Sams Teach Yourself
Windows PowerShell® in 24 Hours

800 East 96th Street, Indianapolis, Indiana, 46240 USA
## Contents at a Glance

**Introduction** .................................................................................................................. 1

### Part I: Introducing Windows PowerShell

**HOUR 1** Getting to Know Windows PowerShell .................................................. 7  
**2** Installing and Configuring Windows PowerShell ........................................ 25  
**3** Mastering the Windows PowerShell Help System ...................................... 47  
**4** Finding and Discovering Windows PowerShell Commands ................. 69

### Part II: Understanding Objects and the Pipeline

**HOUR 5** Thinking in Terms of Objects ................................................................. 91  
**6** Mastering the Windows PowerShell Pipeline ............................................. 109  
**7** Sorting, Filtering, and Measuring Windows PowerShell Output ........... 133

### Part III: Extending the Reach of Windows PowerShell

**HOUR 8** Managing Windows PowerShell Providers ........................................ 153  
**9** Formatting, Exporting, and Converting Windows PowerShell Output .... 175

### Part IV: Managing Computers Remotely with Windows PowerShell

**HOUR 10** Implementing One-to-One Windows PowerShell Remoting .......... 201  
**11** Implementing One-to-Many Windows PowerShell Remoting .............. 221  
**12** Deploying PowerShell Web Access .......................................................... 237

### Part V: Putting Windows PowerShell to Work

**HOUR 13** Multitasking Windows PowerShell ....................................................... 255  
**14** Harnessing Windows PowerShell Workflow .......................................... 275  
**15** Introducing WMI and CIM ................................................................. 293  
**16** Searching and Filtering with Regular Expressions .............................. 313

### Part VI: Enterprise-Class Windows PowerShell

**HOUR 17** Managing Software with Windows PowerShell OneGet ........... 331  
**18** Desired State Configuration Basics ....................................................... 355
Part VII: Scripting with Windows PowerShell

Hour 19  Introduction to Windows PowerShell Scripting .............................. 377
Hour 20  Making PowerShell Code Portable with Modules ............................ 399

Part VIII: Administering Microsoft Enterprise Servers with Windows PowerShell

Hour 21  Managing Active Directory with Windows PowerShell .................. 417
Hour 22  Managing SQL Server with Windows PowerShell ............................ 437
Hour 23  Managing SharePoint Server with Windows PowerShell ................. 453
Hour 24  Managing Microsoft Azure with Windows PowerShell .................. 471
Index ........................................................................................................... 493
# Table of Contents

## Introduction

- Who Should Read This Book ................................................................. 1
- How This Book Is Organized ................................................................. 2
- Conventions Used in This Book ......................................................... 4
- System Requirements ................................................................. 5

## Part I: Introducing Windows PowerShell

**HOUR 1:** Getting to Know Windows PowerShell 7

- Why You Should Learn Windows PowerShell ........................................... 8
- Brief History of Windows PowerShell ......................................................... 10
- Understanding the Windows PowerShell Components ............................... 13
- Investigating the Power and Simplicity of Windows PowerShell .................. 18
- Summary .................................................................................................. 21
- Q&A ......................................................................................................... 22
- Workshop .................................................................................................. 23

**HOUR 2:** Installing and Configuring Windows PowerShell 25

- Determining Your Windows PowerShell Environment .................................. 26
- Installing the Latest Version of Windows PowerShell .................................... 29
- Customizing the Windows PowerShell Console ......................................... 32
- Customizing the Windows PowerShell ISE .................................................. 39
- Summary .................................................................................................. 42
- Q&A ......................................................................................................... 42
- Workshop .................................................................................................. 44

**HOUR 3:** Mastering the Windows PowerShell Help System 47

- Anatomy of a Windows PowerShell Cmdlet ................................................. 47
- Updating the Windows PowerShell Help Library ......................................... 50
- Understanding Windows PowerShell Help Syntax ........................................ 54
HOUR 7: Sorting, Filtering, and Measuring Windows PowerShell Output 133
Sorting Output ...................................................................................................134
Filtering Output .................................................................................................137
Measuring Objects ..............................................................................................144
Summary ...........................................................................................................148
Q&A ...................................................................................................................148
Workshop ...........................................................................................................150

Part III: Extending the Reach of Windows PowerShell

HOUR 8: Managing Windows PowerShell Providers 153
What Are Providers? ..........................................................................................153
Introduction to Default PSDrives ......................................................................155
Using the FileSystem Provider ...........................................................................159
Using the Alias Provider ....................................................................................166
Using the Registry Provider ...............................................................................167
Using Extended Providers ..................................................................................169
Summary ...........................................................................................................171
Q&A ...................................................................................................................171
Workshop ...........................................................................................................172

HOUR 9: Formatting, Exporting, and Converting Windows PowerShell Output 175
How the PowerShell Formatting Subsystem Works ...........................................176
Formatting PowerShell Output .........................................................................180
Exporting PowerShell Output ............................................................................187
Converting PowerShell Output .........................................................................193
Summary ...........................................................................................................197
Q&A ...................................................................................................................198
Workshop ...........................................................................................................198
Part IV: Managing Computers Remotely with Windows PowerShell

HOUR 10: Implementing One-to-One Windows PowerShell Remoting 201
Understanding Classic Windows PowerShell Remote Access .................................. 201
Introducing “True” PowerShell Remoting ............................................................... 203
Enabling Windows PowerShell Remoting .............................................................. 205
Creating a Windows PowerShell Remote Session ............................................... 209
Sending Scripts over the Network ........................................................................ 214
Summary ............................................................................................................... 217
Q&A ...................................................................................................................... 218
Workshop ............................................................................................................. 218

HOUR 11: Implementing One-to-Many Windows PowerShell Remoting 221
One-to-Many Remote Access in the Classic Scenario ............................................. 221
One-to-Many Remoting with Persistent Sessions .................................................. 224
Managing Session Configurations ........................................................................ 225
One-to-Many Remoting with the Windows PowerShell ISE ................................... 228
Passing Input to Remote Commands ..................................................................... 231
Summary ............................................................................................................... 233
Q&A ...................................................................................................................... 233
Workshop ............................................................................................................. 235

HOUR 12: Deploying Windows PowerShell Web Access 237
Introducing Windows PowerShell Web Access ...................................................... 237
Setting Up the Windows PSWA Gateway .............................................................. 239
Testing the Windows PSWA User Experience ....................................................... 244
Managing the Gateway .......................................................................................... 249
Summary ............................................................................................................... 251
Q&A ...................................................................................................................... 252
Workshop ............................................................................................................. 252
Part V: Putting Windows PowerShell to Work

HOUR 13: Multitasking Windows PowerShell 255
Investigating the PowerShell Job Architecture .................................................. 255
Controlling Job Behavior .................................................................................. 259
Understanding Parent and Child Jobs .............................................................. 261
Introducing the –AsJob Parameter .................................................................... 263
Scheduling Jobs ................................................................................................. 266
Reviewing What We’ve Learned ....................................................................... 270
Summary ........................................................................................................... 272
Q&A ................................................................................................................... 273
Workshop ........................................................................................................... 273

HOUR 14: Harnessing Windows PowerShell Workflow 275
Understanding How Windows PowerShell Workflow Works ........................... 276
Writing Your First Windows PowerShell Workflow .......................................... 279
Running a Workflow as a Job ........................................................................... 282
Understanding Workflow Activities .................................................................. 283
Tying Everything Together ................................................................................ 288
Summary ........................................................................................................... 289
Q&A ................................................................................................................... 290
Workshop ........................................................................................................... 291

HOUR 15: Introducing WMI and CIM 293
Defining WMI and CIM .................................................................................... 293
Getting Comfortable with WMI ........................................................................ 296
Using Windows PowerShell WMI Commands .................................................. 300
Using Windows PowerShell CIM Commands ................................................... 305
Summary ........................................................................................................... 308
Q&A ................................................................................................................... 308
Workshop ........................................................................................................... 310

HOUR 16: Searching and Filtering with Regular Expressions 313
Revisiting the Wildcard Operators .................................................................... 313
Understanding Regular Expressions ................................................................. 315
Using the –Match Parameter .......................................................................... 317
### HOUR 23: Managing SharePoint Server with Windows PowerShell 453

- Understanding the Environment ......................................................... 453
- Deploying a Service Application .......................................................... 457
- Deploying a Web Application ................................................................. 459
- Deploying a Site Collection ................................................................. 460
- Setting Permissions on a Site Collection ................................................ 462
- Reporting on a SharePoint Farm ............................................................ 463
- Summary .............................................................................................. 468
- Q&A ...................................................................................................... 468
- Workshop .............................................................................................. 468

### HOUR 24: Managing Microsoft Azure with Windows PowerShell 471

- Defining Microsoft Azure ........................................................................ 472
- Preparing Your Azure-PowerShell Environment ...................................... 474
- Working with Azure Virtual Machines .................................................... 477
- Managing Office 365 and SharePoint Online with Azure ....................... 485
- Summary .............................................................................................. 488
- Q&A ...................................................................................................... 489
- Workshop .............................................................................................. 490

### INDEX 493
Timothy Warner is an IT professional and technical trainer based in Nashville, Tennessee. Tim became acquainted with information technology in 1982 when his dad bought the family a Timex Sinclair 1000 home computer and he taught himself BASIC programming. Today he works as an author/evangelist for Pluralsight and shares Windows PowerShell knowledge with anyone who’ll listen at his Two Minute PowerShell blog: http://2minutepowershell.com. You can reach Tim directly via LinkedIn: http://linkedin.com/in/timothywarner.
Dedication

To all my students, past and present.
Thank you for giving me a professional calling,
and I hope that my work helps you attain your goals.

Acknowledgments

The Windows PowerShell community is a terrific group of people. Thank you, Jeffrey Snover, Bruce Payette, and Lee Holmes et al. for giving the world Windows PowerShell. Thanks to all the PowerShell experts in the world for being so kind and willing to share your knowledge. I seek to emulate your actions every day.

It may take a village to raise a child, but I know that it takes a large office full of talented professionals to publish a book. To that end, I want to thank my wonderful editor Joan Murray for having faith in my abilities. Thanks to my publisher, Greg Wiegand, for being so receptive to my ideas.

Editorial and production staff rarely receive the credit they deserve. Thanks so much to Windows PowerShell MVP Jeff Wouters, my technical editor, for being so thorough with the manuscript. Truly, this book is at least twice as good as it originally was thanks to you.

Thanks to Keith Cline, my copyeditor, for making my writing easier to follow. Keith knows that I gave his editing skills quite a workout, for sure. Sorry, Keith!

I extend my gratitude as well to Andy Beaster, my production editor, and to the ever-helpful Cindy Teeters for streamlining the entire book publishing process. Andy and Cindy are professionals in every sense of the word.

Thanks to my family, friends, and colleagues for your never-ending love and support. Finally, thank you, my reader. I hope that this book helps you accomplish your next professional goal.
We Want to Hear from You!

As the reader of this book, you are our most important critic and commentator. We value your opinion and want to know what we’re doing right, what we could do better, what areas you’d like to see us publish in, and any other words of wisdom you’re willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn’t like about this book—as well as what we can do to make our books better.

Please note that we cannot help you with technical problems related to the topic of this book.

When you write, please be sure to include this book’s title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: consumer@samspublishing.com
Mail: Sams Publishing
     ATTN: Reader Feedback
     800 East 96th Street
     Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.
“Try not. Do...or do not. There is no try”
—Yoda, Star Wars Episode V: The Empire Strikes Back

Hello, and welcome to the world of Windows PowerShell. I’m your instructor, Tim Warner. To me, it’s a good sign that you’re actually reading this Introduction (so few readers of tech books do, in my experience). Perhaps your first question is, “What’s in it for me?” and I’m here to give you those details with minimal muss and fuss.

If you work as a Windows systems administrator or hope to in the future, learning Windows PowerShell is no longer an option. Likewise, if you plan to advance your career in IT administration, you need to know your way around Windows PowerShell scripting and automation. This, then, is what’s in it for you: By learning how to harness Windows PowerShell, you make yourself a more effective and valuable Windows systems administrator. And if you have value in the IT workplace, you have a means of having a stable, lucrative career and an equally stable and lucrative life.

Who Should Read This Book

The first thing I do when I teach “stand up” training classes is to get a feel for my student. What is your background? What do you hope to get out of this training? As I wrote this book, I had the following audiences in the forefront of my mind:

▶ **Microsoft certification candidates**: I’m here to tell you that if you don’t understand Windows PowerShell, you have a high likelihood of failing your Microsoft Certified Professional (MCP) exams. And I’m not just talking about Windows Server 2012 R2 certification, either. Microsoft Learning stresses PowerShell-based administration in all of their products nowadays, so you simply cannot escape the technology, no matter how hard you might try.

▶ **Windows systems administrators**: I’m sure that you’ve been aware of Windows PowerShell over the past several years, and maybe you’ve been avoiding learning the technology because the tech appeared too “programmy” or math heavy. Let me assure you that by the time you complete this book, you won’t be afraid of that anymore because you’ll be convinced how much easier PowerShell makes your life as a “boots on the ground” sysadmin.
IT newcomers: If you are working on a transition into full-time IT work, whether you’re entering IT from an unrelated field or preparing to graduate from trade school or college, then welcome! You have an advantage in learning Windows PowerShell at the outset of your IT career because you’ll be able to seamlessly integrate PowerShell automation into your vision of IT.

If you find that you don’t belong in any of the previous three classifications, don’t worry about it. Set your sights on learning as much as you can and, above all else, having fun, and you’ll be fine.

How This Book Is Organized

These “24 Hour” books begin with the premise that you can learn a technology (Windows PowerShell, in this case) by studying the material in 24 one-hour sessions. Maybe you can use your lunch break as your training hour; then again, the hour after your children finally fall asleep at night might work better.

In any event, allow me to present hour-by-hour details on how I structured the content:

- Hour 1, “Getting to Know Windows PowerShell,” makes the case that knowing Windows PowerShell is mandatory and not optional for Windows systems administrators. You’ll also learn how PowerShell works from an architectural/design standpoint.


- In Hour 3, “Mastering the Windows PowerShell Help System,” you learn how to learn Windows PowerShell. Believe me, this chapter is one of the three most important chapters of the book because you’ll use the help system every day.

- In Hour 4, “Finding and Discovering Windows PowerShell Commands,” you master the Get-Command cmdlet. This is the second of the three most important chapters, again based on how often you’ll use these skills.

- In Hour 5, “Thinking in Terms of Objects,” you use Get-Member to list the methods and properties of PowerShell objects. This hour completes the “triad” of three core chapters that comprise your foundational understanding of Windows PowerShell.

- In Hour 6, “Mastering the Windows PowerShell Pipeline,” you begin to understand that in Windows PowerShell, you’re always working from within a command pipeline, and you also recognize that in a PowerShell pipeline, you’re almost always dealing with objects.
Page 3

How This Book Is Organized

▶ In Hour 7, “Sorting, Filtering, and Measuring Windows PowerShell Output,” you learn how to cut down your output to separate only the data you need.

▶ In Hour 8, “Managing Windows PowerShell Providers,” you learn how you can access and browse various data stores, from environment variables and the Registry to the certificate store and Active Directory, in the same way that you browse your file system from the command line.

▶ In Hour 9, “Formatting, Exporting, and Converting Windows PowerShell Output,” you pick up some valuable skills on creating submission-quality output of your PowerShell pipelines.

▶ In Hour 10, “Implementing One-to-One Windows PowerShell Remoting,” you get a grip on the wonderful remoting architecture in Windows PowerShell. Here we examine how to set up remoting and establish remote sessions with other Windows computers on our network.

▶ In Hour 11, “Implementing One-to-Many Remoting,” you learn how to send PowerShell commands and even entire scripts to an unlimited amount of target computers in parallel. This chapter demonstrates the raw power you have at your fingertips when you use PowerShell to manage your Windows networks.

▶ In Hour 12, “Deploying PowerShell Web Access,” you learn how to set up PowerShell to be accessed from any remote device—even mobile phones and tablets from outside your corporate firewall. This is a cool technology, for sure.

▶ In Hour 13, “Multitasking Windows PowerShell,” you discover the Windows PowerShell jobs architecture, in which you can send simple or complex PowerShell operations to the background of your session. By mastering jobs, you (and PowerShell) can multitask with aplomb.

▶ In Hour 14, “Harnessing Windows PowerShell Workflow,” you take the next step with PowerShell jobs and learn how to design and deploy durable PowerShell tasks that respond to state changes, such as system reboots. Very cool stuff here.

▶ In Hour 15, “Introducing WMI and CIM,” you finally come to terms with two acronyms many Windows systems administrators hear all the time but rarely understand: Windows Management Instrumentation (WMI) and Common Information Model (CIM). By the end of this chapter, you’ll be crystal-clear on how to fetch system state data from the WMI repository by using PowerShell code.

▶ In Hour 16, “Searching and Filtering with Regular Expressions,” you put your string searches on steroids by learning how to use the .NET Framework’s regular expression syntax to perform highly specific find and replace operations on your data—all with PowerShell code.
In Hour 17, “Installing and Managing Software with OneGet,” you learn how to install and manage software, all from the PowerShell console command line. If you’ve used command-line software package management in Linux or OS X, what you learn during this hour will be immediately familiar.

In Hour 18, “Desired State Configuration Basics,” you learn what will doubtless become the next generation of Windows Server systems configuration: Desired State Configuration, or DSC, in Windows PowerShell.

In Hour 19, “Introduction to Windows PowerShell Scripting,” you take everything you’ve learned over the previous 18 hours of training and apply that knowledge toward code reuse. In other words, you’ll learn the basics of writing, configuration, and running Windows PowerShell script files.

In Hour 20, “Making PowerShell Code Portable with Modules,” you build upon what you learned in the preceding hour of training concerning PowerShell scripts and target that knowledge toward packing your code into modular...well, modules.

In Hour 21, “Managing Active Directory with Windows PowerShell,” you embark on a four-hour journey of PowerShell domain-specific management. Here the “domain” is Active Directory Domain Services (AD DS) itself.

In Hour 22, “Managing SQL Server with Windows PowerShell,” you learn how to use the SQL Server PowerShell module and SQL Management Object (SMO) to interact with SQL Server databases and objects through PowerShell code.

In Hour 23, “Managing SharePoint Server with Windows PowerShell,” you learn how to create SharePoint farm objects (web application, site collection, list, and so forth) by using the SharePoint Server 2013 PowerShell snap-in.

In Hour 24, “Managing Microsoft Azure with Windows PowerShell,” we complete the training by applying Windows PowerShell to Microsoft’s public cloud service: Azure.

Conventions Used in This Book

In my experience as an author and a teacher, I’ve found that many readers and students skip over this part of the book. Congratulations for reading it. Doing so will pay off in big dividends because you’ll understand how and why we formatted this book the way that we did.

Try It Yourself

Throughout the book, you’ll find Try It Yourself exercises, which are opportunities for you to apply what you’re learning right then and there in the book. I do believe in knowledge stacking,
so you can expect that later Try It Yourself exercises assume that you know how to do stuff that you did in previous Try It Yourself exercises.

Therefore, your best bet is to read each chapter in sequence and work through every Try It Yourself exercise.

**About the Bitly Hyperlinks**

Whenever I want to point you to an Internet resource to broaden and deepen the content you’re learning, I provide a uniform resource locator (URL, also called an Internet address) in the following form:


You might wonder what the heck this is. The way I look at the situation, if I were reading this title as a print book and needed to type out a URL given to me by the author, I would rather type in a “shortie” URL than some long, crazy URL with all sorts of special characters, you know what I mean?

*The most important thing I have to tell you concerning the bitly short URLs is that the ending part is case sensitive.* Therefore, typing the previous URL as, say, http://bit.ly/UaKpyD isn’t going to get you to the same page as what I intended.

**About the Code Images**

For most Try It Yourself exercises, you’ll see one or more source code images that are annotated with alphabetic letters. The Try It Yourself steps are then cross-referenced with parts of each code image. Hopefully, you find this format convenient to your learning. Remember not to fall into the trap of blindly copying the provided code; instead, remember that learning to program requires (yes, requires) lots and lots of trial and error.

That actually is a point well worth repeating: To become effective with Windows PowerShell, you need to use it daily. Don’t complain about retyping my code examples. Instead, look at it as an opportunity for you to practice.

**System Requirements**

You don’t need a heck of a lot, computer-wise, to perform all the Try It Yourself exercises in this book. However, if you do not meet the necessary system requirements, you are stuck. To that end, make sure that you have the following met prior to beginning your work:

- **A Windows-based computer**: Technically, you don’t need a computer that runs only Microsoft Windows. For instance, I use VMware Fusion to run Windows 8 virtual machines (VMs) on my OS X computer. No matter how you slice it, though, Windows PowerShell has *Windows* in its name for a reason, so you’ll be stuck at the starting gate unless you have a Windows machine at your disposal.
An Internet connection: In learning Windows PowerShell, you’ll be hitting the Web all the time to gain additional insight, obtain code examples, and so forth. Moreover, because Windows PowerShell doesn’t ship with local help files, you’ll need an Internet link to download those at least once.

A VM network and an Azure subscription: You can build a two- or three-node practice network for free. Windows 8.1, for instance, includes Hyper-V. You can also download Oracle VM VirtualBox to deploy a VM-based network. Microsoft is kind enough to offer full-feature evaluation editions of their software, so you shouldn’t have to pay big bucks for licenses. Along those lines, Microsoft offers trial subscriptions of their Microsoft Azure subscription service. As I wrote this book, I made sure that replicating my network environment was as painless as possible for you because I want you to work through every single example in the book to maximize your learning.

Design Elements Used in This Book

Some code statements presented in this book are too long to appear on a single line. In these cases, a line-continuation character (➥) is used to indicate that the following line is a continuation of the current statement.

NOTE

Items of Interest
Notes offer interesting information related to the current topic.

TIP

Useful Tidbits
Tips offer advice or show you an easier way to perform a task.

CAUTION

Potential Pitfalls
Cautions alert you to a possible problem and suggest ways to avoid it.

Okay, that’s enough of the preliminaries. It’s time to learn how to use Windows PowerShell.
HOUR 18
Desired State Configuration Basics

What You’ll Learn in This Hour:
▶ Historical background of DSC
▶ Basic tenets of DSC
▶ DSC authoring environment
▶ Configuring the DSC environment
▶ Writing your first configuration script
▶ A word on DSC push configuration

Desired State Configuration, also called DSC, is the marquee feature in Windows PowerShell v4 and later. Imagine being able to send configuration instructions to your servers such that, with no tedious mouse clicking on your part, the target servers simply (to quote Jean-Luc Picard from Star Trek: The Next Generation) “Make it so.”

I’m not kidding, either. In this hour, you’ll learn precisely what DSC is and how it works, and you’ll see its value proposition with your own eyes. Many of my IT professional colleagues whisper that DSC may very well spell the future standard for Windows server configuration and administration. Let’s make it so!

Historical Background of DSC
Windows PowerShell Principal Architect Jeffrey Snover wrote The Monad Manifesto in 2002, and in so doing outlined what he saw as the chief capabilities of a new command-line automation language for Windows.

Amazingly, Snover and his team at Microsoft realized every major point in that document. Specifically, with the Manifesto’s fourth point, Monad Management Models, they describe the basic elements that the team ultimately delivered in Windows PowerShell v4.
DSC is a Windows PowerShell-based system configuration platform. Here’s the scenario: You and/or your colleagues spend valuable hours manually configuring your Windows servers; I’m talking about tasks such as the following:

- Installing and configuring roles and features
- Installing and configuring other system software and services
- Deploying and maintaining file shares
- Managing Registry settings and environment variables

The preceding list barely scratches the surface of the myriad configuration events that must be performed on each server for that machine to be considered compliant by your organization.

However, if you have any degree of Windows systems administration experience, you know that “configuration drift” is a sad fact of life. Joe Administrator makes one setting, and then a week later Jane Administrator undoes said setting.

The configuration drift problem is all fun and games until questions of service level agreements (SLAs), licensure requirements, and industry/governmental regulations come knocking at your door, metaphorically speaking.

Long story short: DSC fills a need for us Windows server administrators, now more than ever before.

**Competitive Landscape**

Remember that Jeffrey Snover and the Windows PowerShell team are almost all longstanding experts in UNIX/Linux administration and systems programming. This fact should be patently obvious when you compare, say, the day-to-day operation of the Bash shell with how the Windows PowerShell command-line environment behaves.

To that point, there’s no denying the fact that Snover & Co. took a leaf from the competition’s playbooks with regard to this automated systems configuration framework “thing.” Specifically, two market leaders in the systems configuration/automation space are also (partially) open source projects:

- Chef (http://www.chef.io)
- Puppet (http://puppetlabs.com)

Don’t get too bent out of shape, though: Not only are Chef and Puppet compatible with Windows, but Microsoft Azure offers either configuration product as an option for their hosted virtual machines. Figure 18.1 shows a representative screenshot of Puppet.
Basic Tenets of DSC

To begin, you should understand that most DSC configuration involves using Windows PowerShell and the vendor-neutral Managed Object Format (MOF) in a declarative fashion. In programming, declarative code does not spell out exactly how the computer should complete a task. Instead, the code essentially tells the computer to “make it so” however it sees fit.
Structured Query Language (SQL) is a good example of a declarative data access language. When you run a complex SELECT statement, for instance, you leave it to the database itself to determine the system of index/row lookups it uses to satisfy the query results.

Likewise, in DSC, we start by describing how we’d like our servers to look in a standard Windows PowerShell configuration script. Take a look at Figure 18.2, and I’ll explain how DSC works step by step.

**DSC Authoring Environment**

As you saw in Figure 18.2, DCSERVER01 represents our DSC authoring environment. It is on this box, which must be equipped with at least Windows Management Framework v4 or later, that we construct our configuration script.

The configuration script is a bread-and-butter Windows PowerShell file that contains the configuration instructions for one or several target systems. The configuration script is compiled into the vendor-neutral MOF and then transferred to the target systems for ingestion.

**DSC Production Environment**

A component of Windows Management Framework (WMF) 5 called the Local Configuration Manager (LCM) running on the target system is what receives the MOF and applies its configuration settings to the box.

DSC supports two modes for getting the MOF configuration file to the target computer. In the push model, we use the **Start-DSCConfiguration** cmdlet to initiate the MOF push.
In the pull model, the client computer polls an Internet Information Services (IIS) website running on your DSC deployment server and requests any MOF files that are specified for it.

In terms of query intervals, target nodes query the pull server every 30 minutes by default. In the push architecture, nodes reevaluate their MOF file settings every 15 minutes by default if the configuration file had autocorrection enabled. As with anything else in Windows PowerShell, you can edit those query defaults.

Finally, as you observed in Figure 18.2, something called “DSC resources” exist on both the authoring and production servers. We can consider DSC resources to be specialized Windows PowerShell modules that actually form the imperative “engine” that nodes use through their LCM to apply their desired state configurations.

**Differences Between DSC and Group Policy**

Some Windows systems administrators wonder, “What’s the difference between DSC and Group Policy?” One difference is that DSC permanently “tattoos” the configuration settings of target nodes. You’ll recall that once a Group Policy Object (GPO) no longer applies to a machine, those settings can revert to their pre-GPO values.

Another difference is that a single node can have only a single MOF file defining a particular configuration (installing and configuring IIS, for instance). By contrast, we can link multiple GPOs to each of the various Active Directory levels (site, domain, organizational unit, and local computer). Finally, GPOs grant management access principally to the computer’s registry, while DSC MOF resources can “touch” any computer subsystem that’s accessible by PowerShell and, by extension, the .NET Framework.

The bottom line is that DSC won’t necessarily replace GPOs for systems configuration. Remember the focus with DSC, at least at this point, is to declaratively configure our servers such that “configuration drift” and deviation from compliance is no longer an issue for us.

Before we can test out DSC, we need to first prepare our environment.

**Configuring the DSC Environment**

Don’t even think about testing, much less deploying, DSC unless all of the following are true:

- All participating computers have WMF 4.0 or later installed.
- All participating servers have Windows PowerShell remoting enabled.
- All Windows Server 2012 R2 and Windows 8.1 nodes have hotfix KB2883200 installed.
You can leverage Windows PowerShell to verify if that required hotfix has been applied to your system:

```
PS C:\> Get-HotFix -Id KB2883200
```

<table>
<thead>
<tr>
<th>Source</th>
<th>Description</th>
<th>HotFixID</th>
<th>InstalledBy</th>
<th>InstalledOn</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSCSRVR01</td>
<td>Update</td>
<td>KB2883200</td>
<td>COMPANY\trainer</td>
<td>9/30/2013</td>
</tr>
<tr>
<td>12:00:00AM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Windows PowerShell remoting is required because the deployment of DSC MOF files uses Web Services-Management / Windows Remote Management (WS-Man/WinRM).

**A New Microsoft**

In past years, Microsoft took a highly proprietary approach to how their own products interoperated (or didn’t) with those of other vendors, especially open source community projects. Jeffrey Snover went to great lengths to establish Microsoft corporate buy-in for interoperability, and this argument has paid huge dividends with cross-platform capabilities such as WS-Man, Windows Management Instrumentation / Common Information Model (WMI/CIM), and the MOF format. The idea that today we can use DSC to configure Linux computer was utterly inconceivable not too long ago.

You also need to enable the DSC bits on all participating nodes. (*Nodes* is a more descriptive term than *server* because technically DSC can be used in both server and desktop Windows versions.) From an elevated Windows PowerShell console prompt on a Windows Server box, you can run the following:

```
Install-WindowsFeature
```

Of course, we can also use Server Manager (on servers) or Windows Features (on clients) to enable DSC, as shown in Figure 18.3.
Configuring the DSC Environment

FIGURE 18.3
Here we enable Windows PowerShell DSC in Windows Server 2012 R2 (top) and in Windows 8.1 (bottom).

Loading Up DSC Resources
As of this writing, Microsoft gives us 12 in-box resources in WMF v4. These resources and their uses are as follows:

▶ **Archive**: Zipping and unzipping archives
▶ **Environment**: Managing environment variables
▶ **Group**: Managing local groups
▶ **Log**: Writes messages to the Microsoft-Windows-DSC/Analytic event log
▶ **Package**: Installs .msi or Setup.exe software
▶ **Registry**: Managing the computer and user Registry hives
▶ **Script**: Excellent as a “catchall” resource when you can’t get what you need from an existing DSC resource
If you run the following command:

```
Get-DSCResource | Select-Object { $_.parentpath }
```

you’ll see that your built-in DSC resource folders are placed deep in the Windows\system32 hierarchy:

```
C:\Windows\System32\WindowsPowerShell\v1.0\Modules\<modulename>
```

That’s all well and good, but when you need to install your own modules, you should place them in this path:

```
C:\Program Files\WindowsPowerShell\Modules
```

Specifically, you should place the unzipped resource folder directly inside Modules. For instance, in Figure 18.4, I show you where I placed the `xActiveDirectory` experimental module that I downloaded to my server via `OneGet`.

Figure 18.4
Here we see where to place additional DSC resources on a node’s file system. Notice that a DSC resource looks and “feels” an awful lot like a traditional Windows PowerShell script module.
If your nodes are equipped with PowerShell v5 preview (which they shouldn’t unless v5 has been finalized as of your reading this), I suggest you look for DSC resources by querying the repositories:

```
Get-Package -Name x*
```

The x prefix is used to denote prerelease or eXperimental resource packages. Therefore, you use them in production at your own risk.

**DSC Resource Waves**

Aside from OneGet repos, your best bet for discovering useful DSC resources are the DSC Resource Kit “waves” that are regularly released by the Windows PowerShell team. Each wave brings new resources to the table that allow you greater administrative control over more and more products. Sometimes you’ll find that a newer wave release includes updated resources that supersede previously released versions. (The x in experimental is taken very seriously by the PowerShell community.)

Sadly, the DSC resource kit waves aren’t presented in a strictly linear fashion, which can make it tricky figuring out what’s what. To help you along, I’ll pass on the links for the nine wave announcements that are extant as of this writing:


Again, you simply download the resources, unzip them into the proper directory, and run `Get-DSCResource` to verify that they show up. Recall also that you need to install the resources on all participating nodes.
Writing Your First Configuration Script

Okay, it's time to start building out our DSC infrastructure, the first step of which is authoring our configuration script. Remember that although target nodes can apply only one MOF file for a given configuration, you can apply multiple MOFs to a single host as long as you don’t have conflicting configuration definitions. I’m sure that, over time, the Windows PowerShell team will make it easier for administrators to manage these manifold MOF manifests (alliteration alert).

I want you to understand before we get started that creating the MOF files via a PowerShell configuration script represents only one possibility for creating the MOFs. If, perchance, you understood MOF syntax, there's nothing stopping you from creating your own MOFs from scratch using only a text editor.

In other words, we should start to see MOF authoring tools emerge from independent software vendors (ISVs) and the community at large as we progress over time. Welcome to the world of vendor neutrality and community-driven software architectures.

More About MOF Files

Remember that the MOF is not a Microsoft proprietary format, but instead is a vendor-neutral data representation format developed by the Distributed Management Task Force, of which Microsoft is a member.

MOF is used to define both management objects in CIM/WMI, and is also closely related to Web-Based Enterprise Management (WBEM) protocols such as WS-Man.

Figure 18.5 shows you what a typical MOF file looks like. I can’t stress enough that DSC is a potentially vendor-neutral technology, and the tool that you use to create the MOF doesn’t have to be Windows PowerShell. I submit that we’ll see graphical user interface (GUI) MOF creation utilities for DSC not too long in the future; perhaps these tools already exist by the time you’re reading this book.

```powershell
Configuration SampleConfig1
{
    Node "dscclient01"
    {
        File CopyScript
        {
            Ensure = "Present"
            Type = "Directory"
            SourcePath = "\dscserver01\scripts"
            DestinationPath = "C:\scripts"
        }
    }
}
SampleConfig1
```

FIGURE 18.5
A MOF file can be created by using PowerShell, another utility or programming language, or from scratch. As long as the MOF uses legal syntax, the method by which you produce the file is irrelevant.
Writing Your First Configuration Script

Spend a moment studying the configuration script code in Figure 18.6, and I'll walk you through each line. I strongly suggest you write your DSC configuration script in the Windows PowerShell integrated scripting environment (ISE) so that you can take advantage of IntelliSense and the easy script execution controls.

▶ Line 1: We use the `Configuration` keyword in our script to denote a DSC configuration file. The configuration name is arbitrary.

▶ Line 2: The `Configuration` element is enclosed in top-level curly braces.

▶ Line 3: The `Node` keyword specifies the target node. In this first example, we're hard-coding the name of a Windows Server 2012 R2 host named dscclient01. In a later example, we'll parameterize this element with a variable so we can use one config script to target multiple nodes.

▶ Line 4: Indenting curly braces is optional, but an excellent practice to minimize the chance of our forgetting to close a script block and generate a runtime script failure.

▶ Line 5: The “meat and potatoes” of the configuration script are these subblocks. Here we specify the File DSC resource type, passing in an arbitrary name.
Desired State Configuration Basics

Line 6: Another indented curly brace, this time enclosing the File resource script block.

Line 7: Each DSC resource contains a number of named parameters. Like anything else in PowerShell, read the resource’s online documentation to learn the acceptable values for each parameter. The **Ensure=”Present”** line is ubiquitous in DSC configuration scripts, in my experience. This ensures that the policy is enforced.

Lines 8-10: Here we plug in the details for our File DSC resource declarations. What we’re doing is copying the scripts shared folder on my deployment server to a local path on the target node. As of this writing, I needed to add the source and destination node computer accounts to the shared folder’s discretionary access control list (DACL) to make a UNC path work.

Lines 11–14: Here we close up all the script blocks.

Line 15. This is an optional line in which we call the configuration. That way we actually execute the Configuration block when we run the script in the Windows PowerShell ISE.

When you’re ready, run the entire configuration script. If all goes well, you’ll see output that is similar to this:

```
PS C:\Users\Trainer> C:\Users\Trainer\Desktop\DSC\SampleConfig1.ps1

Directory: C:\Users\Trainer\SampleConfig1

Mode     LastWriteTime      Length Name
----     -----------      ------ ----
-a----   12/22/2014 8:23 AM  1736 dscclient01.mof
```

You’ll note that Windows PowerShell does a couple things when you run the DSC configuration script:

- Creates a directory in the root of the C: drive with the same name as the .ps1 script file
- Creates one MOF for each node referenced in the script; the MOF files are named with the node’s hostname

Customizing the Local Configuration Manager

Earlier in this hour, I told you that all DSC-enabled Windows nodes have a client component called the Local Configuration Manager (LCM) that is installed as part of WMF v4.

We can (and probably should) push a separate configuration script to our target nodes to customize the deployment parameters. First, let’s run `Get-DscLocalConfigurationManager` to see what’s what on my dscserver01 machine:
Some of the LCM parameters are more important than others. The **ConfigurationMode** parameter tells the node what to do in terms of how it applies and refreshes DSC configurations. The options here are as follows:

- **Apply**: Applies the configuration once and then doesn’t check for an update or refresh again (one-time application, in other words).

- **ApplyAndMonitor**: Applies the configuration and continues to validate that the node is in compliance with the policy. If configuration drift occurs, the node does nothing.

- **ApplyAndAutoCorrect**: Applies the configuration, periodically checks for compliance, and reapplies the configuration if something changes within the scope of active configurations.

Note also the **RefreshFrequencyMins** parameter. In push mode, the node checks for DSC compliance every 30 minutes. This may be far too frequent for your business needs, so let’s deploy a new set of LCM settings to our localhost and dsclient01 nodes.

Once again, take a look at our script shown in Figure 18.7, and I’ll walk you through selected parts:
By deploying an LCM configuration, we can take fine-grained control over how DSC policies are evaluated and applied by target nodes.

- **Lines 3–5**: These lines create an input parameter for our LCM script. Note that the `$NodeName` parameter is defined as a string array, `[]`, which makes it a snap to target multiple nodes without having to repeat code blocks in the script file.

- **Line 10**: Here we specify a 4-hour refresh interval for the LCM policy refresh mode.

- **Line 17**: Again for convenience, we run the configuration in-line with the code, specifying two target nodes by hostname. Of course, you can import the script into your runspace by using dot sourcing. However, I like the convenience of calling the function directly in the script file. Your mileage may vary, as I’ve said in this book about a hundred times before.

An exhaustive discussion of how to import scripts into your runspace using dot sourcing is included in Hour 19, “Introduction to Windows PowerShell Scripting.”

When you run the LCM config script, you wind up with a single “meta” MOF regardless of how many nodes you target in the script. Likewise, we use a different cmdlet to apply an LCM script: `Set-DscLocalConfigurationManager`. The `-Path` parameter points to the directory that contains our LCM script:

```
Set-DscLocalConfigurationManager -Path "C:\SetupLCM"
```

Let’s do a Try It Yourself exercise so that you can shore up your Windows PowerShell skills and see how DSC works with your own eyes.
Creating and Pushing a DSC Configuration

In this Try It Yourself exercise, you’ll apply much of the PowerShell skills you’ve accrued throughout the book to apply a specific configuration to a target node.

Specifically, you’ll configure a Windows Server 2012 R2 member server named dscclient01 to keep the Internet Information Services (IIS) web server installed and the default website stopped.

We’ll start by using OneGet to download and install the xWebAdministration custom DSC resource module. Next we’ll author our configuration script and push it to a target node. Finally, we’ll verify that the configuration “took” by intentionally producing configuration drift and testing autocorrection. If you don’t have WMF v5 installed on your nodes, go with v4 and simply download the xWebAdministration DSC resource package from TechNet (http://bit.ly/13VAcvz).

1. On your DSC authoring server (dscserver01 in my case), fire up an elevated PowerShell v5 session and install the xWebAdministration package:

   Find-Package -Name "xWebAdministration" | Install-Package -Verbose

   Remember that you need to run this command on all DSC nodes, which means both my authoring server as well as my dscclient01.company.pri target node.

   You’ll also want to verify that the DSC resource has been installed in the proper location on disk:

   PS C:\Program Files\WindowsPowerShell\Modules> dir

   Directory: C:\Program Files\WindowsPowerShell\Modules

   Mode           LastWriteTime    Length Name
   ------           -------------    ------ ----
   d----- 12/22/2014 12:49 PM 625           xWebAdministration
2. Now open an elevated ISE instance and create a new .ps1 script file named `WebServerConfig.ps1`. Check out Figure 18.8, and I’ll walk you through the most important code lines, as has become my habit:

```powershell
configuration SetupIIS
{
    param
        ($nodeName = 'localhost')
    Import-DscResource -Module xWebAdministration
    Node $nodeName
    {
        # Install the IIS server role
        WindowsFeature IIS
        {
            Ensure       = "Present"
            Name         = "Web-Server"
        }
        # Stop the Default Web Site website
        xWebsite DefaultSite
        {
            Ensure       = "Present"
            Name         = "Default Web Site"
            State        = "Stopped"
            PhysicalPath = "C:\inetpub\wwwroot"
            DependsOn    = "[WindowsFeature]IIS"
        }
    }
    SetupIIS -NodeName ($"dscserver01", "$dscclient01")
}
```

**FIGURE 18.8**

Our DSC configuration script ensures that IIS is installed and that the Default Web Site website is stopped.

**Line 5:** Once again, we parameterize the `Node` name to make the script more flexible.

**Line 8:** This is a bit of “smoke and mirrors.” We need to run `Import-DscResource` to load our custom DSC resource into the runspace. However, this is a “dynamic keyword” and is not an honest-to-goodness PowerShell cmdlet.

**Lines 13-17:** Here we invoke the built-in `WindowsFeature` resource to ensure that IIS is installed on the target node.

**Lines 19-25:** Now we call our `xWebSite` custom DSC resource to stop the default website.

**Line 25:** The `DependsOn` property is helpful when a configuration setting will work only if another one is active. Logically, then, we understand that we can stop the default website only if there exists an IIS web server to begin with.

**Line 29:** Modify the call to the configuration to target your own machines.
3. Run your WebServerConfig.ps1 script by pressing F5 in the integrated scripting environment (ISE) and verify that PowerShell created two MOF files in a separate directory named after the script file.

4. When you’re ready, unleash the proverbial hounds and apply the new configuration by running `Start-DscConfiguration`:

   ```powershell
   Start-DscConfiguration -Path "C:\SetupIIS"
   ```

5. Because PowerShell runs DSC configuration pushes as background jobs, we can use traditional syntax to check on job status:

   ```powershell
   PS C:\> Receive-Job -id 7 -Keep
   PSComputerName
   ------------
   dscclient01
   dscserver01
   ```

   Cool. No errors on my end. How has everything gone in your neck of the woods?

6. Connect to one of your target modes and see whether you can start the IIS Manager. If so, is the default website stopped? Figure 18.9 demonstrates my dscclient01 machine's compliance.

![Figure 18.9](image)

**Figure 18.9**

I’m just going to go ahead and say it: DSC rocks! Here we see the configuration applied to my dscclient01 member server.
7. Use Windows PowerShell on one of your target nodes to start the default website. Of course, this will produce configuration drift. (If you haven’t configured your LCM to perform autocorrection, go back and do that now.)

   Start-Website -Name "Default Web Site"

8. If you want, you can simply wait for the next DSC LCM refresh interval to test whether your server turned the default website back off. Alternatively, and perhaps more conveniently, we can force a manual update:

   Update-DscConfiguration

   The update will once again exist as a configuration background job. Note also that you can try Get-DscConfigurationStatus to review a node’s current relationship to DSC.

9. Surprise! You should find that the Update-DscConfiguration job fails:

   PS C:\> Receive-Job -Name Job4 -Keep
   No attempt was made to get a configuration from the pull server because LCM RefreshMode is currently set to Push.

   CategoryInfo          : NotSpecified:
   (root/Microsoft/...gurationManager:String) [], CimException
   + FullyQualifiedErrorId : MI RESULT 1
   + PSComputerName : localhost

   Here’s the deal: DSC push mode is great for test/demo situations because it’s easy to set up. However, we’ll have to run Start-DscConfiguration again from the authoring computer to refresh this policy. It’s only in a pull server scenario that nodes have the ability to refresh their policies. This makes sense because in a client refresh, we need some server from which the client can check to verify it has the correct policies applied.

---

A Word on DSC Push Configuration

Due to space constraints in this book, I’ll simply give you the barebones, “need-to-know” information regarding setting up a DSC pull server. Let’s do that in a stepwise fashion, covering the highest-level steps:

1. Download and install the xPSDesiredStateConfiguration custom DSC resource from the TechNet Script Center or by using OneGet.


3. Create and deploy an LCM configuration script. You can find an excellent example at Pwsh.net (http://bit.ly/1ARmIAl).
These settings are important because we change the configuration mode from push to pull and we specify the URL of the pull server’s web service. We also specify how long the client waits before updating its DSC policies.

The communication between the node and the web service occurs over HTTP or HTTPS, depending on your authentication requirements. That’s an important point, actually; you want to do what you can to ensure that your nodes are pulling configuration from legitimate DSC pull servers. It would be a very bad day indeed if a malicious individual stood up a bogus pull server and borked up your DSC client nodes in the absence of Secure Sockets Layer / Transport Layer Security (SSL/TLS) server authentication.

**Summary**

This was an awesome hour of training, wasn’t it? I hope you’re as stoked about DSC as I am. I don’t know about you, but I can’t stand manually (re)configuring servers. Declaring how a server “should” look and letting DSC take care of maintaining compliance to that configuration is plain old awesome.

In Hour 19 we’ll stay within the ISE because it’s finally time for us to take charge of Windows PowerShell scripting. (And here you never thought you’d be a programmer.)

**Q&A**

**Q.** I know that many PowerShell cmdlets have a -WhatIf flag that allows you to test a cmdlet before it runs. Is there a command or parameter we can use to verify that DSC is functioning on a node?

**A.** Yes, indeed. You’ll want to run `Test-DscConfiguration -Verbose` to instruct Windows PowerShell to process all of its DSC scripts; the output returns True if all tests pass.

**Q.** Will you please look at my script? I ran the following two lines of code:

```
PS C:\> .\LCMConfig.ps1
PS C:\> SetupLCM -NodeName "dscclient01"
```

and got a bunch of red error text, as shown in Figure 18.10. What’s the problem?
FIGURE 18.10
You need to understand the implications of “dot sourcing” your Windows PowerShell scripts.

A. The dot-slash (\.) notation simply tells PowerShell to run the present command from the current working directory and nothing more. This means that running a PowerShell script in this way runs the code contained inside the script but removes any functions, variables, and so forth from the session immediately thereafter.

Dot sourcing occurs when you type a period (.) and then type a partial or full path to a PowerShell script. (Don’t forget to put quotes around the script path, including the dot slash.) The key difference with dot sourcing is that any objects defined inside the script persist in the user’s current runspace. This allows us to run the DSC configuration script manually as was depicted in Figure 18.10.

Q. What is the suggested configuration refresh frequency for DSC?
A. How you configure your nodes’ LCM component, and particularly the ConfigurationFrequencyMins property value, depends entirely on how much tolerance for configuration drift you have.

DSC network traffic is relatively low compared to, say, Group Policy. However, many Windows administrators are cool with setting refresh to 48 hours because the likelihood that a server will fall out of compliance may not be particularly high.
Workshop

Create a DSC configuration file that performs the following two tasks:

▶ Ensures that the Shutdown Event Tracker is enabled
▶ Ensures that the Google Chrome web browser is installed

The only hint I’ll provide is that you need both the Registry built-in DSC resource and the xChrome custom resource to complete the configuration.

Quiz

1. You’d like to see what options are available for the WindowsFeature DSC resource. Which of the following commands accomplishes that goal?
   a. Get-Package
   b. Get-DscResource
   c. Get-Job
   d. Get-DscLocalConfigurationManager

2. The UpdateDscConfiguration cmdlet can be used only in DSC push scenarios.
   a. True
   b. False

3. A DSC authoring server running WMF 4.0 can push a configuration to a server running WMF 5.0.
   a. True
   b. False

Answers

1. The correct answer is B. Here is a run of the WindowsFeature DSC resource properties from my Windows Server 2012 R2 domain controller:

   PS C:\> (Get-DscResource -Name WindowsFeature).Properties

   Name               PropertyType     IsMandatory     Values
   ----               -----------        -----------     ------
   Name               [string]         True {}         
   Credential         [PSCredential]   False {}        
   DependsOn          [string[]]       False {}        
   Ensure             [string]         False {Absent, Present}
   IncludeAllSubFeature [bool]        False {}        
   LogPath            [string]         False {}        
   Source             [string]         False {}        

   Remember that you can run Get-Command -module PSDesiredStateConfiguration to retrieve a list of all DSC commands.
2. The correct answer is B. **Update-DscConfiguration** refreshes the target node only when a DSC pull server is online and available.

3. The correct answer is A. Remember that backward compatibility is a priority for the Windows PowerShell team. To use DSC, all your nodes must have the DSC bits available, which means that your Powershell version is 4 or 5 and your host operating system is Windows Server 2012/R2 or Windows 8/8.1.
Symbols

& (ampersand) operator
- running external commands, 84-85
- running scripts, 390
- in SharePoint Server management shell, 454

< > (angle brackets), 405

* (asterisk) wildcard character
- finding cmdlets, 57, 77-78
- as Kleene star, 316
- as placeholder, 125
- in regular expressions, 320
- in searches, 313-314

\ (backslash), 320

^ (caret), 318-320

: (colon), 156

{ } (curly braces)
- for expressions, 214
- in functions, 384
- in regular expressions, 322

$ (dollar sign), 318-320

$_ (dollar sign underscore) variable, 129

` (grave accent), 128, 224, 406

> (greater than)
- for output redirection, 142, 187
- in WQL, 302

< (less than), 302

# (octothorpe), 129, 393

) (parentheses), 165

% (percent sign), 302

. (period), 320

\. (period backslash) notation, 390

| (pipe), 20, 113, 128

? (question mark)
- as alias, 138-139
- as wildcard character
  - in regular expressions, 320
  - in searches, 315

" " (quotation marks)
- in directory path, 84
- double quotes versus single quotes, 385
- in regular expressions, 321

; (semicolon), 73

-% (stop parsing symbol), 85-86

~ (tilde), 314

7-Zip, installing, 340-341

Flexible Single Master Roles (FSMOs)
- managing, 430-431
- transferring, 431-432

functional levels, setting, 432

Group Policy administration, 432-433

groups, creating, 424-425

installing, 417-423

organizational units (OUs),
creating, 423-424

user accounts
- creating, 425-426
- creating in bulk, 427-430
- managing, 426-427

Active Directory Administrative Center (ADAC), 17, 430

Active Directory Cookbook
(Svidergol and Allen), 423

Active Directory provider, 169-170

activities for workflows, 283-284

CheckPoint-Workflow cmdlet, 284-285

Resume-Workflow cmdlet, 284-287

Sequence, 284, 287-288

Suspend-Workflow cmdlet, 284-287

ADAC (Active Directory Administrative Center), 17, 430

Add-ADGroupMember cmdlet, 425

Add-Computer cmdlet, 418

Add-Content cmdlet, 164

add-ons
- Commands add-on, turning on/off, 40

A

About help files, 60, 119, 167

accessing command history, 33

Active Directory
- child domains, installing, 422-423
- domain controllers, promoting, 421-422
- domains
  - joining, 418-419
  - testing for membership, 418
add-ons

for ISE, 384
Script Browser ISE add-in, installing, 391
Add-PSSnapin cmdlet, 70, 401
Add-PSWAAuthorizationRule cmdlet, 243
Add/Remove Programs (ARP), 335
ADDSDeployment module, 421
Add-WindowsFeature cmdlet, 419
administrator, running Windows PowerShell as, 15
Adobe Portable Document Format (PDF), 308
Advanced Package Tool (APT), 332
Alias provider, 154, 166-167 aliases
? (question mark), 138-139
cd, 136, 155, 157
creating, 64-65
dir, 80, 136, 156
exit, 210
gcm, 77, 79, 133
gm, 100
group, 136
ls, 136, 156
measure, 145
ps, 80
purpose of, 166
select, 138
sl, 157
sort, 134
for Variable PSDrive, 156-157
where, 139
AliasProperties, 99
Allen, Robbie, 423
AllSigned execution policy, 378
Allversions parameter for
Find-Package cmdlet, 338
alternate credentials for remote sessions, 212-214
Amazon Web Services, 472
Amin, Adnan, 466
ampersand (&) operator
running external commands, 84-85
running scripts, 390
in SharePoint Server management shell, 454
anchors in regular expressions, 318-320
angle brackets (<>, 405
APT (Advanced Package Tool), 332
Archive resource (DSC), 361
arguments, passing remotely, 231-232
ARP (Add/Remove Programs), 335
arrays
defined, 128
hash tables, 128-129
AsJob parameter
for running jobs, 263-265
for workflows, 282
asterisk (*) wildcard character
finding cmdlets, 57, 77-78
as Kleene star, 316
as placeholder, 125
in regular expressions, 320
in searches, 313-314
at (@) signs, 446
attribute table for parameters, 56-57
attribute types for parameters, 118
authentication
authorization rules versus, 244
defining, 243-244
automating modules, 402-403
automation scripts, creating, 385-387
Autoruns, 436
AutoSize parameter
for Format-Table cmdlet, 181-182
for Format-Wide cmdlet, 186
Azure. See Microsoft Azure

B
\b character class, 326
background jobs. See jobs
backslash (\), 320
backtick (´). See grave accent (´)
backups, scheduling, 447-449
Backup-SQLDatabase cmdlet, 448
Bginfo, 346
binary modules, 402
browser tools for WMI, 296-299
browsing. See viewing
bulk processing Active Directory user accounts, 427-430
business logic, regular expressions and, 323
Bypass execution policy, 378

call operator (&). See ampersand (&) operator
calling methods in WMI, 304-305
cancel button (PSWA), 247
capabilities property for
Get-PSProvider cmdlet, 154-155
caret (^), 318-320
Cascading Style Sheets (CSS), 195
case sensitivity
of cmdlets, 20, 77
of comparison operators, 143
caseSensitive parameter for
Sort-Object cmdlet, 136
cd alias, 136, 155, 157
CEC (Common Engineering Criteria), 8
certification exams, 9
certificate classes in regular expressions, 321-323
checkpoints in workflows, 284-285
checkPoint-Workflow cmdlet, 284-285
Chef, 356
child domains, installing, 422-423
child jobs, 261-262
choco client, 345
Chocolately, 332-333
  finding packages, 338
  installing, 337
  trusting, 346-347
  upgrading packages, 347-348
  usage example, 344-346
ChocolatelyGUI, 348
CIM (Common Information Model)
  cmdlets for, 305
  remote access, 305
  usage example, 305-307
defined, 293-294
  location, 294-295
  terminology, 296
Class parameter for
  Get-WMIObject cmdlet, 301
classes.
  See also data types
  creating, 99
  defined, 48, 94-95
  in WMI
  defined, 296
  list of, 298-300
Classic Shell, 343
ClassName parameter for
  Get-CimInstance cmdlet, 305
cloud computing.
  See Microsoft Azure
CMatch parameter in regular expressions, 318
Cmd.exe
  running commands from, 83
  call operator (&), 84-85
  from environment path, 84
  Invoke-Item cmdlet, 84
  stop parsing symbol (--%), 85-86
  running scripts from, 390
  switching between
  PowerShell.exe and, 19
cmdlets.
  See also commands
  in Active Directory module, 419-421
  Add-ADGroupMember, 425
  Add-Computer, 418
  Add-Content, 164
  Add-PSSnapin, 70, 401
  Add-PSWAuizationRule, 243
  Add-WindowsFeature, 419
  Backup-SQLDatabase, 448
case sensitivity, 20, 77
  for CIM, 305
  remote access, 305
  usage example, 305-307
  Connect-MSOLService, 486
  Connect-PSSession, 225
  Connect-SPOService, 488
  ConvertTo-Csv, 193
  ConvertTo-Html, 195
  ConvertTo-SecureString, 425
  ConvertTo-Xml, 192, 194-195
  defined, 47
  Disconnect-PSSession, 210, 225
  Enable-PSRemoting, 205-207
  Enter-PSSession, 209-210
  Export-Clixml, 191-195
  Export-Csv, 189-191, 193
  finding, 58, 80
  Find-Package, 336, 338
  Format-List, 183-185
  Format-Table, 180-183
  Format-Wide, 185-187
  Get-ADComputer, 277
  Get-ADDomain, 430
  Get-ADForest, 430
  Get-ADGroup, 424
  Get-ADOrganizationalUnit, 424
  Get-Alias, 77
  Get-AzurePublishSettingsFile, 477
  Get-AzureSqlDatabaseServer, 484
  Get-AzureSubscription, 480
  Get-AzureVM, 479
  Get-AzureVMImage, 478
  Get-ChildItem, 136, 156, 160-162, 224
  Get-CimClass, 305
  Get-CimInstance, 305
  Get-Command, 71, 77-82
  Get-Content, 148, 164
  Get-Credential, 486
  Get-DscLocalConfiguration-Manager, 366
  Get-DSCResource, 362
  Get-EventLog, 19-21
  named parameters, 58
  parameter sets, 55-56
  positional parameters, 57-58
  required parameters, 57
  Get-ExecutionPolicy, 276, 378-379
  Get-Help, 50-51, 71
  Get-Command cmdlet
  versus, 79-82
  ShowWindow parameter, 61-62, 72
  viewing help system
  content, 52-53
  Get-HotFix, 360
  Get-ItemProperty, 169, 224
  Get-Job, 256-257, 262
  Get-Member, 81, 100-102, 111
  finding property names, 135
  output describing
  metadata, 96-97
  Get-Module, 70
  Get-MsolDomain, 487
  Get-Package, 342, 347
  Get-PackageProvider, 334
  Get-PackageSource, 335
  Get-Process, 59-60, 80
  inputs and outputs, 112
  methods available, 101
  properties available, 100
  Get-PSDrive, 156
  Get-PSSessionComputer, 225
  Get-PSSessionConfiguration, 228
  Get-PSSesssionConfiguration
  Get-PSSessionConfiguration, 228
  Get-PSSnapin, 70, 180, 400
  Get-PswaAuthorizationRule, 243
  Get-ScheduledJob, 267
  Get-Service, 77
  inputs and outputs, 113-114
  parameter binding, 114-116, 119-121
  pipelines example, 113-118
  Get-SPManagedAccount, 458
  Get-SPOSite, 488
  Get-SPServiceApplication, 458
  Get-SPServiceInstance, 458
  Get-SPSite, 462, 464
  Get-SPWeb, 465
  Get-SPWebApplication, 464
  Get-SPWebTemplate, 460
  Get-Verb, 49, 76

Cmdlets
495
converting outputs

Start-DscConfiguration, 371
Start-Job, 256
Start-SPAssignment, 465
Start-SPServiceInstance, 458
Start-Website, 372
Stop-AzureVM, 481
Stop-Computer, 305
Stop-Job, 259
Stop-Process, 111
Stop-Service
  inputs and outputs, 113-114
  parameter binding, 114-116, 119-121
  pipelines example, 113-118
Stop-SPAssignment, 465
structure of, 48-49
nouns, 49-50
origin, 76
parameters, 49-50
prefixes, 48, 79
verbs, 49, 75-76
Suspend-Job, 260
tab completion, 82-83, 138
Test-ADDSDomainController-Installation, 421
Test-Connection, 83, 280
Test-Manifest, 409
Test-PswaAuthorizationRule
Uninstall-Package, 349
Uninstall-PswaWebApplication, 250
Unlock-ADAccount, 426
Update-DscConfiguration, 372
Update-Help, 52
Where-Object, 139-141, 317
for WMI, 300
calling methods, 304-305
Get-WMIObject, 300-301
remote access, 302-304
WQL (Windows Management Instrumentation Query Language), 301-302
Write-Host, 38, 385
Write-Output, 38, 385
CNotMatch parameter in regular expressions, 318
code folding, 16, 387
code highlighting, 16
CodePlex, 297
collections
  defined, 93
  output describing metadata, 96-97
colon (:), 156
color coding in ISE, 387
Colors tab (Properties sheet), 35
Column parameter for Format-Wide cmdlet, 186
command history
  accessing, 33
  customizing, 33
  viewing, 35
command prompt session
  switching between Windows PowerShell and, 19
commands. See also cmdlets
  for help system, 50-51
  man, 50
  running external commands, 83
  call operator (&), 84-85
  from environment path, 84
  Invoke-Item cmdlet, 84
  stop parsing symbol (--%), 85-86
  running in modules, 406
Commands add-on, turning on/off, 40
comma-separated value (CSV) files, outputting to, 189-191, 193
comment-based help
  in modules, 410-413
  in scripts, 395
comments
  in modules, 405
  in scripts, 393
Common Engineering Criteria (CEC), 8
Common Information Model (CIM). See CIM (Common Information Model)
comparison operators, 141-144
ComputerGroupName parameter for Get-PswaAuthorizationRule cmdlet, 244
ComputerName parameter for Invoke-Command cmdlet, 214
for one-to-many remoting, 221-224
for one-to-one remoting, 201-203, 214
without Session parameter, 222
conditional logic in scripts, 387
configuration scripts
  creating, 364-366
  defined, 358
  usage example, 368-372
ConfigurationMode parameter for Get-DscLocalConfigurationManager cmdlet, 367
ConfigurationName parameter for Get-PswaAuthorizationRule cmdlet, 244
configurations, 48. See also DSC ( Desired State Configuration)
configuring
  DSC environment, 359-360
  PSWA (Windows PowerShell Web Access), 240-243
Confirm parameter for Install-PSWAWebApplication cmdlet, 240
connecting
  to Microsoft Azure, 477
  to Office 365, 485-487
  to SharePoint Online, 487-488
Connect-MSOLService cmdlet, 486
Connect-PSSession cmdlet, 225
Connect-SPOService cmdlet, 488
console, 13-14
  customizing, 32-39
  starting, 18
Console pane
  positioning, 40-41
  zooming, 41
constrained endpoints, creating, 227-228
-contains comparison operator, 142
Contig, 346
continuation character (\), 73
Control menu, Properties sheet, 32
  Colors tab, 35
  Font tab, 33-34
  Layout tab, 34-35
  Options tab, 32-33
converting outputs, 193-195
Export-Clixml cmdlet versus ConvertTo-Xml cmdlet, 194-195
Export-Csv cmdlet versus ConvertTo-Csv cmdlet, 193
usage example, 196-197
ConvertTo-Csv cmdlet, 193
ConvertTo-Html cmdlet, 195
ConvertTo-SecureString cmdlet, 425
ConvertTo-Xml cmdlet, 192, 194-195
Count parameter for Test-Connection cmdlet, 280
counting with Measure-Object cmdlet, 72-73
CPU property for Get-Process cmdlet, 141
CreateSPGroupAddADGroupSetPermissionLevel.ps1 script, 463
Credential parameter for New-PSSession cmdlet, 212
Credentials capability, 155
CSS (Cascading Style Sheets), 195
CssUri parameter for ConvertTo-Html cmdlet, 195
CSV (comma-separated value) files, outputting to, 189-191, 193
Ctrl+Break (stop script), 42
curly braces ({ }) for expressions, 214
in functions, 384
in regular expressions, 322
CurrentUser scope, 379
cursor size, changing, 33
customizing
console, 32-39
ISE (Integrated Scripting Environment), 40-42
LCM (Local Configuration Manager), 366-368
data types
creating, 99
defined, 94-95
in formatting subsystem, 177-178
in pipelines, 113-114
databases
creating, 445, 481-484
listing in SQL Server instances, 443-444
scheduling backups, 447-449
DCL (DIGITAL Command Language), 76
debugger, 16
debugging pipeline, 112
declarative nature of DSC, 357-358
default file association for scripts, 377
defining authorization rules, 243-244
deleting
jobs from queue, 260-261
packages, 349
Delimiter parameter for Export-Csv cmdlet, 191
deploying
service applications, 457-459
site collections, 460-462
web applications, 459-460
Descending parameter for Sort-Object cmdlet, 134
Desired State Configuration (DSC). See DSC (Desired State Configuration)
digital certificates, 208
DIGITAL Command Language (DCL), 76
dir alias, 80, 136, 156
disabling. See turning on/off disconnected sessions
in PSWA (Windows PowerShell Web Access), 247-248
in remote access, 225
Disconnect-PSSession cmdlet, 210, 225
discovering. See finding
DMTF (Distributed Management Task Force), 294
DN (distinguished name) syntax, 423
documentation
on .NET classes, 98
in repositories, 339-340
dollar sign ($), 318-320
dollar sign underscore ($_) variable, 129
domain controllers
forest root domain controllers, 417
promoting, 421-422
domains
child domains, installing, 422-423
functional levels, setting, 432
groups, creating, 424-425
joining, 418-419
testing for membership, 418
DOS commands. See external commands
dot notation, 82, 104
dot source notation, 391
DotNetFrameworkVersion property for New-ModuleManifest cmdlet, 408
double quotes, single quotes versus, 385
downloading
Microsoft Azure module, 475-476
Microsoft Online Services Sign-in Assistant for IT Professionals, 485
Script Browser ISE add-in, 391
SharePoint Online Management Shell, 487
Windows Azure Active Directory Module for Windows PowerShell, 485
Drives property for Get-PSProvider cmdlet, 155
DSC (Desired State Configuration), 285
authoring environment, 358
configuration scripts, creating, 364-366
configuring environment, 359-360
customizing LCM, 366-368
declarative nature of, 357-358
Group Policy versus, 359
history of, 355-357
packages for, 338
production environment, 358-359
pull configurations, 369-373
resources
list of, 361-362
loading, 362-363
waves, 363
usage example, 368-372
dynamic modules, 402

E

editing
Registry values, 169
scripts, 391-394
ingedit options
changing, 33
Quick Edit mode, 38
EmailAddress parameter for New-ADUser cmdlet, 126
Enable-PSRemoting cmdlet, 205-207
enabling. See also turning on/off
DSC environment, 359-360
remote access, 205-209
Enable-PSRemoting cmdlet, 205-207
with Group Policy, 207-208
in workgroups, 208-209
Encoding parameter for Out-File cmdlet, 188
Enter-PSSession cmdlet, 209-210
enumerations, 101
Environment provider, 154
Environment resource (DSC), 361
environment variables, PATH, 84
-eq comparison operator, 141
error handling, 395
error pipeline, 112
ErrorAction parameter for Test-Connection cmdlet, 280
escape characters in regular expressions, 320
event logs, retrieving, 19-21
Example parameter for help system, 55
exception handling, 395

Exclude capability, 155
exclusion policies
scope, 379
for scripts
changing, 103-104, 276
list of, 378
viewing and setting, 378-379
exit alias, 210
Exit button (PSWA), 247
Exit-PSSession cmdlet, 209-210
Export-Dxml cmdlet, 191-195
Export-Csv cmdlet, 189-191, 193
ExportedCommands property for Test-Manifest cmdlet, 410
exporting
About help files, 60
outputs, 187-193
Export-Dxml cmdlet, 191-193
Export-Csv cmdlet, 189-191
Out-File cmdlet, 187-189
Out-Printer cmdlet, 189
usage example, 196-197
Extensible Application Markup Language (XAML), 283
Extensible Markup Language (XML), outputting to, 191-195
extensions, unhiding, 226
external commands, running, 83
call operator (&), 84-85
from environment path, 84
Invoke-Item cmdlet, 84
stop parsing symbol (-%), 85-86
Find-Package cmdlet, 336, 338
First parameter for Select-Object cmdlet, 139

F

F5 (run script) keyboard shortcut, 42
F7 (previous command buffer) keyboard shortcut, 35
F8 (run selection) keyboard shortcut, 42
Feldon-Lawrence, Ashley, 463
file associations for scripts, 377
file extensions, viewing, 226
File parameter, starting PowerShell with, 390
File resource (DSC), 362
file shares, creating private repositories, 350
file-based operating system,
UNIX/Linux as, 91-92
FilePath parameter
for Invoke-Command cmdlet, 215
for Out-File cmdlet, 187
for Start-Job cmdlet, 259
files, outputting to, 187-189
FileSystem provider, 154, 159-165
Get-ChildItem cmdlet and, 160-162
PSDrives for, 157
usage example, 162-165
Filter capability, 155
“filter left, format right”, 175
Filter parameter
for cmdlets, 140
for Get-ChildItem cmdlet, 161
for Get-WMIObject cmdlet, 302
for New-ADOrganizationalUnit cmdlet, 424
filtering
objects, 139-141
outputs, 137-144
comparison operators, 141-144
Select-Object cmdlet, 138-139
finding
cmdlets, 58, 80
modules, 70, 413-414
object members, 96-102
Get-Member cmdlet, 100-102
.NET classes in, 98
properties/methods, 99-100
packages in Chocolatey, 338
property names, 135-136
URLs (uniform resource locators), 476
Find-Package cmdlet, 336, 338
First parameter for Select-Object cmdlet, 139
Flexible Single Master Roles (FSMOs). See FSMOs (Flexible Single Master Roles)
Font tab (Properties sheet), 33-34
Force parameter
  for Enable-PSRemoting cmdlet, 207
  for Move-ADDirectoryServer-OperationsMasterRole cmdlet, 432
  for Stop-AzureVM cmdlet, 481
foreach construction, 386, 463
foreach keyword, 280
forest root domain controllers, 417
forests, setting functional levels, 432
Format-List cmdlet, 183-185
Format-Table cmdlet, 180-183
formatting outputs, 176-187
  Format-List cmdlet, 183-185
  Format-Table cmdlet, 180-183
  Format-Wide cmdlet, 185-187
  Out-GridView cmdlet, 179-180
  Type property, 177-178
  usage example, 196-197
Format-Wide cmdlet, 185-187
Friedl, Jeffrey, 323
FSMOs (Flexible Single Master Roles)
  managing, 430-431
  transferring, 431-432
Full Circle Blog, 463
Full parameter for help system, 54
  function keyword, 384
  functional levels of domains and forests, setting, 432
functions
  creating, 276-278, 383-385
  defined, 47, 383
  double quotes versus single quotes, 385
Help, 50-52
More, 51
scope, 395
snippets, 383-384
G
garbage collection, 401
  gateway (PSWA). See PSWA (Windows PowerShell Web Access)
gcm alias, 77, 79, 133
  -ge comparison operator, 141
Generate Document Info Report.ps1 script, 466
Get-ADComputer cmdlet, 277
Get-ADDomain cmdlet, 430
Get-ADForest cmdlet, 430
Get-ADGroup cmdlet, 424
Get-ADOrganizationalUnit cmdlet, 424
Get-Alias cmdlet, 77
Get-AzurePublishSettingsFile cmdlet, 477
Get-AzureSqlDatabaseServer cmdlet, 484
Get-AzureSubscription cmdlet, 480
Get-AzureVM cmdlet, 479
Get-AzureVMImage cmdlet, 478
Get-ChildItem cmdlet, 136, 156, 160-162, 224
Get-CimClass cmdlet, 305
Get-CimInstance cmdlet, 305
Get-Command cmdlet, 71, 77-82
Get-Command cmdlet, 148, 164
Get-Credential cmdlet, 486
Get-DscLocalConfiguration-Manager cmdlet, 366
Get-DscResource cmdlet, 362
Get-EventLog cmdlet, 19-21
  named parameters, 58
  parameter sets, 55-56
  positional parameters, 57-58
  required parameters, 57
Get-ExecutionPolicy cmdlet, 276, 378-379
Get-Help cmdlet, 50-51, 71
  Get-Command cmdlet versus, 79-82
  ShowWindow parameter, 61-62, 72
  viewing help system content, 52-53
Get-HotFix cmdlet, 360
Get-ItemProperty cmdlet, 169, 224
Get-Job cmdlet, 256-257, 262
Get-Member cmdlet, 81, 100-102, 111
  finding property names, 135
  output describing metadata, 96-97
Get-Module cmdlet, 70
Get-MsolDomain cmdlet, 487
Get-Package cmdlet, 342, 347
Get-PackageProvider cmdlet, 334
Get-PackageSource cmdlet, 335
Get-Process cmdlet, 59-60, 80
  inputs and outputs, 112
  methods available, 101
  properties available, 100
Get-PSDrive cmdlet, 156
Get-PSProvider cmdlet, 153-156
Get-PSSession cmdlet, 211, 225
Get-PSSessionConfiguration cmdlet, 228
Get-PSSnapin cmdlet, 70, 180, 400
Get-PswaAuthorizationRule cmdlet, 243
Get-ScheduledJob cmdlet, 267
Get-Service cmdlet, 77
  inputs and outputs, 113-114
  parameter binding, 114-116
  119-121
  pipelines example, 113-118
Get-SPManagedAccount cmdlet, 458
Get-SPSite cmdlet, 488
Get-SPServiceApplication cmdlet, 458
Get-SPServiceInstance cmdlet, 458
Get-SPSite cmdlet, 462, 464
Get-SPWeb cmdlet, 465
Get-SPWebApplication cmdlet, 464
Get-SPWebTemplate cmdlet, 460
Get-Verb cmdlet, 49, 76
Get-WindowsTemplate cmdlet, 36
Get-WMIObject cmdlet, 360
remote access, 302-303
WQL and, 301-302
Install-WindowsFeature cmdlet, 40, 72, 240, 419
instances in WMI, 296
Integrated Scripting Environment (ISE). See ISE (Integrated Scripting Environment)
IntelliSense code completion, 16, 103-104
interactive table, viewing output as, 179-180
interface controls for PSWA (Windows PowerShell Web Access), 247
interoperability in Microsoft, 360
Introducing Windows Azure for IT Professionals (Tulloch), 471
invocation operator (&)
running external commands, 84-85
running scripts, 390
in SharePoint Server management shell, 454
Invoke-Command cmdlet, 214-215, 224, 226, 390
Invoke-Expression cmdlet, 390
Invoke-GPUpdate cmdlet, 433
Invoke-Item cmdlet, 84
Invoke-Sqlcmd cmdlet, 443, 446
Invoke-WMIMethod cmdlet, 304
ISE (Integrated Scripting Environment), 15-17. See also scripts
add-ons, 384, 391
advantages of, 16
code folding, 387
color coding, 387
customizing, 40-42
disadvantages of, 16-17
installing, 40
one-to-many remoting, 228-231
starting, 18
verifying installation, 36
working with object members, 102-104
ISESteroids, 17
IsRegistered property for
Get-PackageSource cmdlet, 336
IsValidated property for
Get-PackageSource cmdlet, 336
items in PSProviders, 157-158

J

JobName parameter for running jobs, 265
See also workflows
cmdlets for, 255-256
checking job status, 256-257
deleting jobs from queue, 260-261
resuming jobs, 260
scheduling jobs, 266-270
starting jobs, 256
stopping jobs, 259
viewing job output, 257-259
with workflows, 287
parent and child jobs, 261-262
running remotely, 263-265
scheduling, 266-270
adding triggers, 268-269
creating scheduled jobs, 266-268
usage example, 270-272
viewing job output, 269-270
for WMI remote access, 303-304
workflows as, 282
joining domains, 418-419
Jones, Don, 119, 121, 410

K

Keep parameter for Receive-Job cmdlet, 258, 269
keyboard shortcuts
Ctrl+Break (stop script), 42
F5 (run script), 42
F7 (previous command buffer), 35
F8 (run selection), 42
Tab (cycle through parameters), 59
keywords
foreach, 280
function, 384
param, 386
kill method, 82, 111
killing processes, 110-112
Kleene, Stephen Cole, 316

L

Last parameter for Select-Object cmdlet, 139
Layout tab (Properties sheet), 34-35
LCM (Local Configuration Manager), 358, 366-368
LDAP (Lightweight Directory Access Protocol), 423
-le comparison operator, 142
less than (<), 302
libraries, viewing, 465-466
Liebnitz, Gottfried Wilhelm, 11
Lightweight Directory Access Protocol (LDAP), 423
-like comparison operator, 142
line breaks, 128, 224, 406
line numbers, viewing, 41
Linux
as file-based operating system, 91-92
pipelines, PowerShell pipelines versus, 109-110
list view, 183-185
ListAvailable parameter for Get-Module cmdlet, 70
listings
collection describing metadata, 97
properties available with Get-Process, 100
lists
creating, 466-467
viewing, 465-466
loading
modules, 71-72, 402-403
resources (DSC), 362-363
Local Configuration Manager (LCM), 358, 366-368
LocalMachine scope, 379
Log resource (DSC), 361
LogName parameter for Get-EventLog cmdlet, 56
loops in functions, 386
ls alias, 136, 156
-lt comparison operator, 141
MaxResultCount parameter for Register-ScheduledJob cmdlet, 270
McIlroy, Douglas, 113
match() method for regular expressions, 317-323
Match parameter in regular expressions, 317-323
$matches variable, 318
Microsoft, interoperability in, 360
Microsoft Azure, 356. See also Office 365; SharePoint Online
management portal for Microsoft Azure, 473
management shell for SharePoint Online, 487 for SharePoint Server, 454-455
manifest modules, 402
manifests creating, 407-409 defined, 407 troubleshooting, 409-410
Mann, Steven, 457
Mastering Regular Expressions (Friedl), 323
match() method for regular expressions, 327
Match parameter in regular expressions, 317-323
Microsoft SQL Server 2012 Unleashed (Rankins), 437
MMS (Managed Metadata Service) service application, deploying, 457-459
Module parameter for Get-Command cmdlet, 79
modules, 70-73
comment-based help, adding, 410-413
MOF (Managed Object Format) files, 294-295, 364-366
Monad Manifesto, 10-12, 355
More function, 51
Move-ADDirectoryServerOperations-MasterRole cmdlet, 431
MSI (Microsoft Software Installer), 335
MSPSGallery, 336
multipane interface, 16
multiple criteria, sorting output on, 136
Name parameter for Find-Package cmdlet, 338
for Get-Job cmdlet, 257
for Out-Printer cmdlet, 189
called, 304-305
defined, 296
calling, 304-305
def defined, 99-100
of Get-Process cmdlet, 101
in WMI

Microsoft Office TechCenter, 466
Microsoft Online Services Sign-in Assistant for IT Professionals, 485
Microsoft PowerShiell Gallery, 414
Microsoft Script Center, 387
Microsoft Software Installer (MSI), 335
namespaces in WMI, 503
N
Name parameter
for Find-Package cmdlet, 338
for Get-Job cmdlet, 257
for Out-Printer cmdlet, 189
named parameters for cmdlets, 58, 135-136 for workflows, 280
Namespace parameter for Get-WMIObject cmdlet, 300
358, 366-368
locating. See finding
Locating property for Get-Job cmdlet, 257
Log resource (DSC), 361
logging on to PSWA connection, 244-247
LogName parameter for Get-EventLog cmdlet, 56
loops in functions, 386
ls alias, 136, 156
-lt comparison operator, 141
man command, 50
managed accounts, creating for SharePoint Server, 455-456
Managed Metadata Service (MMS) service application, deploying, 457-459
Managed Object Format (MOF) files, 294-295
creating, 364-366
DSC push/pull models, 358-359
management portal for Microsoft Azure, 473
management shell for SharePoint Online, 487 for SharePoint Server, 454-455
manifest modules, 402
manifests creating, 407-409 defined, 407 troubleshooting, 409-410
Mann, Steven, 457
Mastering Regular Expressions (Friedl), 323
match() method for regular expressions, 327
Match parameter in regular expressions, 317-323
$matches variable, 318
Microsoft SQL Server 2012 Unleashed (Rankins), 437
MMS (Managed Metadata Service) service application, deploying, 457-459
Module parameter for Get-Command cmdlet, 79
modules, 70-73
comment-based help, adding, 410-413
MOF (Managed Object Format) files, 294-295, 364-366
Monad Manifesto, 10-12, 355
More function, 51
Move-ADDirectoryServerOperations-MasterRole cmdlet, 431
MSI (Microsoft Software Installer), 335
MSPSGallery, 336
multipane interface, 16
multiple criteria, sorting output on, 136
Name parameter
for Find-Package cmdlet, 338
for Get-Job cmdlet, 257
for Out-Printer cmdlet, 189
named parameters for cmdlets, 58, 135-136 for workflows, 280
Namespace parameter for Get-WMIObject cmdlet, 300
namespaces in WMI, 503
naming conventions

cmdlets, 8-9
workflows, 280
-ne comparison operator, 141
.NET classes, documentation, 98
.NET Framework
derived, 13
determining version of, 26-27
installing, 27
required versions, 26
New-ADGroup cmdlet, 424
New-ADOrganizationalUnit cmdlet, 423
New-ADUser cmdlet, 125-128, 425
New-Alias cmdlet, 64-65
New-AzureSqlDatabase cmdlet, 483
New-AzureSqlDatabaseServer cmdlet, 481
New-ISESnippet cmdlet, 384
New-Item cmdlet, 163-164, 317, 380
New-JobTrigger cmdlet, 268
New-ModuleManifest cmdlet, 407
New-PSSession cmdlet, 211
New-PSSessionConfigurationFile cmdlet, 227
New-PSSessionOption cmdlet, 248
New-ScheduledJobOption cmdlet, 269
New-SPAuthenticationProvider cmdlet, 459
New-SPMetadataService-Application cmdlet, 458
New-SPWebApplication cmdlet, 459
New-WebBinding cmdlet, 241
Ninite, 350
Noel, Michael, 456
NoExit parameter for PowerShell.exe console, 454
-notcontains comparison operator, 142
-notin comparison operator, 142
-notlike comparison operator, 142
NotMatch parameter in regular expressions, 318
nouns for cmdlets, 49-50
NuGet, 332-333, 336

object-oriented programming (OOP), 94-95

objects
class diagram, explained, 94-95
creating, 99
defined, 93-94
finding members, 96-102
.Get-Member cmdlet, 100-102
.NET classes in, 98
properties/methods, 99-100
measuring, 144-148
methods, 99-100
properties, 99-100
referencing, 94-96
working with, 102-104
octothorpe (#), 129, 393
Office 365, 472
collecting, 485-487
pricing, 485
offline systems, updating help on, 53-54
Olson, Kevin, 393
OneGet
cmdlets for, 333
creating private repositories, 349-350
defined, 333-334
Install-Package cmdlet, 340-341
Ninite versus, 350
removing packages, 349
software installation locations, 341-343
subdirectory installation, 343-344
system requirements, 334
upgrading packages, 347-348
usage example, 344-346

one-to-many remoting

ComputerName parameter, 221-222
implicit remoting, 232-233
passing arguments to cmdlets, 231-232
persistent sessions, 224-225
session configuration management, 225-228
with Windows PowerShell ISE, 228-231

one-to-one remoting

alternate credentials for, 212-214
ComputerName parameter, 201-203
creating remote session, 209-214
enabling, 205-209
Enable-PSRemoting cmdlet, 205-207
with Group Policy, 207-208
in workgroups, 208-209
explained, 203-205
sending scripts over network, 214-215
storing sessions in variables, 212-214
usage example, 215-217

Online parameter for help system, 52
on-premises environment, 485
OOP (object-oriented programming), 94-95
Options tab (Properties sheet), 32-33
OUs (organizational units), creating, 423-424
Out-Default cmdlet, 133, 176-177, 180
Out-File cmdlet, 117-118, 187-189
Out-GridView cmdlet, 178-180, 339
Out-Host cmdlet, 133, 176
Out-Printer cmdlet, 189
outputs
cmdlets for, 133
converting output, 193-195
exporting output, 187-193
filtering output, 137-144
formatting output, 176-187
measuring objects, 144-148
sorting output, 134-137
for Get-Process cmdlet, 112
for Get-Service/Stop-Service cmdlets, 113-114
properties

redirecting, 142
truncation, 188-189

P

package managers, 332
package providers, viewing list of, 334-336
package repositories. See repositories
Package resource (DSC), 361
packages
defined, 332
for DSC, 338
finding in Chocolatey, 338
installing
from command line, 340-341
installation location, 341-343
from subdirectory, 343-344
usage example, 344-346
private repositories, creating, 349-350
removing, 349
upgrading, 347-348
parallel workflows, creating, 283
param keyword, 386
parameter binding, 114-116, 118-119
manual property mapping, 125-129
by property name, 119-122
usage example, 122-125
by value, 119
parameter sets for cmdlets, 55-56
parameters
for cmdlets, 49-50. See also names of specific parameters
attribute table, 56-57
attribute types, 118
cycling through, 59
named parameters, 58
positional parameters, 57-58
required parameters, 57
in functions, 384
parent jobs, 261-262
parentheses (()), 165
passing arguments remotely, 231-232
passing data through pipelines. See parameter binding
passwords, resetting in Active Directory, 427
PATH environment variable, 84
Path parameter
for Get-ChildItem cmdlet, 161
for New-Item cmdlet, 163
for Select-String cmdlet, 324
Pattern parameter for Select-String cmdlet, 324
PDF (Portable Document Format), 308
percent sign (%), 302
period (.), 320
permissions on site collections, 462-463
persistent sessions, 224-225
Pii (personally identifiable information), finding with regular expressions, 325-327
pinning shortcut icons to taskbar, 35
pipe (|), 20, 113, 128
pipelines. See also outputs
case study, 110-112
data types in, 113-114
Get-Service/Stop-Service cmdlets example, 113-118
origin of pipe (|) usage, 113
parameter binding, 114-116, 118-119
manual property mapping, 125-129
by property name, 119-122
usage example, 122-125
by value, 119
streams, 395
types of, 112
Windows/Linux pipelines versus, 109-110
PL-SQL, 440
Portable Document Format (PDF), 308
PoSH acronym, 334
positional parameters for cmdlets, 57-58
positioning
Console pane, 40-41
Script pane, 40-41
pound sign (#). See octothorpe (#)
PowerShell. See Windows
PowerShell
Powershell Code Repository, 387
PowerShell for SharePoint 2013 How-To (Mann), 457
PowerShellExe, switching between Cmd.exe and, 19
PowerShellGet, 413-414. See also OneGet
PowerShellVersion property for New-ModuleManifest cmdlet, 408
PowerShellWebAccess module, cmdlets in, 240
prefixes for cmdlets, 48, 79
premises, 485
printers, outputting to, 189
private repositories, creating, 349-350
Process Explorer, 346
Process scope, 379
processes, killing, 110-112. See also Get-Process cmdlet
production environment for DSC, 358-359
$profile automatic variable, 380
profile scripts
creating, 380-381
tips for, 381-382
verifying presence of, 379-380
ProGet, 350
programming logic in scripts, 385-387
promoting domain controllers, 421-422
Prompt parameter for Read-Host cmdlet, 389
properties. See also parameter binding
defined, 93, 99-100
finding names, 135-136
of Get-Process cmdlet, 100
manual property mapping, 125-129
in WMI, 296
Properties sheet (Control menu), 32
Colors tab, 35
Font tab, 33-34
Layout tab, 34-35
Options tab, 32-33
property name, parameter binding by, 119-122
Property parameter for Format-List cmdlet, 184
for Format-Table cmdlet, 182
for Measure-Object cmdlet, 146
for Sort-Object cmdlet, 134
ProtectedFromAccidentalDeletion parameter for
New-ADOrganizationalUnit cmdlet, 424
ProviderName property for
Get-Package cmdlet, 348
providers
Active Directory provider, 169-170
Alias provider, 166-167
defined, 153-154
FileSystem provider, 159-165
Get-ChildItem cmdlet and, 160-162
usage example, 162-165
PSDrives, 155-156
items, 157-158
navigation system for, 156-157
PSProviders
for external applications, 154
items, 157-158
list of, 153-154
purpose of, 154-155
Registry provider, 167-169
editing values, 169
retrieving values, 167-169
SQL Server provider, 170
ps alias, 80
PSComputerName property, 223
PSDrives, 155-156
items, 157-158
navigation system for, 156-157
PSGallery, 336
PsisContainer property of
DirectoryInfo object, 163
PSJobTypeName property for
Get-Job cmdlet, 257, 262
PSModule, 335
$PSModulePath environment variable, 70, 404
PSPersist parameter in workflows, 284-285
PSProviders
for external applications, 154
items, 157-158
list of, 153-154
PSReadLine, 17
PSWA (Windows PowerShell Web Access), 237-239
configuring, 240-243
defining authorization rules, 243-244
disconnected sessions, 247-248
installing, 239-240
interface controls, 247
managing, 249-250
testing connection, 244-247
usage example, 250-251
public clouds, 472
Public network adapters, enabling remote access, 205-207
pull model (DSC), 359, 369-373
Puppet, 356
push model (DSC), 358
PuTTY, 343-344
Quick Edit mode, 38
quotation marks (" ")
in directory path, 84
double quotes versus single quotes, 385
in regular expressions, 321
R
ranges in regular expressions, 318-320
Rankins, Ray, 437
RDP (Remote Desktop Protocol), 479
Read-Host cmdlet, 389
Receive-Job cmdlet, 257-259, 269
Recurse parameter for Remove-Item cmdlet, 165
redirecting
output, 142, 187
streams, 395
refactoring, 392
referencing objects, 94-96
RefreshFrequencyMins parameter for
Get-DscLocalConfigurationManager cmdlet, 367
RegExBuddy, 325
RegExes. See regular expressions
RegExLib.com, 326
Registered parameter for Get-PSSnapin cmdlet, 70
Register-PackageSource cmdlet, 346
Register-PSSessionConfiguration cmdlet, 227, 381
Register-ScheduledJob cmdlet, 266
Registry provider, 167-169
editing values, 169
retrieving values, 167-169
Registery resource (DSC), 361
regular expressions
anchors and ranges, 318-320
business logic and, 323
classifiers, 321-323
defined, 315-316
with Match parameter, 317-323
qualifiers, 320-321
quotation marks, 321
Select-String cmdlet, 324-325
test environment setup, 316-317
Qualifiers in regular expressions, 320-321
Query parameter
for Get-WMIObject cmdlet, 301
for Invoke-Sqlcmd cmdlet, 446
Query PowerGUI, 18
question mark (?)
as alias, 138-139
as wildcard character
in regular expressions, 320
in searches, 315

Q

ranges in regular expressions, 318-320
Rankins, Ray, 437
RDP (Remote Desktop Protocol), 479
Read-Host cmdlet, 389
Receive-Job cmdlet, 257-259, 269
Recurse parameter for Remove-Item cmdlet, 165
redirecting
output, 142, 187
streams, 395
refactoring, 392
referencing objects, 94-96
RefreshFrequencyMins parameter for
Get-DscLocalConfigurationManager cmdlet, 367
RegExBuddy, 325
RegExes. See regular expressions
RegExLib.com, 326
Registered parameter for Get-PSSnapin cmdlet, 70
Register-PackageSource cmdlet, 346
Register-PSSessionConfiguration cmdlet, 227, 381
Register-ScheduledJob cmdlet, 266
Registry provider, 167-169
editing values, 169
retrieving values, 167-169
Registery resource (DSC), 361
regular expressions
anchors and ranges, 318-320
business logic and, 323
classifiers, 321-323
defined, 315-316
with Match parameter, 317-323
qualifiers, 320-321
quotation marks, 321
Select-String cmdlet, 324-325
test environment setup, 316-317

Q

qualifiers in regular expressions, 320-321
Query parameter
for Get-WMIObject cmdlet, 301
for Invoke-Sqlcmd cmdlet, 446

RegExLib.com, 326
Registered parameter for Get-PSSnapin cmdlet, 70
Register-PackageSource cmdlet, 346
Register-PSSessionConfiguration cmdlet, 227, 381
Register-ScheduledJob cmdlet, 266
Registry provider, 167-169
ingitor values, 169
retrieving values, 167-169

Quest PowerGUI, 18
type accelerator, 327
usage example, 325-327
when to use, 316
remote access. See also PSWA
(Windows PowerShell Web Access)
in CIM, 305
one-to-many remoting
ComputerName parameter, 221-224
implicit remoting, 232-233
passing arguments to cmdlets, 231-232
persistent sessions, 224-225
session configuration management, 225-228
with Windows PowerShell ISE, 228-231
one-to-one remoting
alternate credentials for, 212-214
ComputerName parameter, 201-203
creating remote session, 209-214
enabling, 205-209
explained, 203-205
sending scripts over network, 214-215
storing sessions in variables, 212-214
usage example, 215-217
profile scripts and, 381-382
running jobs, 263-265
in WMI, 302-304
Remote Desktop Protocol (RDP), 479
Remote Server Administration Tools (RSAT), installing, 73-74
RemoteSigned execution policy, 378
Remove-Computer cmdlet, 418
Remove-Item cmdlet, 165
Remove-Job cmdlet, 260-261
Remove-Module cmdlet, 401
Remove-PSSession cmdlet, 211
Remove-PSSnapin cmdlet, 70, 401
Remove-PswaAuthorizationRule cmdlet, 249
removing. See deleting
Rename-Computer cmdlet, 289
REPL (Read, Evaluate, Print, Loop) interface, 13-14
repositories
creating private, 349-350
defined, 333
documentation in, 339-340
trusting, 337, 346-347
viewing, 336-340
required parameters for cmdlets, 57
resetting passwords in Active Directory, 427
resources (DSC), 359
list of, 361-362
loading, 362-363
waves, 363
Restart-Computer cmdlet, 289, 305
Restricted execution policy, 378
Resume-Job cmdlet, 260
Resume-Workflow cmdlet, 284-287
resuming
jobs, 260
workflows, 285-287
Retrieve, as unapproved verb, 406
retrieving
event logs, 19-21
Registry values, 167-169
root site collections, 460
RootModule property for New-ModuleManifest cmdlet, 408
RSAT (Remote Server Administration Tools), installing, 73-74
RSAT for Windows 7 SP1, 74
RSAT for Windows 8, 74
RSAT for Windows 8.1, 74
rule files for formatting subsystem, 178-179
RuleName parameter for Get-PswaAuthorizationRule cmdlet, 244
running
commands in modules, 406
external commands, 83
call operator (&), 84-85
from environment path, 84
Invoke-Item cmdlet, 84
stop parsing symbol (~%), 85-86
jobs remotely, 263-265
scripts, 41-42, 390-391
SELECT queries against tables, 447
Windows PowerShell as administrator, 15
with SQL Server tools, 438-442
workflows as jobs, 282
Russinovich, Mark, 471
S
\S character class, 322
\s character class, 322
Santos, Donabel, 445, 447
Sapien Powershell Studio, 18
Save button (PSWA), 247
Save-Help cmdlet, 53
scheduling
database backups, 447-449
jobs, 266-270
adding triggers, 268-269
creating scheduled jobs, 266-268
usage example, 270-272
viewing job output, 269-270
schemas in WMI, 296
scope
of execution policies, 379
of functions, 395
Scope parameter for Set-ExecutionPolicy cmdlet, 379
screen buffer size, changing, 34
Script Browser ISE add-in, 387, 391
script modules, 402-407
Script pane
positioning, 40-41
zooming, 41
Script resource (DSC), 361
Scriptblock parameter for Invoke-Command cmdlet, 214
scripts. See also configuration scripts; ISE (Integrated Scripting Environment); one-to-many remoting; workflows
creating, 276-279, 388-390
default file association, 377
editing, 391-394
execution policies
changing, 103-104, 276
list of, 378
scope, 379
viewing and setting, 378-379
functions
creating, 276-278, 383-385
defined, 47, 383
double quotes versus single quotes, 385
Help, 50-52
More, 51
scope, 395
snippets, 383-384
pipeline streams, 395
profile scripts
creating, 380-381
tips for, 381-382
verifying presence of, 379-380
programming logic in, 385-387
running, 41-42, 390-391
sending over network, 214-215
sharing, 387
SQL Server scripts, creating, 449
stopping, 42
tips for, 395
workflows versus, 276
searching. See regular expressions; wildcards
Secure Sockets Layer (SSL) certificates, 241
security
Microsoft Azure, 474
script execution policies, 379
trusting repositories, 337, 346-347
select alias, 138
SELECT queries, running against tables, 447
Select-AzureSubscription cmdlet, 480
Select-Object cmdlet, 138-141
Select-String cmdlet, 324-325
semicolon (;), 73
Sequence activity, 284, 287-288
Server Management Objects (SMO), 438
service accounts, creating for SharePoint Server, 455-456
service applications, deploying, 457-459
Service resource (DSC), 362
session configuration management, 225-228
Session parameter without ComputerName parameter, 222
Set-ADAccountPassword cmdlet, 427
Set-ADDomainMode cmdlet, 432
Set-ADForestMode cmdlet, 432
Set-CimInstance cmdlet, 307
Set-ExecutionPolicy cmdlet, 103, 276, 378-379
Set-Location cmdlet, 136, 155, 157
Set-PSSessionConfiguration cmdlet, 227
Set-WSManQuickConfig cmdlet, 205
7-Zip, installing, 340-341
shared files, creating private repositories, 350
SharePoint 2013 Unleashed (Noel), 456
SharePoint Online, 472
connecting to, 487-488
pricing, 485
SharePoint Server
architecture of, 453-454
cmdlets for, 455, 463-464
libraries, viewing, 465-466
lists
creating, 466-467
viewing, 465-466
managed accounts, creating, 455-456
management shell, 454-455
service applications, deploying, 457-459
site collections
deploying, 460-462
permissions on, 462-463
viewing, 464-465
sites, viewing, 465
web applications
deploying, 459-460
viewing, 464
sharepoint.ps1 startup script, 454-455
sharing scripts, 387
shell scripts, 10
shells, 10
shortcut icons, pinning to taskbar, 35
ShouldProcess capability, 155
Show-Command cmdlet, 62-64
ShowSecurityDescriptorUI parameter for Set-PSSessionConfiguration cmdlet, 227
ShowWindow parameter for help system, 53, 61-62, 72
Simple Network Management Protocol (SNMP), 294
single quotes, double quotes versus, 385
site collections
deploying, 460-462
permissions on, 462-463
viewing, 464-465
sites, viewing, 465
SkipNetworkProfileCheck parameter for Enable-PSRemoting cmdlet, 206
sl alias, 157
SMO (Server Management Objects), 438
snap-ins, 69-70, 399-400
automating, 401
installing, 400-401
for SharePoint Server, 454-455
viewing list of, 400
snippets, 16, 276-278, 383-384
SNMP (Simple Network Management Protocol), 294
Snover, Jeffrey, 10, 76, 144, 355, 360
software management. See also OneGet
creating private repositories, 349-350
installation from command line, 340-341
installation location, 341-343
package providers, viewing list of, 334-336
removing packages, 349
repositories, viewing, 336-340
subdirectory installation, 343-344
terminology, 332-334
trusting repositories, 346-347
upgrading packages, 347-348
usage example, 344-346

Sort alias, 134
sorting outputs, 134-137
finding property names, 135-136
grouping output, 136-137
on multiple criteria, 136
Sort-Object cmdlet, 134

Source parameter for Get-Package cmdlet, 342
sp_configure, 440
splat operator. See asterisk (*) wildcard character
SQL (Structured Query Language), 137-138, 358, 440
SQL Server
databases
creating, 445, 481-484
listing in instances, 443-444
scheduling backups, 447-449
management tools, installing, 438-442
running PowerShell with, 438-442
scripts, creating, 449
sqlps module, 442-443
tables
creating, 445-446
inserting data, 446-447
running SELECT queries, 447
SQL Server 2012 with PowerShell V3 Cookbook (Santos), 445, 447
SQL Server Management Studio (SSMS), 438
SQL Server provider, 170
sqlps module, 442-443
sqlps utility, 441-442
SSL (Secure Sockets Layer) certificates, 241
SSMS (SQL Server Management Studio), 438
Stardock Start8, 342
Start-AzureVM cmdlet, 481
Start-DscConfiguration cmdlet, 371
starting
console, 18
ISE (Integrated Scripting Environment), 18
jobs, 256
VMs (virtual machines), 481-484
Windows PowerShell, 14
Start-Job cmdlet, 256
Start-SPAssignment cmdlet, 465
Start-SPServiceInstance cmdlet, 458
StartupScript parameter for Register-PSSessionConfiguration cmdlet, 381
Start-Website cmdlet, 372
State property
for Get-Job cmdlet, 257
for Remove-Job cmdlet, 260-261
stop parsing symbol (–%), 85-86
Stop-AzureVM cmdlet, 481
Stop-Computer cmdlet, 305
Stop-Job cmdlet, 259
stopping
jobs, 259
scripts, 42
VMs (virtual machines), 479, 481-484
Stop-Process cmdlet, 111
Stop-Service cmdlet
inputs and outputs, 113-114
parameter binding, 114-116, 119-121
pipelines example, 113-118
Stop-SPAssignment cmdlet, 465
storing remote sessions in variables, 212-214
streams, 395
Structured Query Language (SQL), 137-138, 358, 440
subdirectories, installing from, 343-344
Submit button (PSWA), 247
subscriptions
Office 365
connecting to, 485-487
pricing, 485
SharePoint Online
connecting to, 487-488
pricing, 485
Surname parameter for New-ADUser cmdlet, 126
suspending workflows, 285-287
Suspend-Job cmdlet, 260
Suspend-Workflow cmdlet, 284-287
Svidergol, Brian, 423
switch construction, 389
switch parameters for cmdlets, 50
Sysinternals tools, 345-346
system configuration, workflows for, 285
system requirements
for OneGet, 334
for workflows, 276
Tab (cycle through parameters)
keyboard shortcut, 59
Tab Complete button (PSWA), 247
tab completion, 82-83, 138, 301, 305
table view, 180-183
tables
creating, 445-446
inserting data, 446-447
running SELECT queries, 447
taskbar, pinning shortcut icons to, 35
TechNet library, 52
TechNet Script Gallery, 387
templates for site collections, 460-462
Test-ADDSDomainController-Installation cmdlet, 421
Test-Connection cmdlet, 83, 280
testing
domains for membership, 418
PSWA connection, 244-247
Test-Manifest cmdlet, 409
Test-PswaAuthorizationRule cmdlet, 249
text files, outputting to, 187-189
ThrottleLimit parameter for
Invoke-Command cmdlet, 226
tilde (~), 314
top-level site collections, 460
trademark issues for Windows
PowerShell, 36
Transactions capability, 155
Transact-SQL (T-SQL), 440
transferring FSMOs (Flexible
Single Master Roles), 431-432
triggers, adding to scheduled jobs,
268-269
troubleshooting manifests,
409-410
truncation of output, 188-189
TrustedHosts list, enabling remote
access, 208-209
trusting repositories, 337, 346-347
T-SQL (Transact-SQL), 440
Tulloch, Mitch, 471
turning on/off
Commands add-on, 40
module autoloading, 403
type accelerator for regular
expressions, 327
Type parameter
in formatting subsystem,
177-178
for New-Item cmdlet, 164

U
unapproved verbs
in modules, 407
Retrieve as, 406
in SQL Server, 443-444
Undefined execution policy, 378
unhiding extensions, 226
uniform resource locators (URLs),
finding, 476
Uninstall-Package cmdlet, 349
Uninstall-PswaWebApplication
cmdlet, 250
UNIX, as file-based operating sys-
tem, 91-92
unloading modules, 402-403
Unlock-ADAccount cmdlet, 426
Unrestricted execution policy, 378
Update() method, 465
updateable, help system as,
51-52
Update-DscConfiguration cmdlet,
372
Update-Help cmdlet, 52
updating
help system on offline sys-
tems, 53-54
local help library, 52
upgrading packages, 347-348
URLs (uniform resource locators),
finding, 476
user accounts
in Active Directory
creating, 425-426
creating in bulk, 427-430
managing, 426-427
for SharePoint Server, creat-
ing, 455-456
User resource (DSC), 362
UserGroupName parameter for
Get-PswaAuthorizationRule cmd-
let, 244
UseTestCertificate parameter for
Install-PSWAWebApplication
cmdlet, 240
Verbose parameter for Install-
Package cmdlet, 340
verbose pipeline, 112
verbs
for cmdlets, 49, 75-76
unapproved verbs
in modules, 407
Retrieve as, 406
in SQL Server, 443-444

V
value, parameter binding by, 119
Variable PSDrive, aliases for,
156-157
variables, storing remote sessions
in, 212-214
VBScript (Visual Basic Scripting
Edition), 10
virtual machines (VMs). See VMs
(virtual machines)
Visual Basic Scripting Edition
(VBScript), 10
VMs (virtual machines)
creating, 477-479
managing, 479-481
starting, 481-484
stopping, 479, 481-484

W
\W character class, 322
\w character class, 322
Wait parameter for
Restart-Computer cmdlet, 289
warning pipeline, 112
waves, resources (DSC), 363
WBEM (Web-Based Enterprise
Management), 294
web access. See PSWA (Windows
PowerShell Web Access)
web applications
deploying, 459-460
viewing, 464
Web Services-Management
(WS-Man) protocol, 203-205
WebApplicationName parameter
for Install-PSWAWebApplication
cmdlet, 240
Web-Based Enterprise
Management (WBEM), 294
WebSiteName parameter for
Install-PSWAWebApplication
cmdlet, 241
WF (Windows Workflow
Foundation), 276
WhatIf parameter for
Install-PSWAWebApplication
cmdlet, 241
where alias, 139
Where-Object cmdlet, 139-141,
317
Width parameter for Out-File cmd-
let, 188
wildcards. See also asterisk (*)
wildcard character; question
mark (?)
in help system, 57
syntax for, 313-315
Wilson, Ed, 395
Win32_BIOS class, 298
Win32_DesktopMonitor class, 300
Win32_LogicalDisk instance,
297-299
Win32_NetworkAdapter class,
300
Win32_OperatingSystem class,
300
Win32_PhysicalMemory class,
298
Win32_Process class, 300
Win32_Product class, 300
Win32_Service class, 300
Win32_StartupCommand class,
300
window position, changing, 34
window size, changing, 34
Windows
determining version of, 28
required versions, 27-28
Windows Azure Active Directory
Module for Windows PowerShell,
485
Windows Management
Framework (WMF), 30
components, 30
installing, 30-31
Windows Management
Instrumentation Query Language
(WQL), 137, 301-302
Windows Management
Instrumentation (WMI). See
WMI (Windows Management
Instrumentation)
Windows pipelines, PowerShell
pipelines versus, 109-110
Windows PowerShell
components
console, 13-14. See also
core
hosts, 17-18
ISE, 15-17. See also ISE
(Integrated Scripting
Environment)
defined, 12-13
determining version of, 28
history of, 10-12
installing latest version, 29-31
.NET Framework
determining version of, 26-27
required versions, 26
PoSH acronym, 334
reasons for learning, 8-10
running as administrator, 15
running with SQL Server tools, 438-442
shortcut icon, pinning to task-
bar, 35
starting, 14
switching between command
prompt session and, 19
trademark issues, 36
Windows
determining version of, 28
required versions, 27-28
"Windows PowerShell: Comment
Your Way to Help" (Jones), 410
Windows PowerShell Web Access
(PSWA). See PSWA (Windows
PowerShell Web Access)
Windows Remote Management
(WinRM), 204-205
Windows Script Host (WSH), 10
Windows Server 2012 R2, 17
Windows Server 2012 Unleashed
(Morimoto), 418
Windows Task Scheduler, 266
Windows Workflow Foundation
(WF), 276
WindowsFeature resource (DSC),
362
WindowsProcess resource (DSC),
362
WinRAR, 338, 342
WinRM (Windows Remote
Management), 204-205
WMF (Windows Management
Framework), 30
components, 30
installing, 30-31
WMI (Windows Management
Instrumentation)
browser tools, 296-299
classes, list of, 298-300
cmdlets for, 300
calling methods, 304-305
Get-WMIObject, 300-301
remote access, 302-304
WQL (Windows
Management
Instrumentation Query
Language), 301-302
defined, 293-294
drawbacks of, 293
location, 294-295
terminology, 296
WMI Client (WMI), 296
WMI Explorer 2.0 (Codeplex.com), 297
WMI Explorer 2014 (Sapien Technologies), 297
WMI Query Language (WQL), 299
WMIC (WMI Client), 296
workflows. See also jobs
activities, 283-284
CheckPoint-Workflow cmdlet, 284-285
Resume-Workflow cmdlet, 284-287
Sequence, 284, 287-288
Suspend-Workflow cmdlet, 284-287
checkpoints in, 284-285
creating, 280-281
defined, 47, 266, 276	naming conventions, 280
parallel workflows, creating, 283
resuming, 285-287
running as jobs, 282
scripts versus, 276
suspending, 285-287
for system configuration, 285
system requirements, 276
usage example, 288-289
workgroups, enabling remote access, 208-209
WorkingSet (WS) property for Get-Process cmdlet, 141
WQL (Windows Management Instrumentation Query Language), 137, 301-302
WQL (WMI Query Language), 299
Wrap parameter for Format-Table cmdlet, 181
Write-Host cmdlet, 38, 385
Write-Output cmdlet, 38, 385
WS (WorkingSet) property for Get-Process cmdlet, 141
WSH (Windows Script Host), 10

X

XAML (Extensible Application Markup Language), 283
XML (Extensible Markup Language), outputting to, 191-195
xp_cmdshell, 439-441
XPS (XML Paper Specification), 308
XPS Document Writer, 308

Z

zooming
Console pane, 41
Script pane, 41