

# Deploying and Managing Active Directory with Windows PowerShell

Tools for cloud-based and  
hybrid environments

Charlie Russel



# Deploying and Managing Active Directory with Windows PowerShell

Tools for cloud-based and  
hybrid environments

Charlie Russel

PUBLISHED BY  
Microsoft Press  
A division of Microsoft Corporation  
One Microsoft Way  
Redmond, Washington 98052-6399

Copyright © 2015 by Charlie Russel

All rights reserved. No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

Library of Congress Control Number: 2015936016  
ISBN: 978-1-5093-0065-5

Printed and bound in the United States of America.

First Printing

Microsoft Press books are available through booksellers and distributors worldwide. If you need support related to this book, email Microsoft Press Support at [mspinput@microsoft.com](mailto:mspinput@microsoft.com). Please tell us what you think of this book at <http://aka.ms/tellpress>.

This book is provided “as-is” and expresses the author’s views and opinions. The views, opinions and information expressed in this book, including URL and other Internet website references, may change without notice.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

Microsoft and the trademarks listed at <http://www.microsoft.com/about/legal/en/us/IntellectualProperty/Trademarks/EN-US.aspx> are trademarks of the Microsoft group of companies. All other marks are property of their respective owners.

**Acquisitions and Developmental Editor:** Karen Szall

**Editorial Production:** Online Training Solutions, Inc. (OTSI)

**Technical Reviewer:** David Coombes; Technical Review services provided by Content Master, a member of CM Group, Ltd.

**Copyeditor:** Kathy Krause (OTSI)

**Indexer:** Susie Carr (OTSI)

**Cover:** Twist Creative • Seattle

*I'd like to dedicate this book to my users: David R. Guy; Trey, Lord Barksdale; Dame Priscilla Katz; Alfredo "Alfie No-Nose" Fettuccine; Stanley T. Behr; Harold Catz; Dr. M. Eep; Ms. G. Gusano; Ms. E. Boots; and, finally, the hardest-working Cavalier King Charles Spaniel ever, Sir William Wallace.*

—CHARLIE RUSSEL



# Contents

---

<i>Introduction</i>	<i>xi</i>
<b>Chapter 1 Deploy your first forest and domain</b>	<b>1</b>
Before you start . . . . .	2
Prerequisites	2
Versions	2
Code	2
Deploy your first forest . . . . .	2
Configure the server IP address	3
Set the server name	6
Install Active Directory Domain Services	6
Create the forest (dcpromo)	7
Summary . . . . .	14
<b>Chapter 2 Manage DNS and DHCP</b>	<b>15</b>
Manage DNS zones . . . . .	16
Manage primary zones	17
Manage secondary zones	22
Manage stub zones	24
Configure conditional forwards	25
Manage zone delegation	26
Manage DNS records . . . . .	26
Create name (A and AAAA) resource records	28
Create CNAME resource records	33
Create MX resource records	34

---

**What do you think of this book? We want to hear from you!**

Microsoft is interested in hearing your feedback so we can improve our books and learning resources for you. To participate in a brief survey, please visit:

<http://aka.ms/tellpress>

Create additional resource records	34
Configure zone scavenging and aging	35
Configure record options including Time To Live (TTL) and weight	36
Manage DHCP	37
Deploy DHCP	37
Configure IPv4	38
Configure IPv6	40
Summary	41
<b>Chapter 3 Create and manage users and groups</b>	<b>43</b>
Create users	43
Create a single user	44
Add users in a batch	48
Create and manage groups	51
Create a new group	52
Add users to a group	52
Manage groups	54
Create and manage OUs	56
Create an OU	57
Add computers and users to an OU	58
Summary	62
<b>Chapter 4 Deploy additional domain controllers</b>	<b>63</b>
Deploy domain controllers	64
Configure networking	64
Install the Active Directory role on the server	67
Join the server to the domain	68
Promote a server to domain controller	68
Clone a domain controller	72
Verify the environment	72
Prepare the source domain controller	73
Create the cloned domain controller	77

Manage FSMO roles . . . . .	79
Transfer FSMO roles . . . . .	80
Seize FSMO roles . . . . .	82
Summary . . . . .	83
<b>Chapter 5 Deploy read-only domain controllers (RODCs)</b>	<b>85</b>
Prepare the forest and domain . . . . .	86
Staged deployment of an RODC . . . . .	87
Prepare the RODC account . . . . .	87
Prepare the RODC target server . . . . .	89
Deploy the RODC target server . . . . .	91
Non-staged deployment of an RODC . . . . .	94
Prepare the RODC target server . . . . .	94
Deploy the non-staged RODC target server . . . . .	97
Summary . . . . .	100
<b>Chapter 6 Deploy additional domains and forests</b>	<b>101</b>
Create a child domain . . . . .	102
Prepare the server . . . . .	102
Install the Active Directory Domain Services role . . . . .	105
Create the new domain . . . . .	105
Create a tree domain . . . . .	108
Prepare the server . . . . .	108
Install the Active Directory Domain Services role . . . . .	111
Create the new domain . . . . .	112
Create a new forest . . . . .	114
Configure networking . . . . .	114
Test the promotion to domain controller . . . . .	114
Deploy the new forest . . . . .	116
Create a trust . . . . .	117
Create a shortcut trust . . . . .	118
Create a forest trust . . . . .	120
Summary . . . . .	120



<b>Chapter 7</b>	<b>Configure service authentication and account policies</b>	<b>121</b>
	Manage service authentication .....	122
	Create service accounts	122
	Configure managed service accounts (MSAs)	126
	Configure group managed service accounts (gMSAs)	129
	Configure virtual accounts	135
	Configure account policies .....	135
	Configure domain user password policy	136
	Configure password settings objects (PSOs)	137
	Summary .....	142
<b>Chapter 8</b>	<b>Back up and restore AD DS</b>	<b>143</b>
	Back up Active Directory .....	144
	Windows Server Backup	144
	Create offline media	152
	Configure Active Directory snapshots	153
	Restore Active Directory .....	155
	Perform a non-authoritative restore	155
	Perform an authoritative restore	157
	Restore an object by using the Active Directory Recycle Bin	162
	Restore an object by using Active Directory snapshots	164
	Summary .....	166
<b>Chapter 9</b>	<b>Manage sites and replication</b>	<b>167</b>
	Configure sites .....	168
	Create a new site	168
	Create a replication subnet	169
	Rename a site	173
	Remove a site	174
	Configure Universal Group Membership Caching (UGMC)	175
	Create a site link	176

Manage replication. . . . .	178
Set the replication schedule	179
Change the replication server	181
Summary. . . . .	182
<b>Chapter 10 Deploy Active Directory in the cloud</b>	<b>183</b>
<i>Sidebar: Types of Active Directory in the cloud</i>	185
Install the Windows PowerShell Azure model . . . . .	185
Install the Windows PowerShell Azure module	186
Load the Windows PowerShell Azure module	187
Connect to an Azure account . . . . .	195
Authenticate to your Azure account	195
Set the current subscription	199
Create a VPN . . . . .	199
Create self-signed certificates	199
Create a point-to-site VPN	201
Create a virtual machine . . . . .	210
Connect to the subscription	210
Set a location	211
Provision a service	212
Provision a storage account	212
Create a virtual machine	213
Configure the domain controller . . . . .	218
Summary. . . . .	219
 <i>Index</i>	 221

---

**What do you think of this book? We want to hear from you!**

Microsoft is interested in hearing your feedback so we can improve our books and learning resources for you. To participate in a brief survey, please visit:

<http://aka.ms/tellpress>



# Introduction

---

If you're a Windows system administrator who is tired of click, click, clicking your way through screen after screen of wizards to do the same job over and over again, this book is for you. If you've been told that Windows PowerShell is hard, this book is definitely for you, because it simply isn't true. Nearly all the commands in this book are a line or two of Windows PowerShell code—one, maybe two cmdlets, with their everyday options laid out in a way that makes them easy to read and understand.

The target audience for this book is the working Windows system administrator, whether your domain is totally on premises, totally in the cloud, or in a hybrid environment. I'm not trying to teach you everything you need to know about Active Directory Domain Services, nor am I pretending to teach you Windows PowerShell. I assume you have at least some familiarity with both but want to take your everyday tasks to the next level.

I've taken care in writing this book to format the Windows PowerShell commands to improve understanding, not obfuscate, and the Microsoft Press production team has done a superb job of maintaining that. We could easily give you the command to promote a server to domain controller as shown here.

```
Install-ADSDomainController -SkipPreChecks -NoGlobalCatalog:$false -
CreateDnsDelegation:$false -CriticalReplicationOnly:$false -DatabasePath
"C:\Windows\NTDS" -DomainName "TreyResearch.net" -InstallDns:$true -LogPath
"C:\Windows\NTDS" -NoRebootOnCompletion:$false -SiteName "Default-First-Site-
Name" -SysvolPath "C:\Windows\SYSVOL" -Force:$true
```

But although that would be technically correct, and easy for me to create, it produces something that is at best daunting, and at worst useless. So, instead, I've chosen to format the Windows PowerShell commands to make them easier to read and follow. The same command, as you'll find it in Chapter 4, "Deploy additional domain controllers," is as shown here.

```
Install-ADSDomainController `
    -SkipPreChecks `
    -NoGlobalCatalog:$false `
    -CreateDnsDelegation:$false `
    -CriticalReplicationOnly:$false `
    -DatabasePath "C:\Windows\NTDS" `
    -DomainName "TreyResearch.net" `
    -InstallDns:$true `
    -LogPath "C:\Windows\NTDS" `
```

```
-NoRebootOnCompletion:$false `
-SiteName "Default-First-Site-Name" `
-SysvolPath "C:\Windows\SYSVOL" `
-Force:$true
```

Both commands produce exactly the same results, but by breaking the command up into multiple lines and using the Windows PowerShell end-of-line escape character—the backtick character (``)—I’ve made the second command much easier to read and understand.

Throughout this book, we use shaded text to show the output of commands. This gives you the output of commands without using a graphical screen shot in most cases. The command is shown in a fixed-width font, with the output being in a shaded fixed-width font, as shown here.

```
Add-ADGroupMember `
  -Identity "Cloneable Domain Controllers" `
  -Members (Get-ADComputer -Identity trey-dc-04).SAMAccountName `
  -PassThru
```

```
DistinguishedName : CN=Cloneable Domain
Controllers,CN=Users,DC=TreyResearch,DC=net
GroupCategory     : Security
GroupScope        : Global
Name              : Cloneable Domain Controllers
ObjectClass       : group
ObjectGUID        : b12b23c1-499b-4dbe-8206-846a17cd2df2
SamAccountName    : Cloneable Domain Controllers
SID               : S-1-5-21-910751839-3601328731-670513855-522
```

Finally, I’ve included not just the actual scripts from this book, but also all of the commands used in each chapter. They’re in the companion content that is available for download, as described in the next section.

## About the companion content

---

The companion content for this book can be downloaded from the following page:

*<http://aka.ms/ADPS/files>*

The companion content includes the following:

- All scripts in the book
- The Windows PowerShell commands from each chapter
- A sample netcfg file for setting up the virtual network described in Chapter 10, “Deploy Active Directory in the cloud”

## Acknowledgments

---

As only writers can fully appreciate, no book ever makes it into a reader’s hands without the work of many, many people, some of whom I’ll never know, but all of whose efforts I greatly appreciate. This is especially true with this book, which was written on a really tight schedule that stressed everyone in the process. I truly appreciate everything everyone did to make this book happen.

Of the people who worked on this book, or supported me during the process—those whom I do know—I’d like to sincerely thank Anne Hamilton, who has been a friend and an ally at Microsoft Press. My editor at Microsoft Press since at least Windows 2000 has been Karen Szall, and I couldn’t possibly have a better editor or friend. Karen, you are the absolute best, full stop.

At Online Training Solutions, Inc. (OTSI), Kathy Krause has been a superb editor with a light but deft and accurate touch, and her team—including Jean Trenary, Jaime Odell, Jeanne Craver, and Kerin Forsyth—have excelled. My sincere thanks. My excellent tech reviewer has been David Coombes, who has carefully gone over every command in the book, and more than once put me back on the right track. Thank you for doing so gently, but firmly. The thorough and accurate index is the hard work of Susie Carr at OTSI.

I’d also like to thank Gaby Kaplan of Microsoft, who has patiently taken my Windows PowerShell documentation bug reports and either fixed them instantly or sent them off to the correct person. She’s a perfect example of the dedication to perfection and community involvement that has made working with the Windows PowerShell team at Microsoft such a pleasure over the years.

From the wonderful group of Windows PowerShell MVPs at Microsoft that I’ve been so fortunate to know and work with over the years, I’d like to call out three names for special mention: Jeffrey Hicks, whose blog continues to provide useful answers and insights, and that specifically answered a conundrum that had me stumped; Richard Siddaway, for his amazing help with the problems of filtering in ADUser; and my friend Thomas Lee, whose blogs and script snippets have educated and enlightened me for years.

I'd also like to thank my co-workers at Kaseya, who have been supportive and understanding during the writing of this book. I'd especially like to thank my boss, Michael Duncan, who is a real pleasure to work for, and my fellow system administrators, Dan Lowry and Eugene Hoang. You guys do an amazing job and are great to know and work with.

Finally, my wife and frequent co-author, Sharon Crawford. Without you, this book would never have been completed. You make my life a joy.

## Free ebooks from Microsoft Press

---

From technical overviews to in-depth information on special topics, the free ebooks from Microsoft Press cover a wide range of topics. These ebooks are available in PDF, EPUB, and Mobi for Kindle formats, ready for you to download at:

*<http://aka.ms/mspressfree>*

Check back often to see what is new!

## Errata, updates, & book support

---

We've made every effort to ensure the accuracy of this book and its companion content. You can access updates to this book—in the form of a list of submitted errata and their related corrections—at:

*<http://aka.ms/ADPS/errata>*

If you discover an error that is not already listed, please submit it to us at the same page.

If you need additional support, email Microsoft Press Book Support at:

*[mspinput@microsoft.com](mailto:mspinput@microsoft.com)*

Please note that product support for Microsoft software and hardware is not offered through the previous addresses. For help with Microsoft software or hardware, go to:

*<http://support.microsoft.com>*

## **We want to hear from you**

---

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

*<http://aka.ms/tellpress>*

The survey is short, and we read every one of your comments and ideas. Thanks in advance for your input!

## **Stay in touch**

---

Let's keep the conversation going! We're on Twitter:

*<http://twitter.com/MicrosoftPress>*





## CHAPTER 1

# Deploy your first forest and domain

In this chapter, I cover how to create a new Active Directory Domain Services (AD DS) forest where one has never existed before. This is, in some ways, the easiest task you're likely to face, but it's also one where getting it right is *really* important. The decisions you make here will affect the entire organization for the life of this deployment.

### **Active Directory Windows PowerShell nouns used in this chapter:**

- ADDSDomainController
- ADDSForestInstallation
- ADDSForest
- ADRootDSE
- ADObject

### **Other Windows PowerShell commands used in this chapter:**

- Get-NetAdapter
- Get-Member
- Set-NetIPAddress
- New-NetIPAddress
- Set-DnsClientServerAddress
- Get-NetIPAddress
- Rename-Computer
- Install-WindowsFeature
- Get-Command
- Format-Table
- Update-Help
- ConvertTo-SecureString

## Before you start

---

This section sets some expectations. And yes, much of this has been covered in the introduction of the book, but in my experience most people don't read that. So I'll take a bit of liberty and do it again.

## Prerequisites

This book assumes that you know the basics of both Active Directory and Windows PowerShell. I won't attempt to teach you how to use either. But, that being said, I hope and expect you'll learn something about both of them.

## Versions

This book is being written against Windows Server Technical Preview, Build 9841. This includes Windows PowerShell 5.0, but no changes to Active Directory Domain Services (AD DS) beyond those in Windows Server 2012 R2 that affect the examples in the book. If I use a feature beyond that built into Windows 8.1 and Windows Server 2012 R2, I'll call it out explicitly. Most examples will work with Windows Server 2008 R2 and Windows Server 2012.

## Code

By its nature, this book includes a lot of code. Most is fairly basic—one or two lines of code, because most actions you need to do in AD DS are ones that lend themselves to a few commands in Windows PowerShell. Where the task requires a bit more, I give you a full script, complete with built-in comment-based help, as shown later in the `Get-myADVersion` script. Other scripts are a bit more casual and might not include full comment-based help. These scripts tend to be the kind of simple, one-off scripts that all Windows PowerShell users create to simplify their work. I don't include full and complex error-handling routines as part of the scripts—not that I don't think they're useful, but when performing actions against Active Directory, I really would prefer to have errors be errors and have the script fail, rather than hide any of that or try to recover and continue.

## Deploy your first forest

---

Most Windows system administrators will probably never have to create a new forest in an environment where there has never been one before. Most of us join a company and an environment that has been up and running for some time, and our tasks are focused on maintaining that existing environment—adding users and groups, adding domain controllers to existing domains, and even adding new domains to an existing forest. I'll cover all of those tasks in this book, and you can certainly jump ahead to the chapter that covers what you want to accomplish. But for those who are tasked with creating a new environment, it's important to do the job right, and that means planning *first*.

This is not a book on how to plan a new namespace and Active Directory forest. Instead of covering that here, I suggest that you read Chapters 3 and 4 of *Windows Server 2008 Administrator's Companion* (Microsoft Press, 2008). Yes, it's been a while since I wrote those chapters, but they're still valid today and will give you a solid understanding of the process.

Before you begin, make sure you have identified all the elements you'll need to configure as you set up the server you'll use to create your new forest and domain, and what the values for those are. The exact list you'll need will vary depending on the results of the preliminary planning you've done, and your network configuration, but it will likely include at least the following:

- Server IP address
- Server name
- Domain Name System (DNS) namespace for the root domain of the new forest
- Domain name for the root domain of the new forest
- DNS server type (Active Directory–integrated, or stand-alone)

A comment here about the server IP address: your domain controllers should ideally all use static IP addresses, but definitely your first domain controller should be at a fixed IP address.

## Configure the server IP address

You can configure the server's name before the IP address, but when you do, it costs an extra reboot because the name change requires a reboot, so I like to do the IP address first. Setting a fixed IP address for a computer requires four commands—one to get the name and index of the network adapter you're setting to a fixed IP address, and three to configure the settings for that adapter.

### Get the adapter alias and index

Before you can configure new settings for a network adapter, you need to know either the adapter's *interface alias* (name) or *interface index*. The interface alias corresponds to the name shown in the Network Connections dialog box (ncpa.cpl). To determine the interface alias and interface index, use the Get-NetAdapter cmdlet.

```
Get-NetAdapter
```

Name	InterfaceDescription	ifIndex	Status	MacAddress	LinkSpeed
10 Network	Microsoft Hyper-V Network Adapter #2	4	Up	00-15-5D-32-10-02	10 Gbps
50 Network	Microsoft Hyper-V Network Adapter	3	Disabled	00-15-5D-32-50-02	1 Gbps

The default output from `Get-NetAdapter` uses the `Name` column for the `InterfaceAlias` property and the `ifIndex` column for the `InterfaceIndex` property. To view all the properties and the actions associated with `Get-NetAdapter`, use the following.

```
Get-NetAdapter | Get-Member
```

## Set a fixed IP address

To set a fixed IP address for this first domain controller in the forest, you need to first disable Dynamic Host Configuration Protocol (DHCP) and then set the IPv4 and IPv6 addresses. For the lab network used in this book, I have chosen 192.168.10.0/24 as the IPv4 subnet, and 2001:db8:0:10::/64 as the IPv6 subnet.

To disable DHCP on the 10 Network adapter, use the following command.

```
Set-NetIPInterface -InterfaceAlias "10 Network" -DHCP Disabled -PassThru
```

The `Set-NetIPInterface` cmdlet is a quiet cmdlet that doesn't return anything by default, so I added the `-PassThru` parameter to have it report back on the status of the IP interface.

Next, set the static IPv4 address to 192.168.10.2 by using the following command.

```
New-NetIPAddress `
  -AddressFamily IPv4 `
  -InterfaceAlias "10 Network" `
  -IPAddress 192.168.10.2 `
  -PrefixLength 24 `
  -DefaultGateway 192.168.10.1
```

Now set the IPv6 address to 2001:db8:0:10::2 by using the following command.

```
New-NetIPAddress `
  -AddressFamily IPv6 `
  -InterfaceAlias "10 Network" `
  -IPAddress 2001:db8:0:10::2 `
  -PrefixLength 64 `
  -DefaultGateway 2001:db8:0:10::1
```

The `New-NetIPAddress` cmdlet automatically selects the IPv4 or IPv6 address family based on the settings in the command, so you can omit the `-AddressFamily` parameter from the preceding commands if you want.

## Set the DNS server addresses

The last part of setting a fixed IP address is to set the DNS server addresses. Because your first domain controller in the new forest should also be your DNS server, that's pretty easy to do by using the `Set-DnsClientServerAddress` cmdlet.

```
Set-DnsClientServerAddress `
    -InterfaceAlias "10 Network" `
    -ServerAddresses 192.168.10.2,2001:db8:0:10::2
```

So, when you pull all that together and run it on the first domain controller in your new forest, you can then run `Get-NetIPAddress` and get something like the following.

```
Get-NetIPAddress -InterfaceAlias "10 Network"
```

```
IPAddress      : 2001:db8:0:10::2
InterfaceIndex  : 4
InterfaceAlias  : 10 Network
AddressFamily   : IPv6
Type            : Unicast
PrefixLength    : 64
PrefixOrigin    : Manual
SuffixOrigin    : Manual
AddressState    : Preferred
ValidLifetime  : Infinite ([TimeSpan]::MaxValue)
PreferredLifetime : Infinite ([TimeSpan]::MaxValue)
SkipAsSource    : False
PolicyStore     : ActiveStore

IPAddress      : 192.168.10.2
InterfaceIndex  : 4
InterfaceAlias  : 10 Network
AddressFamily   : IPv4
Type            : Unicast
PrefixLength    : 24
PrefixOrigin    : Manual
SuffixOrigin    : Manual
AddressState    : Preferred
ValidLifetime  : Infinite ([TimeSpan]::MaxValue)
PreferredLifetime : Infinite ([TimeSpan]::MaxValue)
SkipAsSource    : False
PolicyStore     : ActiveStore
```

## Set the server name

Before you actually deploy your new forest, you should set the name of your domain controller to match your naming convention. Changing the name of a computer causes a reboot, which is why you should delay that change until after all the IP address setting is done. To change the name of the new server to `trej-dc-02`, use the `Rename-Computer` cmdlet by using the following syntax.

```
Rename-Computer -NewName trej-dc-02 -Restart -Force -PassThru
```

This changes the name of the server and automatically restarts it. The `-Force` parameter suppresses the confirmation prompt, and the `-PassThru` parameter returns the results of the command. After the server restarts, you're ready to actually deploy your forest.

## Install Active Directory Domain Services

Before you can promote the server to be a domain controller, you need to install the Active Directory Domain Services role on the server. Installing a role or feature uses the `Install-WindowsFeature` cmdlet. This cmdlet replaces the `Add-WindowsFeature` cmdlet used in Windows Server 2008 R2. For compatibility, `Add-WindowsFeature` is an alias to `Install-WindowsFeature`. The command to install AD DS, including the management tools required, is as follows.

```
Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools
```

This installs AD DS on the server and includes both the graphical and Windows PowerShell tools that are used to manage and deploy Active Directory. For the purposes of this book, this includes two Windows PowerShell modules—`ActiveDirectory` and `ADDSDeployment`.

**NOTE** The `Install-WindowsFeature` cmdlet includes additional parameters not shown here. The ones of most interest are the `-IncludeAllSubfeature`, `-Credential`, `-ComputerName`, and `-Vhd` parameters. The `-Vhd` parameter deserves some explanation. By using this parameter, you can use `Install-WindowsFeature` to add Windows Server roles and features to an offline VHD file, allowing you to “pre-load” features without having to bring the virtual machine (VM) online. The VHD file can be local or remote. If it is remote, the Universal Naming Convention (UNC) path to the VHD is the value of the parameter. When the `-Vhd` parameter is combined with the `-ComputerName` parameter, the VHD can actually be modified from the remote computer.

## Create the forest (dcpromo)

Beginning with Windows Server 2000, and right up until Windows Server 2012, the command-line way to create a new domain controller was to use the dcpromo command. But beginning with Windows Server 2012, dcpromo has been replaced with the ADDSDeployment module. This module supports remoting so that you can promote a server to a domain controller, create a new domain, or even create a new forest, without logging on to the server that is being promoted. To view the cmdlets in this module, use the following syntax.

```
Get-Command -Module ADDSDeployment | Format-Table Name
```

```
Name
----
Add-ADDSReadOnlyDomainControllerAccount
Install-ADSDomain
Install-ADSDomainController
Install-ADDSForest
Test-ADSDomainControllerInstallation
Test-ADSDomainControllerUninstallation
Test-ADSDomainInstallation
Test-ADDSForestInstallation
Test-ADDSReadOnlyDomainControllerAccountCreation
Uninstall-ADSDomainController
```

As you can tell, almost all of the various promote/demote/test possibilities are included in the module. The five Test cmdlets need a bit of explanation. Each of these cmdlets allows you to actually test whether all prerequisites are met before you run the Install or Add cmdlet of the same noun. This way you can fully test your environment before committing. The Install and Add nouns actually perform these same tests and will error out if any of them fail. However, the time to find out that you've got a problem is not the weekend you're actually performing the installation, but well before, so that you can correct any deficiencies and be prepared for success.

## Update Windows PowerShell help

Before you go any further, it's a good idea to update your Windows PowerShell help files. Unfortunately, there are only stub help files (man pages) included with Windows PowerShell. This allows Microsoft to update the help files on a regular basis, but it isn't terribly helpful if you're using an unfamiliar command. The only full help file included with Windows PowerShell is that for the Update-Help cmdlet.



You need to be running with Administrative privileges to update the help files. You can update directly from Microsoft (the default) or update from a network share. The basic command is the following.

```
Update-He1p
```

Yes, it is just that simple. This downloads and installs help files for all modules in the current session and for any modules found in the \$PSModulePath locations. If you run it on a computer that already has the help files installed, it will check the current version against the updated version and install only those that are new. You can install help files from a network share by using the -SourcePath parameter:

```
Update-He1p -SourcePath \\trey-dc-02\PSHe1p
```

It's a good idea to get in the habit of updating help files whenever you add new modules to a server. If you have servers that don't have Internet access, or if you just want to control your Internet bandwidth, you can use the Save-Help cmdlet to download and save the newest help files to a network share. The command to force an update to the current help files and then save them to the \\trey-dc-02\PSHelp share is the following.

```
Save-He1p -DestinationPath \\trey-dc-02\PSHe1p -force
```

## Test the forest creation

Before you start your weekend forest creation, only to discover in the middle of the process that you don't have the necessary prerequisites, it's a good practice to use the appropriate Test cmdlet to verify your environment. For creating the first forest in this book, that means using the Test-ADDSForestInstallation cmdlet. To test the trey-dc-02 server, which is sitting in a completely isolated lab environment and has no DNS on the network, use the Test-myForestCreate.ps1 script.

### **Test-myForestCreate.ps1**

```
Import-Module ADDSDeployment
Test-ADDSForestInstallation `
    -DomainName 'TreyResearch.net' `
    -DomainNetBiosName 'TREYRESEARCH' `
    -DomainMode 6 `
    -ForestMode 6 `
    -NoDnsOnNetwork `
    -NoRebootOnCompletion
```

This script imports the ADDSDeployment module into the current session and then tests the environment to find out whether installing the new forest will succeed. (And before I get comments—yes, I know that the Import-Module step is no longer required. But it's a good habit from the old days to explicitly load a nonstandard module when I know I'm going to need it.) The results of the test are shown in Figure 1-1.

```

Administrator@trej-dc-02 >
PS C:\temp> Test-myForestCreate.ps1
SafeModeAdministratorPassword: *****
Confirm SafeModeAdministratorPassword: *****
WARNING: Windows Server Technical Preview domain controllers have a default for the security setting named "Allow
cryptography algorithms compatible with Windows NT 4.0" that prevents weaker cryptography algorithms when establishing
security channel sessions.
For more information about this setting, see Knowledge Base article 942564
(http://go.microsoft.com/fwlink/?LinkId=104751).
WARNING: A delegation for this DNS server cannot be created because the authoritative parent zone cannot be found or it
does not run Windows DNS server. If you are integrating with an existing DNS infrastructure, you should manually
create a delegation to this DNS server in the parent zone to ensure reliable name resolution from outside the domain
"TreyResearch.net". Otherwise, no action is required.

Message                Context                RebootRequired        Status
-----                -
Operation completed succe... Test.VerifyDcPromoCore.DCP...        False                Success

PS C:\temp>

```

FIGURE 1-1 The results of Test-myForest.ps1

As you can tell, the Test-ADDSTreeInstallation cmdlet returns two warnings. One is about the security settings; it warns about compatibility with some older versions of Windows NT due to a change in the cryptography. This is normal and expected, and it can be ignored unless you have computers or devices on your network that require settings that are compatible with Windows NT 4.0. The second is a delegation warning for DNS. This is also expected in most cases. Neither warning is sufficient to stop the installation or create problems, so you're ready to proceed.

## Deploy the first domain controller and forest

At this point, you've configured your server, added the necessary Windows PowerShell modules and the Windows Server roles, and tested your environment. All is ready to do the actual initial deployment of your first domain controller and root AD DS forest.

The actual command to install the new forest and domain is nearly identical to the Test-ADDSTreeInstallation command in the Test-myForest script. The main difference is that this time, you *do* want to reboot the server when the installation is finished, and because you just ran the tests, you can skip them.

```

Install-ADDSTree `
  -DomainName 'TreyResearch.net' `
  -DomainNetBiosName 'TREYRESEARCH' `
  -DomainMode 6 `
  -ForestMode 6 `
  -NoDnsOnNetwork `
  -SkipPreChecks `
  -Force

```

The other thing added here is a `-Force` parameter to suppress any confirmation prompts. You'll still be prompted for the value of the Directory Services Restore Mode (DSRM) password. You can avoid even that by using the `-SafeModeAdministratorPassword` parameter with a `SecureString` value equivalent to your password. If you're automating a lot of forest (or domain) creations, such as in a lab environment, use this syntax to set the DSRM password to a value of `P@ssw0rd!`.

```
$pwdSS = ConvertTo-SecureString -String 'P@ssw0rd!' -AsPlainText -Force
```

**NOTE** This is a good time to point out the difference between single quotation marks and double quotation marks in Windows PowerShell. Both are used to identify strings, but a single quote doesn't allow the expansion or interpretation of special characters or variables inside the quotation marks, whereas double quotation marks do allow expansion. It's generally considered good practice to use single quotation marks unless you actually need variable expansion, but I don't always follow that practice. Here, however, it's a particularly good idea to use single quotation marks around a password string to avoid any interpretation of special characters.

The acceptable values for `ForestMode` and `DomainMode` are shown in Table 1-1.

**TABLE 1-1** Acceptable `DomainMode` and `ForestMode` values

Functional level	Numeric	String
Windows Server 2003	2	Win2003
Windows Server 2008	3	Win2008
Windows Server 2008 R2	4	Win2008R2
Windows Server 2012	5	Win2012
Windows Server 2012 R2	6	Win2012R2

The default forest functional level for Windows Server is typically the same as the Windows Server version, with the exception that the default for Windows Server 2008 R2 is a forest functional level of Windows Server 2003.

The domain functional level can never be less than the forest functional level, but it can be higher. If the `DomainMode` isn't specified, it is computed from the environment.

**MORE INFO** For more information about AD DS functional levels, see the "Understanding Active Directory Domain Services (AD DS) Functional Levels" TechNet article at <https://technet.microsoft.com/library/understanding-active-directory-functional-levels.aspx>.

When you create the new forest, the server is rebooted, and the only account active on the server is the TREYRESEARCH\Administrator account, which has the same password as the safe mode password you used with Install-ADDSTForest.

To find out what Forest Mode, Domain Mode, and Schema Version you've just created, use the following.

#### **Get-myADVersion.ps1**

```
<#
.Synopsis
Get the current Schema version and Forest and Domain Modes
.Description
The Get-myADVersion script queries the AD to discover the current AD schema version,
and the forest mode and domain mode. If run without parameters, it will query the
current AD context, or if a Domain Controller is specified, it will query against
that DC's context. Must be run as a user with sufficient privileges to query AD DS.
.Example
Get-myADVersion
Queries against the current AD context.
.Example
Get-myADVersion -DomainController Trey-DC-02
Gets the AD versions for the Domain Controller "Trey-DC-02"
.Parameter DomainController
Specifies the domain controller to query. This will change the response to match
the AD context of the DC.
.Inputs
[string]
.Notes
    Author: Charlie Russe1
    Copyright: 2015 by Charlie Russe1
           : Permission to use is granted but attribution is appreciated
    Initial: 3/7/2015 (cpr)
    ModHist:
           :
#>
[CmdletBinding()]
Param(
    [Parameter(Mandatory=$False,Position=0)]
    [string]
    $DomainController
)

if ($DomainController) {
    $AD = Get-ADRootDSE -Server $DomainController
    Get-ADObject $AD.SchemaNamingContext -Server $DomainController `
        -Property ObjectVersion
```

```

} else {
    $AD = Get-ADRootDSE
    Get-ADObject $AD.SchemaNamingContext -Property ObjectVersion
}
$Forest = $AD.ForestFunctionality
$Domain = $AD.DomainFunctionality

# Use a Here-String to print out the result.
$VersionCodes = @"

Forest: $Forest
Domain: $Domain

```

Where the Schema version is:

```

72 = Windows Server Technical Preview Build 9841
69 = Windows Server 2012 R2
56 = Windows Server 2012
47 = Windows Server 2008 R2
44 = Windows Server 2008
31 = Windows Server 2003 R2
30 = Windows Server 2003
13 = Windows 2000
"@
$VersionCodes

```

The result of running Get-myADVersion is shown in Figure 1-2.

```

Administrator@trej-dc-02 >
PS C:\temp> Get-myADVersion

DistinguishedName : CN=Schema,CN=Configuration,DC=TreyResearch,DC=net
Name              : Schema
ObjectClass       : d#0
ObjectGUID        : ec3054a6-d386-4c08-a76f-b31b426462ea
ObjectVersion     : 72

Forest: Windows2012R2Forest
Domain: Windows2012R2Domain

Where the Schema version is:
72 = Windows Server Technical Preview Build 9841
69 = Windows Server 2012 R2
56 = Windows Server 2012
47 = Windows Server 2008 R2
44 = Windows Server 2008
31 = Windows Server 2003 R2
30 = Windows Server 2003
13 = Windows 2000

PS C:\temp>

```

**FIGURE 1-2** Results showing that the schema version for Preview Build 9841 is 72

Install-ADDSForest has some additional options that might be useful in your environment and that allow you to tweak the initial configuration. Table 1-2 shows a fuller list of the options for Install-ADDSForest.

**TABLE 1-2** Key parameters for Install-ADDSForest

Parameter	Type	Description
-DomainName	String	The fully qualified domain name of the new domain (TreyResearch.net in this book's example).
[-CreateDnsDelegation]	Boolean	Attempts to create a DNS delegation to the new DNS server.
[-DatabasePath ]	String	The location to store the domain database. Must be a local fixed disk.
[-DnsDelegationCredential ]	PSCredential	A credential object with permission to create the DNS delegation.
[-DomainMode ]	DomainMode	The AD DS domain functional level of the new domain.
[-DomainNetbiosName ]	String	The NetBIOS name of the new domain (TREYRESEARCH in this book's example).
[-ForestMode ]	ForestMode	The AD DS forest functional level of the new forest.
[-Force]	Boolean	Suppresses confirmation prompts.
[-InstallDns]	Boolean	Installs Active Directory Integrated DNS server. Default value is calculated based on the environment.
[-LogPath ]	String	Path to the log of the install.
[-NoDnsOnNetwork]	Boolean	Specifies that there are no DNS servers present on the network. Active Directory Integrated DNS is installed, and the network adapter or adapters are configured to use 127.0.0.1 and ::1 as the DNS server.
[-NoRebootOnCompletion]	Boolean	Prevents the server from rebooting after the installation completes. Fair warning—the server is in an interim state and is not stable. Using this switch is really a bad idea.
[-SafeModeAdministratorPassword ]	SecureString	Sets the DSRM password. If it is not specified, the user is prompted for the password and a confirming password.
[-SkipAutoConfigureDns]	Boolean	Skips automatic configuration of DNS settings. Used if the DNS Server service is already installed.
[-SkipPreChecks]	Boolean	Doesn't test the environment to find out whether the installation will succeed. Only recommended when you're separately running Test-ADDSForestInstallation.
[-SysvolPath ]	String	Fully qualified local path to the fixed disk where the SYSVOL file is written.

## Summary

---

In this chapter, you learned how to use Windows PowerShell to create a new Active Directory Domain Services deployment with a new Active Directory forest and root domain. You learned how to configure a network adapter to use a fixed IP address, including setting the DNS server address. You renamed the server to a more human-friendly name that fits with your organizational naming convention, and you installed additional roles and features on the server.

After configuring the networking on the server, you tested your environment to ensure that you were fully prepared to deploy the new forest, and when the test was successful, you promoted the server to be the root domain controller in your new forest.

In the next chapter, you'll learn how to configure your DNS server, adding DNS zones and resource records, and you'll also learn how to configure DHCP entirely with Windows PowerShell.





# Create and manage users and groups

Now that we have a forest and domain, and we've got the basics of networking and name resolution sorted, the next step is to add some users to our domain. We'll start with adding a simple user, interactively, and then create a bunch of users by using a script and a comma-separated values (CSV) file. We'll create a new group and then add a group of users into that group, using a filter to ensure that we add the correct set of users. Then we'll create a new organizational unit (OU) and move users and computers into the OU. Pretty basic stuff, really, but essential for any domain administrator.

## **Active Directory Windows PowerShell nouns used in this chapter:**

- ADUser
- ADGroup
- ADGroupMember
- ADAccountPassword
- ADPrincipalGroupMembership
- ADObject
- ADComputer

## **Other Windows PowerShell commands used in this chapter:**

- Import-CSV
- ConvertTo-SecureString
- Get-Command
- Test-Path
- Read-Host
- Write-Host

## **Create users**

---

Use the `New-ADUser` cmdlet to create new users. Most user properties can be directly added by using the parameters of `New-ADUser` detailed in Table 3-1, shown later in this section. Those user attributes not explicitly available as direct parameters to `New-ADUser` can be added by using the `OtherAttributes` parameter, which accepts a hashtable of attribute names and values.

## Create a single user

The first thing you'll want to do for your new domain is create an administrative user that isn't "Administrator." That first Administrator account is sometimes referred to as the 500 account because the last three digits of its security identifier (SID) are 500, as we can tell from a quick `Get-ADUser`.

```
Get-ADUser -Identity Administrator
```

```
DistinguishedName : CN=Administrator,CN=Users,DC=TreyResearch,DC=net
Enabled           : True
GivenName        :
Name             : Administrator
ObjectClass      : user
ObjectGUID       : a196f5de-343f-48d5-8aab-5289bfa6fab6
SamAccountName   : Administrator
SID              : S-1-5-21-910751839-3601328731-670513855-500
Surname          :
UserPrincipalName :
```

The 500 account is a bit too well known to use for everyday administration and should be given a really long and onerous password that is locked away somewhere very secure and then left alone except in dire emergencies. So let's give ourselves a working administrative account, and then we'll change the password on the 500 account and retire it from everyday use.

To add a new user, use the `New-ADUser` cmdlet. There are three basic ways to use `New-ADUser`:

1. Create a user by specifying all details on the command line.
2. Create a user from a template object—either one you create or an existing user.
3. Use a CSV file to create multiple users from a list of users and properties.

We're going to use option #1 to create our first administrative user. We need to specify the settings for the new user at the command line. Then we need to add the user to the appropriate Active Directory Domain Services (AD DS) security groups. First, to create the user, "Charlie," use the following commands.

```
$SecurePW = Read-Host -Prompt "Enter a password" -asSecureString
New-ADUser -Name "Charlie Russel" `
    -AccountPassword $SecurePW `
    -SamAccountName 'Charlie' `
    -DisplayName 'Charlie Russel' `
    -EmailAddress 'Charlie@TreyResearch.net' `
    -Enabled $True `
    -GivenName 'Charlie' `
    -PassThru `
    -PasswordNeverExpires $True `
    -Surname 'Russel' `
    -UserPrincipalName 'Charlie'
```

The Read-Host in the previous code prompts for a password and masks what the user enters, and the result of the New-ADUser command is displayed at the console because I used the -PassThru parameter, as shown in Figure 3-1.

```

Administrator@trej-dc-02 >
PS C:\> $SecurePw = Read-Host -Prompt "Enter a password" -asSecureString
Enter a password: *****
PS C:\> New-ADUser -Name "Charlie Russel" `
>>     -AccountPassword $SecurePw `
>>     -SamAccountName 'Charlie' `
>>     -DisplayName 'Charlie Russel' `
>>     -EmailAddress 'Charlie@TreyResearch.net' `
>>     -Enabled $True `
>>     -GivenName 'Charlie' `
>>     -PassThru `
>>     -PasswordNeverExpires $True `
>>     -Surname 'Russel' `
>>     -UserPrincipalName 'Charlie'
>>

DistinguishedName : CN=Charlie Russel,CN=Users,DC=TreyResearch,DC=net
Enabled           : True
GivenName        : Charlie
Name             : Charlie Russel
ObjectClass      : user
ObjectGUID       : e59556ce-bbeb-4f59-a9bb-f84bbc2855b4
SamAccountName   : Charlie
SID              : S-1-5-21-910751839-3601328731-670513855-1110
Surname          : Russel
UserPrincipalName : Charlie

PS C:\>

```

**FIGURE 3-1** Creating a new user by using New-ADUser

This creates our first user but doesn't make the user a member of any domain security groups except Domain Users, the default. To add the user to security groups, we need to use the Add-ADGroupMember cmdlet. And because the goal is to give Charlie the same set of security groups as the Administrator account, we'll use Windows PowerShell to get the list of security groups that the Administrator is a member of, and then loop through the list and add Charlie to each of the groups.

```

$SuperUserGroups = @(
$SuperUserGroups = (Get-ADUser -Identity "Administrator" -Properties *).MemberOf

ForEach ($Group in $SuperUserGroups) {
    Add-ADGroupMember -Identity $Group -Members "Charlie"
}

(Get-ADUser -Identity Charlie -Properties *).MemberOf

```

```

CN=Group Policy Creator Owners,CN=Users,DC=TreyResearch,DC=net
CN=Domain Admins,CN=Users,DC=TreyResearch,DC=net
CN=Enterprise Admins,CN=Users,DC=TreyResearch,DC=net
CN=Schema Admins,CN=Users,DC=TreyResearch,DC=net
CN=Administrators,CN=Builtin,DC=TreyResearch,DC=net

```

As we can tell from the `Get-ADUser` command in the previous code, the account Charlie is now a member of five security groups: Group Policy Creator Owners, Domain Admins, Enterprise Admins, Schema Admins, and Administrators. These are the same security groups to which the Administrator account belongs. We'll want to come back to AD DS groups later, but let's focus on users first.

In the creation of this first user, we used the most common parameters of the `New-ADUser` cmdlet, but they're only a fraction of the options available. Your situation might well require you to add significantly more information to each AD DS account. The available parameters for `New-ADUser` that relate to users are listed in Table 3-1.

**TABLE 3-1** The user property parameters of `New-ADUser`

Parameter	Type
Name	String
AccountExpirationDate	Datetime
AccountNotDelegated	Boolean
AccountPassword	SecureString
AllowReversiblePasswordEncryption	Boolean
AuthenticationPolicy	ADAuthenticationPolicy
AuthenticationPolicySilo	ADAuthenticationPolicySilo
AuthType	ADAuthType
CannotChangePassword	Boolean
Certificates	X509Certificate[]
ChangePasswordAtLogon	Boolean
City	String
Company	String
CompoundIdentitySupported	Boolean
Country	String
Credential	PSCredential
Department	String
Description	String
DisplayName	String
Division	String
EmailAddress	String
EmployeeID	String
EmployeeNumber	String

Parameter	Type
Enabled	Boolean
Fax	String
GivenName	String
HomeDirectory	String
HomeDrive	String
HomePage	String
HomePhone	String
Initials	String
Instance	ADUser
KerberosEncryptionType	ADKerberosEncryptionType
LogonWorkstations	String
Manager	ADUser
MobilePhone	String
Office	String
OfficePhone	String
Organization	String
OtherAttributes	Hashtable
OtherName	String
PassThru	Switch
PasswordNeverExpires	Boolean
PasswordNotRequired	Boolean
Path	String
POBox	String
PostalCode	String
PrincipalsAllowedToDelegateToAccount	ADPrincipal[]
ProfilePath	String
SamAccountName	String
ScriptPath	String
Server	String
ServicePrincipalNames	String[]
SmartcardLogonRequired	Boolean

Parameter	Type
State	String
StreetAddress	String
Surname	String
Title	String
TrustedForDelegation	Boolean
Type	String
UserPrincipalName	String

**NOTE** In this table of parameters, and in others throughout the book, I've deliberately ignored the parameters that don't directly relate to the object we're working with. This means I haven't included Common Parameters, nor have I included Confirm or WhatIf parameters.

## Add users in a batch

There are multiple ways to add users in a batch, but probably the simplest is to use a CSV file. You can easily create the CSV file in Microsoft Excel or any plain text editor, and then use Windows PowerShell to read the values in the CSV file and add the users. In my lab, all my animals have their own domain accounts, so I'll use them to show how to quickly and easily create new users. All are initially created as Domain Users, with a default password, and then one account gets elevated and prompts for a password. The list of users and their basic properties are in the following code.

### **TreyUsers.csv**

```
Name,GivenName,Surname,DisplayName,SAMAccountName,Description
David Guy,David,Guy,Dave R. Guy,Dave,Customer Appreciation Manager
Alfredo Fettucine,Alfredo,Fettucine,Alfie NoNose,Alfie,Shop Foreman
Stanley Behr,Stanley,Behr,Stanley T. Behr, Stanley,WebMaster
Priscilla Catz,Priscilla,Catz,Dame Priscilla,Priscilla,Shop Steward
Harold Catz,Harold,Catz,Harold S. Catz,Harold,Engineering Manager
William Wallace,William,Wallace,Sir William Wallace,Wally,Marketing Manager
Trey Barksdale,Trey,Barksdale,Lord Barksalot,Trey,Sales Manager
Charlie Russel,Charlie,Russel,Charlie Russel,Charlie,SuperUser Account
```

As you can tell, I've only used the most basic information for each new user. To read the CSV file, use the Import-CSV cmdlet, and then loop through each user from the CSV file and create the user with New-ADUser by using a basic ForEach loop.

## Create-TreyUsers.ps1

```
<#
.Synopsis
Creates the TreyResearch.net users
.Description
Create-TreyUsers reads a CSV file to create an array of users. The users are then added
to the users container in Active Directory. Additionally, Create-TreyUsers adds the
user Charlie to the same AD DS Groups as the Administrator account.
.Example
Create-TreyUsers
Creates AD Accounts for the users in the default "TreyUsers.csv" source file
.Example
Create-TreyUsers -Path "C:\temp\NewUsers.txt"
Creates AD accounts for the users listed in the file C:\temp\NewUsers.txt"
.Parameter Path
The path to the input CSV file. The default value is ".\TreyUsers.csv".
.Inputs
[string]
.Notes
    Author: Charlie Russel
    Copyright: 2015 by Charlie Russel
           : Permission to use is granted but attribution is appreciated
    Initial: 3/26/2015 (cpr)
    ModHist:
           :
#>
[CmdletBinding()]
Param(
    [Parameter(Mandatory=$False,Position=0)]
    [string]
    $Path = ".\TreyUsers.csv"
)

$TreyUsers = @()
If (Test-Path $Path ) {
    $TreyUsers = Import-CSV $Path
} else {
    Throw "This script requires a CSV file with user names and properties."
}

ForEach ($user in $TreyUsers ) {
    New-AdUser -DisplayName $User.DisplayName `
              -GivenName $user.GivenName `
              -Name $User.Name `
              -SurName $User.SurName `
              -SAMAccountName $User.SAMAccountName `
```

```

-Enabled $True `
-PasswordNeverExpires $true `
-UserPrincipalName $user.SAMAccountName `
-AccountPassword (ConvertTo-SecureString -AsPlainText -Force -String
"P@ssw0rd!" )
If ($User.SAMAccountName -eq "Charlie" ) {
    $cprpwd = Read-Host -Prompt 'Enter Password for account: Charlie' -AsSecureString
    Set-ADAccountPassword -Identity Charlie -NewPassword $cprpwd -Reset
    $SuperUserGroups = @(
    $SuperUserGroups = (Get-ADUser -Identity "Administrator" -Properties * ).MemberOf

    ForEach ($Group in $SuperUserGroups ) {
        Add-ADGroupMember -Identity $Group -Members "Charlie"
    }
    Write-Host "The user $User.SAMAccountName has been added to the following AD
Groups: "
    (Get-ADUser -Identity $user.SAMAccountName -Properties * ).MemberOf
}
}
}

```

**NOTE** As you'll notice, I've included the same superuser account as in the previous section. If you've already added that account, just change the account name and details or remove the account from the list.

When we run the Create-TreyUsers script, we get output only about the superuser account that was created, as shown in Figure 3-2.

```

Administrator@trej-dc-02 >
PS C:\Download\build> .\Create-TreyUsers.ps1 -Path .\TreyUsers.csv
Enter Password for account: Charlie: *****
The user @(Name=Charlie Russel; GivenName=Charlie; Surname=Russel; DisplayName=Charlie Russel; SAMAccountName=Charlie; Description=SuperUser Account).SAMAccountName has been added to the following AD Groups:
CN=Group Policy Creator Owners,CN=Users,DC=TrayResearch,DC=net
CN=Domain Admins,CN=Users,DC=TrayResearch,DC=net
CN=Enterprise Admins,CN=Users,DC=TrayResearch,DC=net
CN=Schema Admins,CN=Users,DC=TrayResearch,DC=net
CN=Administrators,CN=Builtin,DC=TrayResearch,DC=net
PS C:\Download\build>

```

**FIGURE 3-2** Creating multiple AD DS users from a CSV file

If you want more detail about the individual accounts that you created, modify the New-ADUser command in the script to include the PassThru parameter. With that change, though, you'll get a lot more detail than you likely want. Instead, try a quick one-line search to find the users.



```
(Get-ADUser -Filter {Enabled -eq "True"} -Properties DisplayName).DisplayName
```

```
Dave R. Guy  
Alfie NoNose  
Stanley T. Behr  
Dame Priscilla  
Harold S. Catz  
Sir William Wallace  
Lord Barksalot  
Charlie Russe1
```

Now that's just introduced a whole new set of issues with the Filter parameter. I'll cover filters, both traditional Windows PowerShell filters as we used here and LDAP filters, later in the "Manage groups" section, but for the moment let's examine this particular one-line search. The goal of the search is to get a list of the users we just created. Get-ADUser is the cmdlet to use to get users, but we only want to get a list of users that are actually enabled, to avoid accounts like the Guest account and some other special accounts. To get the DisplayName value, we need to add that property to the list of properties returned by Get-ADUser because it isn't part of the default properties.

## Create and manage groups

---

Use the ADGroup set of cmdlets to create, delete, modify, or list Active Directory groups, and either the ADGroupMember set of cmdlets or the ADPrincipalGroupMembership set of cmdlets to add, remove, and list the members of an Active Directory group. By using the ADGroupMember cmdlets, you add or remove one or more users, groups, service accounts, or computers to or from a group, whereas with the ADPrincipalGroupMembership cmdlets you add or remove a user, group, service account, or computer to or from one or more groups. Or, to try to make that a little clearer—if you want to add many objects into one group, use Add-ADGroupMember, but if you want to add one object into many groups, use Add-ADPrincipalGroupMembership. Or you can ignore one or the other set of cmdlets and use looping to accomplish the same thing, as I did earlier when I added the user Charlie into multiple groups by using the Add-ADGroupMember cmdlet and a ForEach loop.

In AD DS, two types of groups are supported: security groups and distribution groups. And there are three scope levels for each: Domain Local, Global, and Universal. To demonstrate how these cmdlets work, let's create a new group, Accounting Users, as a security group with Universal scope, and add a couple of users to the group. Then we'll search to get a list of users in the group, by using both standard Windows PowerShell filtering and LDAP filtering.

## Create a new group

Creating a new group is easy and uses the same basic techniques as creating a new user. The difference is that there are far fewer properties and parameters to creating a new group. For example, use the following command to create a new security group called Accounting Users and give that group Universal scope.

```
New-ADGroup -Name 'Accounting Users' `
            -Description 'Security Group for all accounting users' `
            -DisplayName 'Accounting Users' `
            -GroupCategory Security `
            -GroupScope Universal `
            -SAMAccountName 'AccountingUsers' `
            -PassThru
```

The results of this command are shown in Figure 3-3. Notice that even though we didn't specify the full path where we wanted to create the Accounting Users group, Windows PowerShell defaulted to putting the group in the Users container. To override that default, specify the Path parameter. Windows PowerShell will use the default container for your domain if you don't specify a path.



```
Administrator@trej-dc-02 >
PS C:\> New-ADGroup -Name 'Accounting Users' `
>> -Description 'Security Group for all accounting users' `
>> -DisplayName 'Accounting Users' `
>> -GroupCategory Security `
>> -GroupScope Universal `
>> -SAMAccountName 'AccountingUsers' `
>> -PassThru
>>

DistinguishedName : CN=Accounting Users,CN=Users,DC=TreyResearch,DC=net
GroupCategory     : Security
GroupScope        : Universal
Name              : Accounting Users
ObjectClass       : group
ObjectGUID        : d4629307-b10f-4342-80a5-e192d430bf8c
SamAccountName    : AccountingUsers
SID               : S-1-5-21-910751839-3601328731-670513855-1137

PS C:\>
```

FIGURE 3-3 Adding a new AD DS security group

## Add users to a group

Let's start by adding a couple of members to the Accounting Users group we just created. For this, because we're adding multiple users to a single group, we'll use the Add-ADGroupMember cmdlet.

Add-ADGroupMember has the following syntax.

```
Add-ADGroupMember [-Identity] <ADGroup> [-Members] <ADPrincipal[]>
[-AuthType {Negotiate | Basic}] [-Credential PSCredential]
[-Partition <String>] [-PassThru] [-Server <String>]
[-Confirm] [-WhatIf] [<CommonParameters>]
```

The Identity parameter accepts a Distinguished Name (DN), GUID, security identifier (SID) or SAM account name to identify which group you want to add members to. The Members parameter accepts an *array* of new members that you want to add to the group. The new members can be identified by the same methods as the group identifier, but the parameter also accepts user, computer, and group object variables that identify the members to be added. You *cannot*, however, pass objects to Add-ADGroupMember through the pipeline.

To add Dave R. Guy and Stanley T. Behr to the Accounting Users group, use the following command.

```
Add-ADGroupMember -Identity AccountingUsers -Members Dave,Stanley -PassThru
```

```
DistinguishedName : CN=Accounting Users,CN=Users,DC=TreyResearch,DC=net
GroupCategory     : Security
GroupScope        : Universal
Name              : Accounting Users
ObjectClass       : group
ObjectGUID        : d4629307-b10f-4342-80a5-e192d430bf8c
SamAccountName    : AccountingUsers
SID               : S-1-5-21-910751839-3601328731-670513855-1137
```

To verify that the members were added, because the PassThru parameter doesn't really help with that, use the Get-ADGroupMember cmdlet.

```
Get-ADGroupMember -Identity AccountingUsers
```

```
distinguishedName : CN=Stanley Behr,CN=Users,DC=TreyResearch,DC=net
name               : Stanley Behr
objectClass        : user
objectGUID         : 17527a2f-2710-49d7-ad6d-ce6342bb8c63
SamAccountName     : Stanley
SID                : S-1-5-21-910751839-3601328731-670513855-1131

distinguishedName : CN=David Guy,CN=Users,DC=TreyResearch,DC=net
name               : David Guy
objectClass        : user
objectGUID         : 93534ac0-bbd4-4a29-aae0-470b8e604b18
SamAccountName     : Dave
SID                : S-1-5-21-910751839-3601328731-670513855-1129
```

## Manage groups

Now, let's take this a bit further. Let's create another new security group for management. We'll call the group Managers, and we'll use the Description property to add members to the group. So, first create the group by using New-ADGroup.

```
New-ADGroup -Name 'Managers' `
            -Description 'Security Group for all Managers' `
            -DisplayName 'Managers' `
            -GroupCategory Security `
            -GroupScope Universal `
            -SAMAccountName 'Managers' `
            -PassThru
```

```
DistinguishedName : CN=Managers,CN=Users,DC=TreyResearch,DC=net
GroupCategory      : Security
GroupScope         : Universal
Name               : Managers
ObjectClass        : group
ObjectGUID         : 625b4911-301a-4249-8c39-40b88734a124
SamAccountName     : Managers
SID                : S-1-5-21-910751839-3601328731-670513855-1138
```

Now we need to select just the users who are managers to add to the group. We can do that by using the Description property, because we populated that when we created the users, and we know it includes "Manager" in the description for those who are managers. This would be easy if we could just pass the results of Get-ADUser directly through the pipeline to Add-ADGroupMember, but we can't. So, instead, we'll create an array of SAM account names from the results of Get-ADUser.

```
$ManagerArray = (Get-ADUser -Filter {Description -like "*Manager*"} `
                 -Properties Description).SAMAccountName
```

Now we'll use the \$ManagerArray variable with Add-ADGroupMember.

```
Add-ADGroupMember -Identity "Managers" -Members $ManagerArray -PassThru
```

And finally, to confirm the identity of who we added, use this.

```
Get-ADGroupMember -Identity Managers | ft -auto SAMAccountName,Name
```

```
SAMAccountName Name
-----
Trey            Trey Barksdale
Wally           William Wallace
Harold          Harold Catz
Dave            David Guy
```

But there's a problem with that—it really doesn't confirm that the users we added were managers. We could try changing that Format-Table command to the following.

```
ft -auto SAMAccountName,Name,Description
```

Unfortunately, that just yields an empty column for the Description field. And we can understand why with this.

```
Get-ADGroupMember -Identity Managers | Get-Member
```

```

      TypeName: Microsoft.ActiveDirectory.Management.ADPrincipal

Name                MemberType          Definition
----                -
Contains            Method              bool Contains(string propertyName)
Equals              Method              bool Equals(System.Object obj)
GetEnumerator        Method              System.Collections.IDictionaryEnumerator
GetEnumerator()
GetHashCode          Method              int GetHashCode()
GetType             Method              type GetType()
ToString            Method              string ToString()
Item                ParameterizedProperty
Microsoft.ActiveDirectory.Management.ADPropertyValueCollection Item(string p...
distinguishedName Property           System.String distinguishedName {get;set;}
name                Property           System.String name {get;}
objectClass         Property           System.String objectClass {get;set;}
objectGUID          Property           System.Nullable`1[[System.Guid, mscorlib,
Version=4.0.0.0, Culture=neutral, ...
SamAccountName     Property           System.String SamAccountName {get;set;}
SID                Property           System.Security.Principal.SecurityIdentifier
SID {get;set;}

```

We can't add a -Properties Description to the Get-ADGroupMember, because it doesn't support that parameter, so instead, we pass the results through Get-ADUser, which does support the Properties parameter, and now we get the following.

```

Get-ADGroupMember -Identity Managers `
    | Get-ADUser -Properties Description `
    | Format-Table -auto SAMAccountName,Name,Description

```

SAMAccountName	Name	Description
Trey	Trey Barksdale	Sales Manager
Wally	William Wallace	Marketing Manager
Harold	Harold Catz	Engineering Manager
Dave	David Guy	Customer Appreciation Manager

That worked. Now we can clearly tell that each of the members of the Managers group is described as a manager in Active Directory.

How about adding a user to multiple groups at a time? We saw earlier in the “Create users” section that we could do that with a loop, but wouldn’t it be more efficient to just do it in a single command? Let’s give Alfie the power he’s always wanted and make him a superuser, just like Charlie. And, instead of looping, we’ll use the Add-ADPrincipalGroupMembership cmdlet.

```
$Groups = (Get-ADUser -Identity Charlie -Properties *).MemberOf
Add-ADPrincipalGroupMembership -Identity Alfie -MemberOf $Groups
```

```
(Get-ADUser -Identity Alfie -Properties MemberOf).MemberOf
```

```
CN=Group Policy Creator Owners,CN=Users,DC=TreyResearch,DC=net
CN=Domain Admins,CN=Users,DC=TreyResearch,DC=net
CN=Enterprise Admins,CN=Users,DC=TreyResearch,DC=net
CN=Schema Admins,CN=Users,DC=TreyResearch,DC=net
CN=Administrators,CN=Builtin,DC=TreyResearch,DC=net
```

Now Alfie has his wish; he’s a superuser. But really, that’s more than we think he should have, so instead, we’ll just give him basic Domain Admins membership by removing him from the groups he really shouldn’t be in.

```
Remove-ADPrincipalGroupMembership -Identity Alfie `
    -MemberOf "Enterprise Admins",`
    "Schema Admins",`
    "Group Policy Creator Owners" `
    -PassThru
```

And after we confirm that we really want to do it, Alfie has been reduced to a more reasonable level, as shown with the following.

```
(Get-ADUser -Identity Alfie -Properties MemberOf).MemberOf
```

```
CN=Domain Admins,CN=Users,DC=TreyResearch,DC=net
CN=Administrators,CN=Builtin,DC=TreyResearch,DC=net
```

## Create and manage OUs

---

Organizational units, or OUs, are used to segregate groups of users, computers, or other objects in Active Directory without the overhead of creating a whole new domain for them. You can apply different group policies to different OUs and have different password requirements.

## Create an OU

Use the `New-ADOrganizationalUnit` cmdlet to create a new OU. The cmdlet parameters can be used to set commonly used properties of OUs, such as `DisplayName`, `Description`, and `ProtectedFromAccidentalDeletion`. The only required parameter is the `Name` parameter.

For properties that aren't covered by the cmdlet parameters shown in Table 3-2, use the `OtherAttributes` parameter. The `OtherAttributes` parameter accepts a hashtable with property name and property value pairs.

**TABLE 3-2** The parameters of `New-ADOrganizationalUnit`

Parameter	Type
<code>Name</code>	String
<code>City</code>	String
<code>Country</code>	String
<code>Credential</code>	PSCredential
<code>Description</code>	String
<code>DisplayName</code>	String
<code>Instance</code>	ADOrganizationalUnit
<code>ManagedBy</code>	ADPrincipal
<code>OtherAttributes</code>	Hashtable
<code>PassThru</code>	Toggle
<code>Path</code>	String
<code>PostalCode</code>	String
<code>ProtectedFromAccidentalDeletion</code>	Boolean
<code>Server</code>	String
<code>State</code>	String
<code>StreetAddress</code>	String
<code>Name</code>	String

There are three basic ways to use `New-ADOrganizationalUnit`:

1. Create an OU by specifying all details on the command line.
2. Create an OU from a template object—either one you create or an existing OU.
3. Use a CSV file to create multiple OUs from a list of OUs and properties.

To create a new Engineering OU for our TreyResearch.net domain, use the following command.

```
New-ADOrganizationalUnit -Name Engineering `
                        -Description 'Engineering department users and computers' `
                        -DisplayName 'Engineering Department' `
                        -ProtectedFromAccidentalDeletion $True `
                        -Path "DC=TreyResearch,DC=NET" `
                        -PassThru
```

Note that the path specified is actually the default path, so we could have skipped that parameter, and the same is true for the ProtectedFromAccidentalDeletion parameter, which defaults to True. Because we used the -PassThru parameter, the command returned the following.

```
City           :
Country        :
DistinguishedName : OU=Engineering,DC=TreyResearch,DC=NET
LinkedGroupPolicyObjects : {}
ManagedBy     :
Name           : Engineering
ObjectClass    : organizationalUnit
ObjectGUID     : c2b42af8-a80b-48c1-949d-c8dbd6d60ee9
PostalCode     :
State         :
StreetAddress  :
```

And, as we can tell in Figure 3-4, the new Engineering OU is created in the root of the TreyResearch.net domain tree.

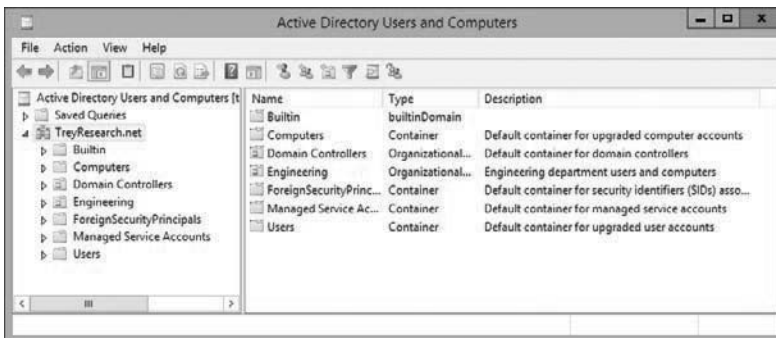


FIGURE 3-4 The TreyResearch.net domain, showing the new Engineering OU

## Add computers and users to an OU

Now that we have an Engineering OU, we should move our engineering users and computers into that OU. You might expect that there would be a Move-ADUser cmdlet, but there isn't, and you might even quite reasonably expect that you could use Set-ADUser with a Path



parameter to do the job. But no, there isn't a Path parameter and you can't move users that way either. After poking around a bit, however, it occurred to me to use Windows PowerShell to help me find the solution.

```
Get-Command -Module ActiveDirectory -Verb Move | ft -auto CommandType,Name
```

```
CommandType      Name
-----
Move-ADDirectoryServer
Move-ADDirectoryServerOperationMasterRole
Move-ADObject
```

Well, I don't want to move the directory server, nor the Flexible Single Master Operation (FSMO) roles, at least not right now, so those won't help. But users and computers are just a form of Active Directory object, so that last item looks promising. Let's find out what its syntax is.

```
syntax Move-ADObject
```

Syntax for Move-ADObject is:

```
Move-ADObject [-Identity] <ADObject> [-TargetPath] <string> [-WhatIf]
[-Confirm] [-AuthType ADAAuthType] [-Credential <pscredential>]
[-Partition <string>] [-PassThru] [-Server <string>]
[-TargetServer <string>] [<CommonParameters>]
```

That looks like it should do what we want. We need to specify the identity of the object we want to move, and the target path we want to move it to. And it supports a WhatIf parameter to make sure it's going to do what we expect. So, I remember that one of my users is an engineering manager, but which one? Well, let's find out.

```
Get-ADUser -Filter {Description -like "*Engineering*" }
```

```
DistinguishedName : CN=Harold Catz,CN=Users,DC=TreyResearch,DC=net
Enabled            : True
GivenName         : Harold
Name              : Harold Catz
ObjectClass       : user
ObjectGUID        : 944bb855-0342-4875-a8d2-8447ab6f93e5
SamAccountName    : Harold
SID               : S-1-5-21-910751839-3601328731-670513855-1133
Surname           : Catz
UserPrincipalName : Harold
```

Ah, yes, Harold. Of course. So, now that we know who we want to move, let's verify where we want to move him to.

```
Get-ADOrganizationalUnit -Filter {Name -eq "Engineering" }
```

```
City           :  
Country        :  
DistinguishedName : OU=Engineering,DC=TreyResearch,DC=net  
LinkedGroupPolicyObjects : {}  
ManagedBy     :  
Name           : Engineering  
ObjectClass    : organizationalUnit  
ObjectGUID     : c2b42af8-a80b-48c1-949d-c8dbd6d60ee9  
PostalCode     :  
State          :  
StreetAddress  :
```

The Distinguished Name is the target path for our move, so let's check that we've got everything as we want it.

```
Get-ADUser -Filter {Description -like "*Engineering*"} | Move-ADObject `
    -TargetPath (Get-ADOrganizationalUnit -Filter {Name -eq "Engineering" }) `
    -WhatIf
```

```
What if: Performing the operation "Move" on target "CN=Harold
Catz,CN=Users,DC=TreyResearch,DC=net".
```

That looks like we're moving Harold, which was the plan, so we remove the WhatIf, and issue the command again.

```
Get-ADUser -Filter {Description -like "*Engineering*"} | Move-ADObject `
    -TargetPath (Get-ADOrganizationalUnit -Filter {Name -eq "Engineering" })
```

Oops, we forgot to include the `-PassThru` parameter, so our move happened silently. No problem, let's just verify that the user Harold is in the correct OU.

```
Get-ADUser -Identity Harold
```

```
DistinguishedName : CN=Harold Catz,OU=Engineering,DC=TreyResearch,DC=net  
Enabled           : True  
GivenName         : Harold  
Name              : Harold Catz  
ObjectClass       : user  
ObjectGUID        : 944bb855-0342-4875-a8d2-8447ab6f93e5  
SamAccountName    : Harold  
SID               : S-1-5-21-910751839-3601328731-670513855-1133  
Surname           : Catz  
UserPrincipalName : Harold
```

And the DistinguishedName property shows that he is in the Engineering OU. Good. Now, let's just move Harold's desktop over to the same OU. A quick check finds that TREY-DESK-22 is assigned to Harold.

```
Get-ADComputer -Filter {Description -like "*Harold*" }
```

```
DistinguishedName : CN=TREY-DESK-22,CN=Computers,DC=TreyResearch,DC=net
DNSHostName       : trey-desk-22.TreyResearch.net
Enabled           : True
Name              : TREY-DESK-22
ObjectClass       : computer
ObjectGUID        : 46df71bd-ba88-4b26-9091-b8db6e07261a
SamAccountName    : TREY-DESK-22$
SID               : S-1-5-21-910751839-3601328731-670513855-1141
UserPrincipalName :
```

Looks like Harold only has one computer, so let's do it by simply specifying the identity of the computer we want to move. Move-ADObject accepts the DN or the GUID for the Identity parameter, or the result of Get-ADUser, Get-ADGroup, Get-ADComputer, Get-ADServiceAccount, Get-ADOrganizationalUnit, or Get-ADFineGrainedPasswordPolicy. We've got both the DN and the GUID in the output from your Get-ADComputer, so after a bit of copy and paste we get the following.

```
Move-ADObject -Identity "46df71bd-ba88-4b26-9091-b8db6e07261a" `
              -TargetPath " OU=Engineering,DC=TreyResearch,DC=net" `
              -PassThru
```

DistinguishedName	Name	ObjectClass	ObjectGUID
-----	----	-----	-----
CN=TREY-DESK-22, OU=Engine...	TREY-DESK-22	computer	46df71bd-ba88-4b26-9091-b8db6e07261a

Even though we only moved a single computer and a single user, the same methods can be used to move hundreds or even thousands of users. Of course, making a mistake when moving one user is a nuisance, for both you and the user, but it's fairly easily corrected. Making a mistake by moving thousands of users is still easily corrected but is likely to cause somewhat more annoyance. Therefore, always check your work before committing to large changes that will affect many users.

## Summary

---

In this chapter, you learned how to create and manage users, groups, and OUs. You learned how to filter against the properties of users, groups, and computers to selectively act on the results of that filter. You also learned how to add users to groups and move users and computers into an OU.

In the next chapter, you'll learn how to deploy additional domain controllers into your existing domain and how to manage the FSMO roles in your domain.



# Index

## A

- A resource records
  - creating 28
  - described 26
- AAAA resource records
  - creating 28
  - described 26
- account policies 135–142
- AccountExpirationDate parameter 131
- AccountNotDelegated parameter 131
- AccountPassword parameter 131
- Active Directory
  - See also* Active Directory Domain Services (AD DS)
  - authoritative restores 157–162
  - Azure AD 185
  - backup methods 144
  - cloud, types in 185
  - non-authoritative restores 155–157
  - offline media backup, creating 152
  - restore methods 155
  - role, installing on servers 67
  - snapshots 153, 154
  - Windows Server Backup 144–151
- Active Directory Administrative Center (ADAC) 137, 138
- Active Directory Domain Services (AD DS)
  - See also* Active Directory; DNS; sites
  - Active Directory Recycle Bin 162–164
  - authoritative restores 157–162
  - Azure AD 185
  - database, installing from media 92–94
  - DHCP, activating in 38
  - functional levels 10
  - supported groups 51–54
  - installing 6
  - members, adding to groups 53
  - non-authoritative restores 155–157
- Active Directory Domain Services (AD DS) (*continued*)
  - replication 178–180
  - restore methods 155
  - role, installing 105, 111
  - schema, updating 79
  - sites, configuring 168–174
- Active Directory Domain Services role 105, 111
- Active Directory Recycle Bin
  - enabling 153, 162
  - restoring objects, using 162–164
- Active Directory Services Interface (ADSI) 122
- Active Directory snapshots
  - configuring 153
  - restoring objects, using 164–166
- AD DS (Active Directory Domain Services) *See* Active Directory Domain Services (AD DS)
- ActiveDirectory module 6
- ADAC (Active Directory Administrative Center) 137, 138
- Add noun 7
- Add-ADDSReadOnlyDomainControllerAccount cmdlet 88, 93
- Add-ADGroupMember cmdlet 52
- Add-ADPrincipalGroupMembership cmdlet 56
- Add-AzureAccount cmdlet 198
- Add-AzureProvisioningConfig cmdlet 215
- Add-Computer cmdlet 68, 96
- Add-DhcpServerInDC cmdlet 38
- Add-DhcpServerv4ExclusionRange cmdlet 39
- Add-DhcpServerv4Scope cmdlet 39
- Add-DhcpServerv6ExclusionRange cmdlet 40
- Add-DhcpServerv6Scope cmdlet 40
- Add-DnsServerConditionalForwarderZone cmdlet 25
- Add-DnsServerPrimaryZone cmdlet 17
- Add-DnsServerResourceRecord cmdlet 27
- Add-DnsServerResourceRecordA cmdlet 27, 28
- Add-DnsServerResourceRecordAAAA cmdlet 27
- Add-DnsServerResourceRecordCName cmdlet 27

## Add-DnsServerResourceRecordDnsKey cmdlet

- Add-DnsServerResourceRecordDnsKey cmdlet 27
- Add-DnsServerResourceRecordDS cmdlet 27
- Add-DnsServerResourceRecordMX cmdlet 27
- Add-DnsServerResourceRecordPtr cmdlet 27
- Add-DnsServerSecondaryZone cmdlet 22
- Add-DnsServerStubZone cmdlet 24
- Add-DnsServerZoneDelegation cmdlet 26
- Add-KDSRootKey cmdlet 129
- AD-Domain-Services role
  - installing on servers 105, 111
  - servers, adding to 67
- AddressFamily parameter 4
- ADDSDeployment module 6, 7
- Add-WBBareMetalRecovery cmdlet 149, 150
- Add-WBFileSpec cmdlet 148
- Add-WBSystemState cmdlet 150
- Add-WBTarget cmdlet 149
- Add-WBVolume cmdlet 147
- Add-WindowsFeature cmdlet 6
- ADFineGrainedPasswordPolicy cmdlets 137
- ADFineGrainedPasswordPolicySubject cmdlets 137
- ADGroupMember cmdlets 51
- adprep tool 86, 87
- ADPrepCredential parameter 69
- ADPrincipalGroupMembership cmdlets 51
- ADSI (Active Directory Services Interface) 122, 123
- ADUserResultantPasswordPolicy cmdlets 137
- alias 33
- AllowDomainControllerReinstall parameter 69
- AllowPasswordReplicationAccountName parameter 69
- AllVolumes parameter 148
- AlternateWINSsServer parameter 76
- ApplicationPartitionsToReplicate parameter 69
- array 53
- AuthenticationPolicy parameter 131
- AuthenticationPolicySilo parameter 131
- authoritative restores 155, 157–162
- AuthType parameter 131, 169
- AutomaticInterSiteTopologyGenerationEnabled parameter 169
- AutomaticTopologyGenerationEnabled parameter 169
- Azure
  - authenticating to accounts 195–198
  - cloud service, creating 212, 213
  - domain controllers, configuring as 218, 219
  - free trial 184
  - self-signed certificates 199–201
  - storage accounts, provisioning 212, 213

- Azure (*continued*)
  - subscription list, getting 199
  - subscriptions, connecting to 195, 210
  - virtual machines, creating 210–218
  - virtual networks, creating 202–207
- Azure AD
  - authenticating to 185
  - authentication method 195–198
  - Azure subscriptions, connecting to 195
  - described 185
- Azure module for Windows PowerShell
  - See also* Windows PowerShell
  - Azure account, authenticating 195–198
  - Azure AD, connecting to account 195
  - downloading and installing 185–188, 195
  - loading 187, 188
  - unique nouns 188

## B

- backtick character xii
- backup domain controllers (BDCs) *See* read-only domain controllers (RODCs)
- backup policies
  - replacing 151
  - schedule, setting for 150
  - Windows Server Backup 147
- backups
  - bare-metal 144
  - critical-volumes 144
  - deploying 151
  - full server 144
  - offline media, creating 152
  - system state 144
  - Windows Server Backup 144–151
- bare-metal backup 144
- bcdedit tool 155
- BizSpark 184
- bridge connections 178

## C

- canonical records 33
- Certificates parameter 131
- certificates, self-signed 199–201

## child domains

*See also* domains

creating 105–108

described 102

parameters 106

preparing the server 102–104

promotions, testing 105, 106

CloneComputerName parameter 76

## cloning domain controllers

DCCloneConfig.xml, creating 76, 77

environment requirements 72, 73

MSAs, removing 75

problem applications 74, 75

source domain controller, preparing 73

## cloud

Active Directory types 185

services, Azure 212, 213

## cmdlets

Add-ADDsReadOnlyDomainControllerAccount  
88, 93

Add-ADGroupMember 52

Add-ADPrincipalGroupMembership 56

Add-AzureAccount 198

Add-AzureProvisioningConfig 215

Add-Computer 68, 96

Add-DhcpServerInDC 38

Add-DhcpServerv4ExclusionRange 39

Add-DhcpServerv4Scope 39

Add-DhcpServerv6ExclusionRange 40

Add-DhcpServerv6Scope 40

Add-DnsServerConditionalForwarderZone 25

Add-DnsServerPrimaryZone 17

Add-DnsServerResourceRecord 27

Add-DnsServerResourceRecordA 27, 28

Add-DnsServerResourceRecordAAAA 27

Add-DnsServerResourceRecordCName 27

Add-DnsServerResourceRecordDnsKey 27

Add-DnsServerResourceRecordDS 27

Add-DnsServerResourceRecordMX 27

Add-DnsServerResourceRecordPtr 27

Add-DnsServerSecondaryZone 22

Add-DnsServerStubZone 24

Add-DnsServerZoneDelegation 26

Add-KDSRootKey 129

Add-WBBareMetalRecovery 149, 150

Add-WBFileSpec 148

Add-WBSystemState 150

Add-WBTarget 149

cmdlets (*continued*)

Add-WBVolume 147

Add-WindowsFeature 6

ConvertTo-SecureString 91

Enable-ADOptionalFeature 153

Enable-PSRemoting 172

Export-DnsServerZone 21

Export-PfxCertificate 201

Format-Table 7, 103, 109, 155, 170, 171, 211

Get-ADDCloningExcludedApplicationList 74

Get-ADDefaultDomainPasswordPolicy 136

Get-ADGroupMember 53

Get-ADObject 11

Get-ADRootDSE 11

Get-ADTrust 117

Get-AzureAccount 199

Get-AzureLocation 211

Get-AzureRemoteDesktopFile 218

Get-AzureStorageAccount 213

Get-AzureSubscription 199, 210

Get-AzureVM 214

Get-AzureVMImage 214, 215

Get-AzureVNetConfig 209

Get-ChildItem 200, 201

Get-Command 7, 59, 145, 187, 188

Get-Credential 68

Get-Date 129

Get-DnsServerResourceRecord 36

Get-DnsServerScavenging 35

Get-DnsServerZone 20

Get-Help 32

Get-Member 4, 103

Get-myADVersion 11, 12

Get-NetAdapter 3

Get-NetFirewallRule 172

Get-NetIPAddress 5, 109

Get-WBBackup 155, 156

Get-WBBackupSet 155

Get-WBDisk 148

Get-WBPolicy 151

Get-WBVolume 147, 148

Get-WindowsFeature 67, 126, 127

Get-WMIObject 128

Import-CSV 48

Import-Module 8

Import-PfxCertificate 201

Install-ADDSDomainController 68, 105, 152

Install-ADDSEForest 9, 13



cmdlets (*continued*)

- Install-ADServiceAccount 132
- Install-WindowsFeature 6, 67, 105
- Mount-ADDatabase 165
- New-ADDCCloneConfigFile 76
- New-ADOrganizationalUnit 57, 139
- New-ADReplicationSite 168, 169
- New-ADReplicationSiteLink 177
- New-ADReplicationSubnet 169
- New-ADServiceAccount 130, 131
- New-ADUser 43, 44
- New-AzureQuickVM 213
- New-AzureService 212
- New-AzureStorageAccount 213
- New-AzureVM 216
- New-AzureVMConfig 215
- New-NetIPAddress 4, 64
- New-Object 120
- New-ScheduledTaskAction 134
- New-SelfSignedCertificate 199
- New-VM 176
- New-WBBackupTarget 149
- New-WBFileSpec 148
- New-WBPolicy 147
- ntdsutil 152
- printing syntax of 32
- Read-Host 44, 45
- Register-ScheduledTask 135
- Remove-ADComputerServiceAccount 129
- Remove-ADReplicationSubnet 174
- Remove-ADServiceAccount 129
- Remove-AzureVNConfig 209
- Remove-DnsServerZone 25
- Rename-Computer 6, 68
- Restart-Computer 75
- Save-Help 8
- Select-AzureSubscription 210
- Set-ADObject 139
- Set-AzureStaticVnetIP 216
- Set-AzureSubnet 216
- Set-AzureSubscription 213
- Set-DhcpServerv4OptionValue 40
- Set-DhcpServerv6OptionValue 41
- Set-DnsClientServerAddress 5
- Set-DnsServerConditionalForwarderZone 25
- Set-DnsServerPrimaryZone 19
- Set-DnsServerResourceRecord 36

cmdlets (*continued*)

- Set-DnsServerScavenging 35
- Set-DnsServerSecondaryZone 23
- Set-DnsServerStubZone 24
- Set-DnsServerZoneDelegation 26
- Set-LocalUserPassword 125
- Set-NetFirewallRule 172
- Set-NetIPInterface 4, 64
- Set-VM 176
- Set-WBPolicy 151
- Set-WBSchedule 150
- Start-DnsServerScavenging 35
- Start-VM 176
- Start-WBSystemStateRecovery 155
- Test-ADDSDomainControllerInstallation 105
- Test-ADDSEForestInstallation 8
- Test-ADDSEReadOnlyDomainControllerAccount 88
- Test-ADForestInstallation 114
- Test-Path 49
- Test-TailspinForest 114
- Uninstall-ADServerAccount 129
- Update-Help 7
- viewing in module 7
- WBBareMetalRecovery 150
- WindowsServerBackup module 145, 146
- Write-Host 50
- CNAME resource records
  - creating 33
  - described 27
- companion content download xii
- CompoundIdentitySupported parameter 131
- ComputerName parameter 6
- computers
  - organizational unit (OU), adding to 58–61
  - SamAccountName 130
- conditional forwards 25
- Confirm:\$False parameter 129
- ConvertTo-SecureString cmdlet 91
- Cost parameter 177
- CreateDnsDelegation parameter 13, 69
- CreatePtr parameter 28
- Credential parameter 6, 69, 131, 169, 177
- CriticalReplicationOnly parameter 69
- critical-volumes backup 144
- CSV files, adding users with 48–51

## D

- DatabasePath parameter 13, 69
- dcpromo command 7
- deploying
  - DHCP 37, 38
  - forests 116, 117
- deployments
  - See also* read-only domain controllers (RODCs)
  - media 92–94
  - non-staged 94–99
  - staged 87–93
  - target servers 89–94, 97–99
  - testing 71, 72
  - warnings 72
- Description parameter 131, 169, 177
- DHCP (Dynamic Host Configuration Protocol)
  - See also* IP addresses
  - creating local groups 38
  - deploying 37, 38
  - disabling 4
- Directory Services Recovery Mode (DSRM) 155
- Directory Services Restore Mode (DSRM) password 10
- disaster recovery, seizing roles 82, 83
- Dismount-ADDatabase function 154
- DisplayName parameter 131
- Distinguished Name (DN) 53, 60, 61
- DNS
  - See also* Active Directory Domain Services (AD DS)
  - canonical records 33
  - conditional forwards 25
  - delegation warning 9
  - forward lookup zones 17
  - name resolution 17
  - network adapters, settings 104, 110, 111
  - pointer records 27
  - primary zones 17–23
  - resource records 26–35
  - reverse lookup zones 17, 18, 22
  - secondary zones 22–24
  - server address, setting 5
  - Service records 27
  - Start of Authority records 27
  - stub zones 16, 17, 24
  - text records 27
  - zone aging 35, 36
  - DNS (*continued*)
    - zone delegation 26
    - zone file locations 17
    - zone scavenging 35, 36
    - zone transfers 20, 23, 24
    - zonename 17
    - zones, described 16
- DnsDelegationCredential parameter 13, 69
- DNSHostName parameter 131
- DNSKey resource records 27
- domain controller roles 79
- domain controllers
  - See also* forests; read-only domain controllers (RODCs)
  - for child domains 102–108
  - cloning 72–78
  - configuring 218, 219
  - converting to servers 64
  - creating 7
  - FSMO roles 79–83
  - initial, deploying 9–13
  - IP address configuration 64
  - fixed IP address, setting 4
  - naming 6
  - promotions, testing 114–116
  - replication 178–181
  - roles, seizing 82, 83
  - security groups 73, 74
  - servers, promoting 102–105
  - site links, creating 176–178
  - sites, assigning to 170
- Domain Mode, discovering 11
- Domain Name System (DNS) *See* DNS
- domain naming master role 79
- domain service accounts, creating 125
- DomainMode
  - parameter 13
  - values 10
- DomainName parameter 13, 69
- DomainNetbiosName parameter 13
- domains
  - See also* child domains; tree domains
  - adding 79
  - administrative user, creating 44
  - allocating relative identifiers (RIDs) 79
  - child domains 102–108

## dot-sourcing functions

- domains (*continued*)
  - creating 7, 105
  - FSMO roles 79–83
  - functional level 10
  - KDS root key, creating 129
  - password policies 136, 137
  - prepping for domain controllers 86
  - prepping for RODC 86
  - promotions, testing 105, 106, 112, 113
  - removing 79
  - security groups, adding users 45, 46
  - servers, joining 68
  - sites, removing 174
  - tree domains 102, 108–113
  - users, adding in batches 48–51
  - users, creating 43–51
- dot-sourcing functions 153
- DS (Delegated Signer) resource records 27
- DSRM (Directory Services Recovery Mode) 155
- Dynamic Host Configuration Protocol (DHCP) *See* DHCP (Dynamic Host Configuration Protocol)

## E

- ebooks, free downloads xiv
- EffectiveImmediately parameter 129
- Enable-ADOptionalFeature cmdlet 153
- Enabled parameter 131
- Enable-PSRemoting cmdlet 172
- errata xiv
- Export-DnsServerZone cmdlet 21
- Export-PfxCertificate cmdlet 201

## F

- features, installing, adding, and preloading 6
- fixed IP address, setting 4, 5
- flexible single master operations (FSMO) roles
  - seizing 83
  - transferring 80–82
  - types 79
- Force parameter 6, 13, 129, 156

- Forest Mode, discovering 11
- forest objects 120
- ForestMode
  - parameter 13
  - values 10
- forests
  - See also* domain controllers; tree domains
  - Active Directory Recycle Bin, enabling for 162
  - creating 7, 114–117
  - deploying 116, 117
  - functional level default 10
  - initial, deploying 9–13
  - key parameters 13
  - networking, configuration 114
  - objects 120
  - prepping for domain controllers 86
  - promotions, testing 114–116
  - shortcut trusts 118
  - testing 8
  - trusts, creating 117–120
- Format-Table cmdlet 7, 103, 109, 155, 170, 171, 211
- forward lookup
  - records 26
  - zones 17
- FSMO (flexible single master operations) roles
  - seizing 82, 83
  - transferring 80–82
  - types 79
- full server backup 144
- functions 153, 154

## G

- Get-ADDCloningExcludedApplicationList cmdlet 74
- Get-ADDefaultDomainPasswordPolicy cmdlet 136
- Get-ADGroupMember cmdlet 53
- Get-ADObject cmdlet 11
- Get-ADRootDSE cmdlet 11
- Get-ADTrust cmdlet 117
- Get-AzureAccount cmdlet 199
- Get-AzureLocation cmdlet 211
- Get-AzureRemoteDesktopFile cmdlet 218
- Get-AzureStorageAccount cmdlet 213
- Get-AzureSubscription cmdlet 199, 210

Get-AzureVM cmdlet 214  
 Get-AzureVMImage cmdlet 214, 215  
 Get-AzureVNetConfig cmdlet 209  
 Get-ChildItem cmdlet 200, 201  
 Get-Command cmdlet 7, 59, 145, 187, 188  
 Get-Credential cmdlet 68  
 Get-Date cmdlet 129  
 Get-DnsServerResourceRecord cmdlet 36  
 Get-DnsServerScavenging cmdlet 35  
 Get-DnsServerZone cmdlet 20  
 Get-Help cmdlet 32  
 Get-Member cmdlet 4, 103  
 Get-myADVersion cmdlet 11, 12  
 Get-myADVersion.ps1 script 11, 12  
 Get-NetAdapter cmdlet 3  
 Get-NetFirewallRule cmdlet 172  
 Get-NetIPAddress cmdlet 5, 109  
 Get-WBBackup cmdlet 155, 156  
 Get-WBBackupSet cmdlet 156  
 Get-WBBackupSet cmdlet 155, 156  
 Get-WBDisk cmdlet 148  
 Get-WBPolicy cmdlet 151  
 Get-WBVolume cmdlet 147, 148  
 Get-WindowsFeature cmdlet 67, 126, 127  
 Get-WMIObject cmdlet 128  
 gMSAs (group managed service accounts)  
   creating 130–132  
   described 129  
   domains, preparing for 129  
   installing 132, 133  
   KDS root key 129  
   passwords 132  
   scheduled tasks, using for 134, 135  
   security groups, creating 130  
   services, using for 133, 134  
   virtual accounts, configuring 135  
 group managed service accounts (gMSAs) *See* gMSAs  
   (group managed service accounts)  
 groups  
   creating 52  
   managing 54–56  
   scope levels 51  
   supported types 51  
   users, adding 52, 53  
   users, adding to multiple 56

## H

help files, Windows PowerShell  
   installing from network share 8  
   updating 7, 8, 68, 105, 111  
 HomePage parameter 131

## I

IFM (install from media)  
   creating media 92  
   process 87  
   requirements 152  
 Import-CSV cmdlet 48  
 Import-Module cmdlet 8  
 Import-PfxCertificate cmdlet 201  
 IncludeAllSubfeature parameter 6  
 infrastructure master role 79  
 install from media (IFM)  
   creating media 92  
   deployment option 87  
   process 92  
   requirements 152  
 Install noun 7  
 Install-ADDSDomainController cmdlet 68, 105, 152  
 Install-ADDSDomainController cmdlet parameters  
   68–70  
 Install-ADDSDomainController cmdlet 9  
 Install-ADDSDomainController cmdlet parameters 13  
 Install-ADServiceAccount cmdlet 132  
 InstallationMediaPath parameter 70, 94, 152  
 InstallDns parameter 13, 70  
 Install-WindowsFeature cmdlet 6, 67, 105  
 Instance parameter 131, 169, 177  
 interface alias 3  
 interface index 3  
 InterfaceAlias property 4  
 InterfaceIndex property 4  
 intersite replication 167, 178  
 InterSiteTopologyGenerator parameter 169  
 InterSiteTransportProtocol parameter 177  
 intrasite replication 167, 178  
 IP addresses  
   *See also* DHCP (Dynamic Host Configuration  
   Protocol); IPv6  
   fixed, assigning 103, 104, 109, 110, 114

## IPv6, configuring

- IP addresses (*continued*)
  - fixed, setting 4
  - IPv4, configuring 38, 39
- IPv6, configuring 65
  - server, configuring 3–6
- IPv4Address parameter 76
- IPv4DefaultGateway parameter 76
- IPv4DNSResolver parameter 76
- IPv4SubnetMask parameter 76
- IPv6
  - configuring scope 40, 41
  - host address records 26
- IPv6DNSResolver parameter 76

## K

- KCC (Knowledge Consistency Checker) 176
- KDS (Key Distribution Services) 129
- KerberosEncryptionType parameter 131
- Key Distribution Services (KDS) 129
- Knowledge Consistency Checker (KCC) 176

## L

- LoadExisting parameter 17
- LogPath parameter 13, 70

## M

- managed service accounts (MSAs) *See* MSAs (managed service accounts)
- ManagedBy parameter 169
- ManagedPasswordIntervalInDays parameter 131
- media deployment 92–94, 98, 99
  - See also* IFM (install from media)
- Microsoft Azure *See* Azure
- Microsoft Press Book Support xiv
- modules directory 124
- Mount-ADDatabase cmdlet 165
- Mount-ADDatabase function 153
- MoveInfrastructureOperationMasterRoleIfNecessary parameter 70

- MSAs (managed service accounts)
  - associating with services 128
  - described 122
  - installation requirements 126–128
  - passwords 126–128
  - removing 75, 128, 129
- MX (Mail Exchange) resource records
  - creating 34
  - described 27

## N

- Name parameter 169, 177
- name resolution 17
- name resource records 28
- naming domain controllers 6
- network adapters
  - assigning IP addresses 3, 103, 104
  - capturing into variables 65
  - configuring 64–67
  - disabling DHCP 4
  - DNS settings, configuring 104, 110, 111
  - interface alias 3
- network configuration, exporting 209
- network deployment
  - non-staged 97, 98
  - passwords 91
  - staged 91
- network settings, VPN 202
- networking
  - configuring 64–67
  - configuring for forests 114–117
- New-ADDCCloneConfigFile cmdlet 76
- New-ADDCCloneConfigFile parameter 76
- New-ADOrganizationalUnit cmdlet 57, 139
- New-ADReplicationSite cmdlet 168, 169
- New-ADReplicationSiteLink cmdlet 177
- New-ADReplicationSubnet cmdlet 169
- New-ADServiceAccount cmdlet 130, 131
- New-ADSnapshot function 153
- New-ADUser cmdlet 43, 44, 46–48
- New-AzureQuickVM cmdlet 213
- New-AzureService cmdlet 212
- New-AzureStorageAccount cmdlet 213

New-AzureVM cmdlet 216  
 New-AzureVMConfig cmdlet 215  
 New-NetIPAddress cmdlet 4, 64  
 New-Object cmdlet 120  
 New-ScheduledTaskAction cmdlet 134  
 New-SelfSignedCertificate cmdlet 199  
 New-VM cmdlet 176  
 New-WBBackupTarget cmdlet 149  
 New-WBFileSpec cmdlet 148  
 New-WBPolicy cmdlet 147  
 NoDnsOnNetwork parameter 13, 70  
 NoGlobalCatalog parameter 70  
 non-authoritative restore 155–157  
 None parameter 76  
 non-staged deployments 94–99  
 NoRebootOnCompletion parameter 13, 70  
 Notify parameter 20  
 NotifyServers parameter 20  
 nouns 7, 188  
 NS (Name Server) resource records 27  
 ntdsutil cmdlet 152

## O

object references, updating and replicating 79  
 objects
 

- corruption recovery 153
- restore using Active Directory Recycle Bin 162–164
- restore using snapshots 164–166
- Windows Server Backup 147–151

 offline media, creating 152  
 Offline parameter 76  
 operations master *See* FMSO (flexible single master operation) roles  
 organizational unit (OU)
 

- computers, adding 58–61
- creating 56–58, 138
- users, adding 58–61

 OtherAttributes parameter 43, 57, 131, 169, 177

## P

### parameters

AccountExpirationDate 131  
 AccountNotDelegated 131  
 AccountPassword 131  
 AddressFamily 4  
 ADPrepCredential 69  
 AllowDomainControllerReinstall 69  
 AllowPasswordReplicationAccountName 69  
 AllVolumes 148  
 AlternateWINSSEServer 76  
 ApplicationPartitionsToReplicate 69  
 AuthenticationPolicy 131  
 AuthenticationPolicySilo 131  
 AuthType 131, 169  
 AutomaticInterSiteTopologyGenerationEnabled 169  
 AutomaticTopologyGenerationEnabled 169  
 Certificates 131  
 child domains 106  
 CloneComputerName 76  
 CompoundIdentitySupported 131  
 ComputerName 6  
 Confirm:\$False 129  
 Cost 177  
 CreateDnsDelegation 13, 69  
 CreatePtr 28  
 Credential 6, 69, 131, 169, 177  
 CriticalReplicationOnly 69  
 DatabasePath 13, 69  
 Description 131, 169, 177  
 DisplayName 131  
 DnsDelegationCredential 13, 69  
 DNSHostName 131  
 DomainMode 13  
 DomainName 13, 69  
 DomainNetbiosName 13  
 EffectiveImmediately 129  
 Enabled 131  
 Force 6, 13, 129, 156  
 ForestMode 13  
 HomePage 131  
 IncludeAllSubfeature 6  
 Install-ADDSDomainController 68  
 InstallationMediaPath 70, 94, 152

## parameters

### parameters (*continued*)

- InstallDns 13, 70
- Instance 131, 169, 177
- InterSiteTopologyGenerator 169
- InterSiteTransportProtocol 177
- IPv4Address 76
- IPv4DefaultGateway 76
- IPv4DNSResolver 76
- IPv4SubnetMask 76
- IPv6DNSResolver 76
- KerberosEncryptionType 131
- LoadExisting 17
- LogPath 13, 70
- ManagedBy 169
- ManagedPasswordIntervalInDays 131
- MoveInfrastructureOperationMasterRoleIf-Necessary 70
- Name 169, 177
- New-ADReplicationSite 169
- New-ADReplicationSiteLink 177
- New-ADUser 46–48
- NoDnsOnNetwork 13, 70
- NoGlobalCatalog 70
- None 76
- NoRebootOnCompletion 13, 70
- Notify 20
- NotifyServers 20
- Offline 76
- OtherAttributes 43, 57, 131, 169, 177
- PassThru 4, 28, 131, 169, 177
- Path 76, 132
- PreferredWINSsServer 76
- PrincipalsAllowedToDelegateToAccount 132
- PrincipalsAllowedToRetrieveManagedPassword 132
- ProtectedFromAccidentalDeletion 140, 169
- ReadOnlyReplica 70
- RedundantServerTopologyEnabled 169
- ReplicationFrequencyInMinutes 177
- ReplicationSchedule 169, 177
- ReplicationSourceDC 70
- RestartComputer 156
- RestrictToOutboundAuthenticationOnly 132
- RestrictToSingleComputer 132
- SafeModeAdministratorPassword 10, 13, 69, 91
- SamAccountName 132
- ScheduleHashingEnabled 169
- SecondaryServers 20

### parameters (*continued*)

- SecureSecondaries 20, 24
- Server 132, 169, 177
- ServicePrincipalNames 132
- site links 177
- SiteName 69, 76, 106
- SitesIncluded 177
- SkipAutoConfigureDns 13, 70
- SkipPreChecks 13, 69
- SourcePath 8, 145
- SRV resource records 34, 35
- Static 76
- SystemKey 70
- SysvolPath 13, 70
- TopologyCleanupEnabled 169
- TopologyDetectStaleEnabled 169
- TopologyMinimumHopsEnabled 169
- tree domains 112
- TrustedForDelegation 132
- UniversalGroupCachingEnabled 169
- UniversalGroupCachingRefreshSite 169
- UseExistingAccount 70
- UserID 135
- Vhd 6
- WindowsServer2000BridgeheadSelectionMethod-Enabled 169
- WindowsServer2000KCCISTGSelectionBehavior-Enabled 169
- WindowsServer2003KCCBehaviorEnabled 169
- WindowsServer2003KCCIgnoreSchedule-Enabled 169
- WindowsServer2003KCCSiteLinkBridging-Enabled 169
- ZoneFile 19
- PassThru parameter 4, 28, 131, 169, 177
- password policies
  - assigning 142
  - changing 136
  - default 136
  - domains 136, 137
  - PSOs (password settings objects) 137–142
- password settings objects (PSOs) 137–142
- passwords
  - Directory Services Restore Mode (DSRM), setting 10
  - gMSAs (group managed service accounts) 132
  - MSAs 126, 128
  - network deployment 91
  - service accounts 122

- Path parameter 76, 132
- PDC emulator role 79
- permissions, zone transfers 23
- pointer records 27
- policies 147
- PreferredWINSserver parameter 76
- primary zones
  - creating 17–19
  - exporting 21–23
  - settings 19–21
- PrincipalsAllowedToDelegateToAccount parameter 132
- PrincipalsAllowedToRetrieveManagedPassword parameter 132
- promoting servers to domain controllers 64
- ProtectedFromAccidentalDeletion parameter 140, 169
- PSOs (password settings objects) 137–142
- PTR resource records 27, 28

## Q

- quotation marks, in Windows PowerShell 10

## R

- RDP (Remote Desktop Protocol) 171
- Read-Host cmdlet 44, 45
- read-only domain controllers (RODCs)
  - See also* deployments; domain controllers
  - account, prepping 87, 88
  - cloning 175
  - described 85
  - media deployment 92–94, 98, 99
  - network deployment 91, 97, 98
  - non-staged deployments 94–99
  - prepping domains for 86
  - staged deployments 87–93
  - target server, deploying 91–94, 97–99
- ReadOnlyReplica parameter 70
- record options 36
- RedundantServerTopologyEnabled parameter 169
- Register-ScheduledTask cmdlet 135
- relative identifiers (RIDs) 79
- Remote Desktop Protocol (RDP) 171, 217
- Remove-ADComputerServiceAccount cmdlet 129
- Remove-ADReplicationSubnet cmdlet 174
- Remove-ADServiceAccount cmdlet 129
- Remove-ADSnapshot function 154
- Remove-AzureVnConfig cmdlet 209
- Remove-DnsServerZone cmdlet 25
- Rename-Computer cmdlet 6, 68
- renaming servers 68
- Repair-ADAttribute function 154
- Repair-ADUserGroup function 154
- replicating object references 79
- replication
  - domain controllers 178–180
  - intersite 167, 178
  - intrasite 167, 178
  - KCC (Knowledge Consistency Checker) 176
  - managing 178–180
  - scheduling 179–181
  - servers, changing 181, 182
  - subnets 169–172, 174
- ReplicationFrequencyInMinutes parameter 177
- ReplicationSchedule parameter 169, 177
- ReplicationSourceDC parameter 70
- resource records
  - alias 33
  - cmdlets for adding 27
  - creating 28–35
  - options, configuring 36
  - types 26, 27
- Restart-Computer cmdlet 75
- RestartComputer parameter 156
- restoring AD DS
  - Active Directory Recycle Bin 162–164
  - authoritative restores 157–162
  - non-authoritative restores 155–157
  - snapshots 164–166
  - types of restores 155
- RestrictToOutboundAuthenticationOnly parameter 132
- RestrictToSingleComputer parameter 132
- reverse lookup
  - record, PTR 27
  - zones 17, 18, 22
- RID master role 79
- RIDs (relative identifiers) 79
- rodcprep command 86
- RODCs (read-only domain controllers)
  - See also* deployments; domain controllers
  - account, prepping 87, 88
  - described 85



RODCs (read-only domain controllers) *(continued)*

- media deployment 92–94, 98, 99
- network deployment 91, 97, 98
- non-staged deployments 94–99
- prepping domains for 86
- staged deployments 87–93
- target server, deploying 91–94, 97–99

## roles

- Active Directory Domain Services, installing 105, 111
- Active Directory, installing 67
- AD DS, installing 105, 111
- domain naming master 79
- FSMO 79–83
- infrastructure master 79
- installing 6
- offline VHD file, adding to 6
- PDC emulator 79
- RID master 79
- schema master 79

## RSAT tools, installing 38

**S**

SafeModeAdministratorPassword parameter 10, 13, 69, 91

SAM account 53, 54

SamAccountName 130, 132

Save-Help cmdlet 8

scheduled tasks 134, 135

ScheduleHashingEnabled parameter 169

schema master role 79

Schema Version, discovering 11

## scripts

- Get-myADVersion.ps1 11, 12
- TechNet Script Center 124
- Test-myForestCreate.ps1 8

secondary zones 22, 23

SecondaryServers parameter 20

SecureSecondaries parameter 20, 24

## security groups

- creating 140–142
- domain controllers, adding 73, 74
- domain controllers, removing 74
- creating for gMSAs computers 130

security groups *(continued)*

- SamAccountName 130
- users, adding 45, 46

security identifier (SID) 53, 79

Select-AzureSubscription cmdlet 210

## self-signed certificates

- client 200, 201
- root 199, 200

Sender Policy Framework (SPF) records 27

server IP addresses, configuring 3–6

Server parameter 132, 169, 177

## servers

*See also* service accounts

Active Directory role, installing 67, 105, 111

AD-Domain-Services role, adding 67

AD-Domain-Services role, installing 105, 111

joining to domains 68

names, changing 96

preparing to promote 102–105, 108–111

promoting to domain controllers 7, 64, 68–72, 102–105

renaming 6, 68, 102, 103, 108, 109, 114

replication, changing 181, 182

## service accounts

*See also* servers

described 122

domain service accounts 125

gMSAs (group managed service accounts) 129–135

local, creating 122–125

MSAs (managed service accounts) 126–129

passwords 122, 128, 132

service authentication *See* service accounts

Service records 27

ServicePrincipalNames parameter 132

services, using gMSA 133, 134

Set-ADObject cmdlet 139

Set-AzureStaticVnetIP cmdlet 216

Set-AzureSubnet cmdlet 216

Set-AzureSubscription cmdlet 213

Set-DhcpServerv4OptionValue cmdlet 40

Set-DhcpServerv6OptionValue cmdlet 41

Set-DnsClientServerAddress cmdlet 5

Set-DnsServerConditionalForwarderZone cmdlet 25

Set-DnsServerPrimaryZone cmdlet 19

Set-DnsServerResourceRecord cmdlet 36

Set-DnsServerScavenging cmdlet 35

Set-DnsServerSecondaryZone cmdlet 23  
 Set-DnsServerStubZone cmdlet 24  
 Set-DnsServerZoneDelegation cmdlet 26  
 Set-LocalUserPassword cmdlet 125  
 Set-NetFirewallRule cmdlet 172  
 Set-NetIPInterface cmdlet 4, 64  
 Set-VM cmdlet 176  
 Set-WBPolicy cmdlet 151  
 Set-WBSchedule cmdlet 150  
 shortcut trusts 117–119  
 SID (security identifier) 79  
 site links 176–178  
 SiteName parameter 69, 76, 106  
 sites  
     *See also* Active Directory Domain Services (AD DS)  
     bridge connections 178  
     described 167, 168  
     intersite replication 167, 178  
     intrasite replication 167, 178  
     links, creating 176–178  
     new, creating 168  
     removing 174  
     renaming 173, 174  
     replication, managing 178–182  
     subnets 169–172, 174  
     UGMC, enabling 175  
 SitesIncluded parameter 177  
 SkipAutoConfigureDns parameter 13, 70  
 SkipPreChecks parameter 13, 69  
 snapshots, creating for AD DS 153, 154  
 SOA (Start of Authority) resource records 27  
 SourcePath parameter 8, 145  
 SRV (Service) resource records  
     described 27  
     parameter types 34, 35  
 staged deployments 87–94  
 Start of Authority records 27  
 Start-DnsServerScavenging cmdlet 35  
 Start-VM cmdlet 176  
 Start-WBSystemStateRecovery cmdlet 155  
 Static parameter 76  
 storage accounts 213  
 stub zones  
     contents 16  
     deploying and managing 24  
     DNS records 17  
 subdomains *See* child domains

subnets  
     creating 169–172  
     Gateway 201  
     removing 174  
 subscriptions  
     connecting to 210  
     current 199  
     storage accounts for 213  
 system state backup 144  
 SystemKey parameter 70  
 SysvolPath parameter 13, 70  
 SYSVOL, recovering 144

## T

tasks, scheduling with gMSAs 134, 135  
 TechNet Script Center 124  
 Test-ADDSDomainControllerInstallation cmdlet 105  
 Test-ADDSForestInstallation cmdlet 8  
 Test-ADDSReadOnlyDomainControllerAccount  
     cmdlet 88  
 Test-ADForestInstallation cmdlet 114  
 testing  
     deployments 71, 72  
     forest environment 7, 8  
 Test-myForestCreate.ps1 script 8  
 Test-Path cmdlet 49  
 Test-TailspinForest cmdlet 114  
 text records 27  
 Time To Live (TTL) 36  
 tools  
     adprep 86, 87  
     bcdedit 155  
     RSAT 38  
 TopologyCleanupEnabled parameter 169  
 TopologyDetectStaleEnabled parameter 169  
 TopologyMinimumHopsEnabled parameter 169  
 tree domains  
     *See also* domains; forests  
     creating 112, 113  
     described 102  
     parameters 112  
     preparing server 108–111  
     promotions, testing 112, 113  
 TrustedForDelegation parameter 132

## trusts

### trusts

- forests, creating 117–120
- shortcut 117
- TTL (Time To Live) 36
- TXT resource records 27

## U

- UGMC (Universal Group Membership Caching) 175
- Uninstall-ADServerAccount cmdlet 129
- Universal Group Membership Caching (UGMC) 175
- UniversalGroupCachingEnabled parameter 169
- UniversalGroupCachingRefreshSite parameter 169
- Update-Help cmdlet 7
- updating object references 79
- UseExistingAccount parameter 70
- user account descriptions 124
- user policies *See* user account descriptions
- UserID parameter 135
- users

- administrative 44
- array 53
- batches, adding 48–51
- confirming identity 54
- groups, adding to 52, 53
- local, creating 123
- modules 124
- multiple groups, adding to 56
- organizational unit (OU), adding to 58–61
- single, creating 43–48

## V

- VHD files 6
- Vhd parameter 6
- virtual accounts, configuring with gMSAs 135
- virtual machines
  - base images 214
  - cloned, creating 77–79
  - configuration objects, creating 214–216
  - creating 210, 213–218
  - deploying 216–218
  - networking configurations 216
  - setting locations 211
  - sizes 215

### virtual networks

- addresses 201
- Azure, creating 202–207
- configuration, saving 209
- virtual private network *See* VPN
- virtualization platforms 73
- VMs *See* virtual machines
- volumes 148
- Volume Shadow Copy Service (VSS) 153
- VPN
  - address space 201
  - client, downloading 207–209
  - network settings 202
  - point-to-site, creating 201–207
  - site-to-site 199
- VSS (Volume Shadow Copy Service) 153

## W

- warnings
  - compatibility 9
  - delegation 9
  - deployments 72
- WBBBackupTarget object 148
- WBBareMetalRecovery cmdlet 150
- WBBareMetalRecovery object 149, 150
- WBFileSpec object 148
- WBPolicy object 147
- WBSchedule 150
- WBSystemState object 150
- WBVolume object 147
- Windows Management Instrumentation (WMI) 171
- Windows PowerShell
  - See also* Azure module for Windows PowerShell
  - ActiveDirectory module 6
  - ADDSDeployment module 6, 7
  - ADSI adapter 38
  - Azure moduleDHCP groups, creating 38
  - help files 7, 8, 68, 105, 111
  - quotation marks 10
  - Windows Server Backup, and 147
- Windows Remote Management (WinRM) 171
- Windows Server, adprep 86
- Windows Server Backup
  - authoritative restore 157–162
  - backup types 144

- Windows Server Backup (*continued*)
  - configuring 147–151
  - described 144
  - installing 145, 146
  - non-authoritative restores 155–157
  - objects 147–151
  - policies 147
- Windows Time Service 79
- WindowsServer2000BridgeheadSelectionMethod-Enabled parameter 169
- WindowsServer2000KCCISTGSelectionBehaviorEnabled parameter 169
- WindowsServer2003KCCBehaviorEnabled parameter 169
- WindowsServer2003KCCIgnoreScheduleEnabled parameter 169
- WindowsServer2003KCCSiteLinkBridgingEnabled parameter 169
- WindowsServerBackup module, cmdlets 145, 146
- WinRM (Windows Remote Management) 171
- WMI (Windows Management Instrumentation) 171
- Write-Host cmdlet 50

## Z

- zone files 17
- zone transfer permissions 23
- ZoneFile parameter 19
- zones
  - aging 35, 36
  - converting 17
  - creating 17
  - delegation 26
  - files 17
  - scavenging 35, 36
  - transfers 20
  - transfer settings 23, 24



## About the author

---

Charlie Russel has been a Microsoft MVP for Windows XP, Security, Windows Server, and now Windows PowerShell. He's a chemist by education, an electrician by trade, a UNIX and Windows sysadmin and Oracle DBA because he raised his hand when he should have known better, and an IT Manager for a major international software and security company by choice.

Charlie is the author of more than three dozen books about operating systems and enterprise environments and is the original author of the Get-Help pages for the Windows PowerShell Azure module. His recent books include *Exam Ref 70-411: Administering Windows Server 2012 R2* (Microsoft Press, 2014), *Working with Windows Small Business Server 2011 Essentials* (Microsoft Press, 2011), *Windows Small Business Server 2011 Administrator's Companion* (Microsoft Press, 2011), *Windows Server 2008 Administrator's Companion* (Microsoft Press, 2008), and *Oracle DBA Automation Quick Reference* (Prentice Hall PTR, 2004).