, verwenden Sie den Befehl

```
> Get-Mailbox thomas.joos@contoso.int |fl DisplayName, Database, Alias,
  OrganizationalUnit
```

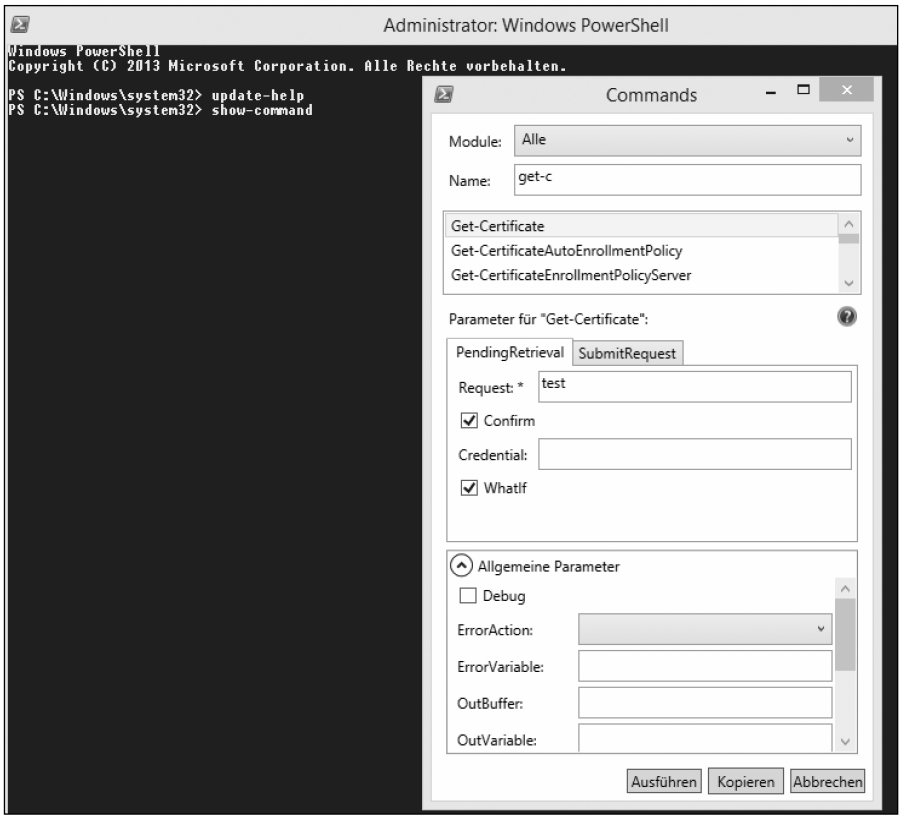

Bild 1.3: Die PowerShell hilft beim Zusammenstellen von Befehlen mit einer grafischen Oberfläche.

Auf diese Weise lassen Sie Informationen zu allen Objekten ausgeben, die mit *Get-Cmdlets* aufrufbar sind und dank "|fl" filtern Sie die Angaben.

Sie können in der PowerShell auch alle Exchange-Datenbanken ausgeben lassen, die auf den Servern Ihrer Organisation angelegt wurden. Rufen Sie dazu in der Exchange Management Shell den Befehl *Get-MailboxDatabase* auf. Sie erhalten mit diesem Befehl nicht nur eine formatierte Liste aller Postfachdatenbanken eines Servers, sondern alle Postfachdatenbanken auf allen Exchange-Servern in der Organisation. Auch hier gilt wieder, dass diese Befehle für alle Get-Cmdlets gelten. Diese können zu den Bordmitteln von Windows gehören oder zu anderen Servern wie SQL Server oder SharePoint. Auch hier ist der Umgang mit der PowerShell identisch.

Über den Befehl *Dismount-Database* heben Sie in Exchange die Bereitstellung einer Datenbank über die PowerShell wieder auf. Sie müssen dazu lediglich den Namen der Datenbank mitgeben, damit die Exchange Management Shell weiß, von welcher

Datenbank die Bereitstellung aufgehoben werden soll. Sie können so zum Beispiel den Befehl *Dismount-Database (Get-MailboxDatabase)* eingeben. In diesem Fall verwendet das Cmdlet *Dismount-Database* die Informationen, die das Cmdlet *Get-MailboxDatabase* ausgibt.

Geben Sie den Namen einer Datenbank direkt mit dem Befehl ein, müssen Sie Anführungszeichen verwenden. Das gilt für alle Objekte, in denen Leerzeichen vorkommen. Alternativ rufen Sie nur den Befehl *Dismount-Database* auf und geben dann den Namen ein, den Sie vorher mit *Get-MailboxDatabase* abgefragt haben. Das funktioniert ebenfalls in den meisten Cmdlets. Drücken Sie die Taste "Pfeil nach oben" auf der Tastatur, erscheint der eingegebene Befehl noch einmal. Schließlich verwenden Sie nach der Nutzung von *Dismount-Database* den Befehl *Mount-Database*, damit die Datenbank wieder bereitgestellt wird.

Eine weitere Möglichkeit der Exchange Management Shell besteht im Auslesen der Postfächer innerhalb einer Postfachspeicherdatenbank. Mit dem Befehl *Get-Mailbox* erhalten Sie eine Liste aller Postfächer der Organisation. Sie sehen dabei auch, auf welchem Server die einzelnen Postfächer liegen und ob ein Grenzwert eingetragen ist, der das Senden verbietet. Über den Befehl *Get-Mailbox | Format-Table DisplayName, Database* lassen Sie sich die Postfächer sortiert nach Postfachdatenbank und Anzeigenamen ausgeben.

1.1.3 Skripte in der PowerShell erstellen

Sie haben natürlich die Möglichkeiten, einzelne PowerShell-Befehle in Skripte mit der Endung PS1 zusammenzufassen. Um ein Skript auszuführen, verwenden Sie den Befehl mit der Syntax .*Pfad und Name der PS1-Datei*. Vergessen Sie zu Beginn das Präfix ".\" nicht. Sie müssen für die Ausführung ferner die Sicherheitseinstellungen der PowerShell anpassen.

In der PowerShell ist die sogenannte Ausführungsrichtlinie für Skripte jetzt standardmäßig auf "RemoteSigned" gesetzt. Die Ausführungsrichtlinie bestimmt, ob Skripte ausgeführt werden dürfen und ob diese digital signiert sein müssen. Standardmäßig blockierte die PowerShell Skripte. Sie können die Ausführungsrichtlinie mit dem Cmdlet *Set-ExecutionPolicy* ändern und mit *Get-ExecutionPolicy* anzeigen. Dabei stehen folgende Einstellungen zur Verfügung:

- Restricted: Standardeinstellung, keine Skripte erlaubt.
- AllSigned: Nur signierte Skripte sind erlaubt.
- RemoteSigned: Bei dieser Einstellung müssen Sie Skripte durch eine Zertifizierungsstelle signieren lassen.
- Unrestricted: Mit dieser Einstellung funktionieren alle Skripte.

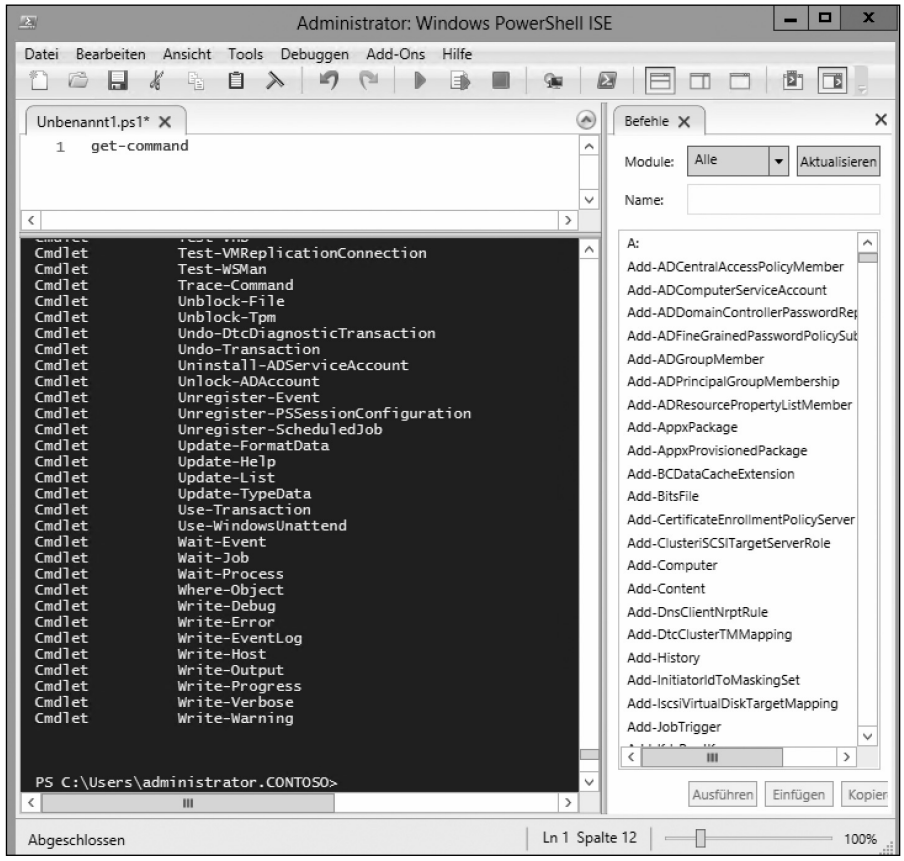

Bild 1.4: In der grafischen Oberfläche der PowerShell sehen Sie die verfügbaren Cmdlets alphabetisch aufgelistet.

Ebenfalls interessant ist die Oberfläche zur Erstellung von Skripten und Ausführung von Befehlen für die Windows PowerShell, das so genannte "Windows PowerShell Integrated Scripting Environment" (ISE). Auch diese rufen Sie über die Startseite durch Eintippen von *powershell ise* auf, oder indem Sie den Befehl *ise* in einer PowerShell-Sitzung eingeben. Die grafische Oberfläche bietet die Möglichkeit, Skripte für die PowerShell in einer einheitlichen Oberfläche zu erstellen. Der Vorteil dabei ist, dass Sie beim Eingeben von Befehlen bereits Vorschläge für Cmdlets unterbreitet bekommen. Außerdem sind die Befehle farblich hervorgehoben, sodass sich die einzelnen Optionen besser unterscheiden lassen.

Im oberen Bereich können Sie Skriptbefehle angeben und diese dann als Skript abspeichern. Des Weiteren haben Sie die Möglichkeit, eine PowerShell remote auf einem anderen Computer zu öffnen. Verwenden Sie dazu den Menüpunkt "Datei". Die PowerShell zeigt das ISE als zusätzliche Registerkarte an. Geben Sie im oberen Bereich

Befehle ein, werden diese nicht sofort ausgeführt, sondern wie in einem normalen Skript zunächst aufgelistet. Sind Sie fertig mit der Eingabe der Befehle, starten Sie deren Ausführung, indem Sie auf das grüne Abspielsymbol mit der QuickInfo "Skript ausführen" klicken.

Über den Menüpunkt "Ansicht" passen Sie die verschiedenen Bereiche des ISE an Ihre Bedürfnisse an. So lässt sich zum Beispiel der Bereich zum Erstellen von Skripten an der rechten Seite anordnen. Skripte können Sie während der Ausführung bearbeiten und so Fehler schneller beheben. Laden Sie ein Skript über "Datei / Öffnen", sehen Sie im Befehlsfenster dessen Bestandteile. Markieren Sie eine Zeile im Skript, können Sie über den Menüpunkt "Debuggen / Haltepunkt umschalten" eine Pause im Skript einbauen.

1.1.4 Variablen in der PowerShell verwenden

Interessant ist auch die Möglichkeit, dass Sie innerhalb der Shell Variablen definieren können, die aktuelle Informationen automatisch abfragen. Diese Variablen lassen sich dann später innerhalb eines Skripts verwenden. Möchten Sie beispielsweise das aktuelle Datum als Variable "$heute" hinterlegen, nutzen Sie in der Shell den Befehl *$heute = Get-Date*. Anschließend wird das heutige Datum in der Variablen hinterlegt. Geben Sie dann in der Shell *$heute* ein, erhalten Sie das aktuelle Datum.

Auf einzelne Bestandteile der Variable können Sie auch getrennt zugreifen. Interessiert Sie etwa aus dem Datum lediglich die Uhrzeit, lesen Sie ohne viel Aufwand mit *$heute.ToShortTimeString()* nur die Uhrzeit in Stunden und Minuten aus der Variable aus. Eine weitere Möglichkeit ist das Formatieren der Ausgabe. So erzwingen Sie per Eingabe des Befehls *$heute.ToString("MMMM")* die Ausgabe des Monats oder über *$heute.ToString("MM")* den Monat als Zahl innerhalb des Kalenderjahres. Generell können Sie hinter den meisten Befehlen, die einen Status oder eine Statistik ausgeben, noch den Zusatz "|fl" eingeben. Dieser Zusatz bewirkt, dass Sie eine formatierte Liste erhalten, die deutlich mehr Informationen ausgibt als der Befehl ohne diesen Zusatz.

Im Beispiel des Datums ist es jedoch auch möglich, einzelne Bestandteile ohne Variable auszulesen. Der Befehl *Get-Date -displayhint date* zeigt nur das Datum, *Get-Date -displayhint time* nur die Uhrzeit an. Auch können Sie ermitteln, welche Art von Objekt von einem bestimmten Cmdlet abgerufen wird, indem Sie die Ergebnisse des Befehls *Get* mit einem Pipeline-Operator (|) an den Befehl *Get-Member* übergeben.

1.1.5 PowerShell über das Netzwerk nutzen

Damit Sie einen Computer über die PowerShell remote verwalten können, aktivieren Sie zunächst die Remoteverwaltung auf dem Computer. Dazu geben Sie auf dem entspre-