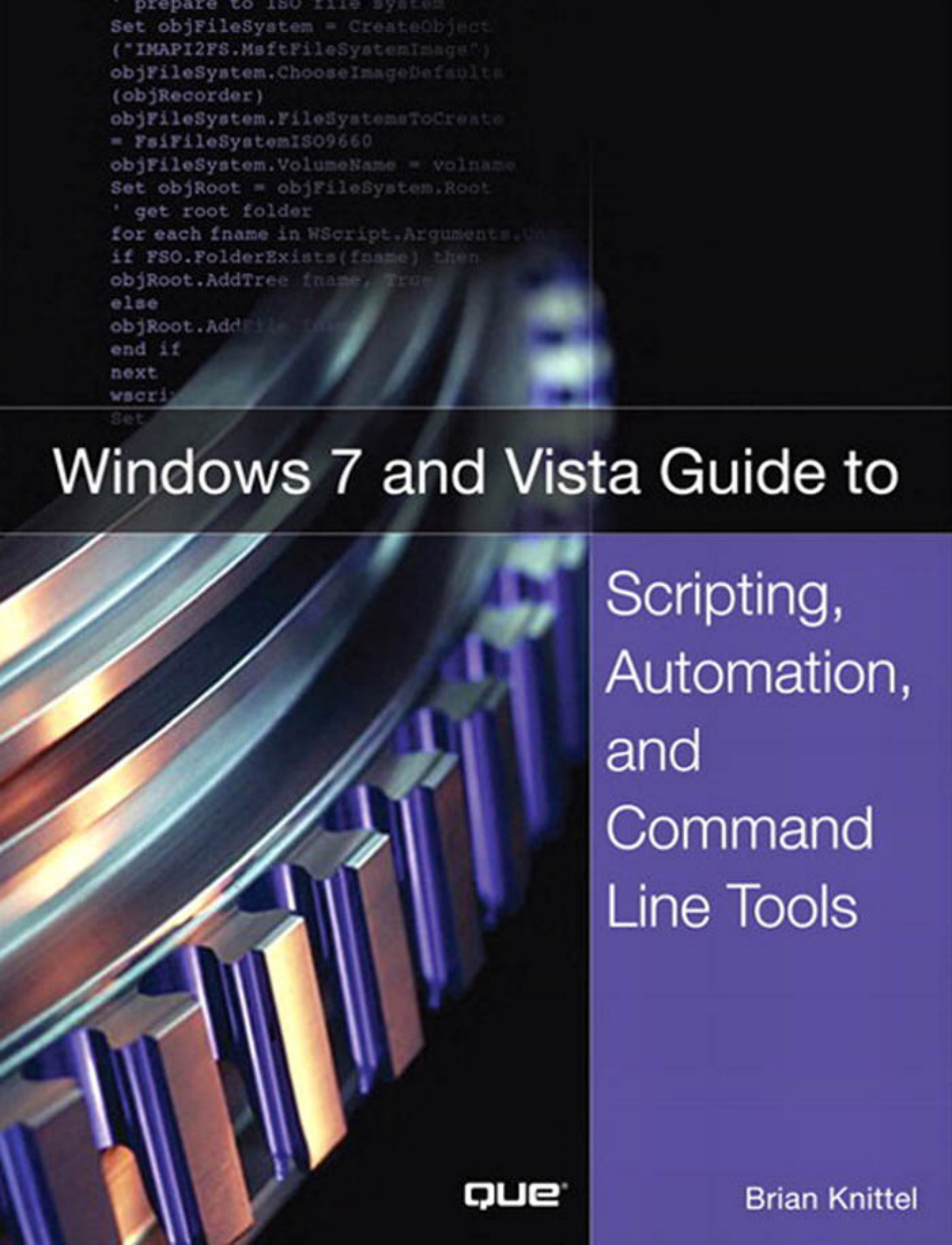
Windows 7 and Vista Guide to
Scripting, Automation, and Command Line Tools
que
Brian Knittel

Windows 7 and Vista: Guide to Scripting, Automation, and Command Line Tools

ISBN-13: 978-0-7897-3728-1
ISBN-10: 0-7897-3728-0

Library of Congress Cataloging-in-Publication data is on file.

Printed in the United States of America

First Printing: December 2010

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Que Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Bulk Sales

Que Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

U.S. Corporate and Government Sales
1-800-382-3419
corpsales@pearsontechgroup.com

For sales outside of the United States, please contact

International Sales
international@pearson.com

Associate Publisher
Greg Wiegand

Acquisitions Editor
Rick Kughen

Development Editor
Todd Brakke

Managing Editor
Sandra Schroeder

Project Editor
Mandie Frank

Copy Editor
Megan Wade

Indexer
Tim Wright

Proofreader
Jovana Shirley

Technical Editor
Ron Barrett

Publishing Coordinator
Cindy Teeters

Designer
Anne Jones

Compositor
Studio Galou, LLC

Contents at a Glance

Introduction 1

I Scripting with Windows Script Host

1 Windows Script Host 9
2 VBScript Tutorial 49
3 Scripting and Objects 93
4 File and Registry Access 123
5 Network and Printer Objects 207
6 Messaging and Faxing Objects 231
7 Windows Management Interface 279
8 Active Directory Scripting Interface 319
9 Deploying Scripts for Computer and Network Management 375

II The Command Line Environment

10 The CMD Command-Line Shell 433
11 Batch Files for Fun and Profit 491
12 The MS-DOS Environment Under Windows 521
13 Command-Line Utilities 543

III Introduction to Windows PowerShell

14 Windows PowerShell 593
15 PowerShell Programming 621
16 Using PowerShell 683

IV Appendices

A VBScript Reference 713
B CMD and Batch File Language Reference 725
C Command Line Program Reference 735
D Index of Patterns and Sample Programs 747
Index 753

Online Content:

E Automation Object Reference
F WSF and WSC File Format Reference
G Creating Your Own Scriptable Objects

Table of Contents

Introduction 1

I Scripting with Windows Script Host

1 Windows Script Host 9

What Is a Windows Script? 9
- The "Script" Part 9
- The "Windows" Part 10
- The "Host" Part 11
- How Is This Different from Writing Batch Files? 13

Scripting Languages 13
- VBScript 14
- JScript 14
- Perl 15
- Python 15
- Open Object REXX 15
- Ruby 15
- Choosing a Language 16

A Simple Script 16

Types of Script Files 19
- JSE and VBE: Encoded Scripts 20
- Windows Script Files (WSF) 21
- Windows Script Components (WSC) 23
- WSH Settings 23

Creating Your First Script File 24
- Making and Securing a Script Folder 24
- Creating a Script 26
- Script Editing Tools 27

How Windows Runs Scripts 28
- Wscript and Cscript 28
- Ways to Run a Script 29
- Passing Information to Scripts 31
- Saving the Results from Scripts 32
- Wscript and Cscript Command Options 33

Running Your Own Scripts 36
Adding Scripts to the Path 37
Running Scripts with a Shortcut Icon 38
Making a Script Shortcut 39
Running Scripts from Batch Files 39
Running Scripts Automatically 40
Security Concerns 40
Trust Policy and Script Signing 42
Debugging Scripts 42
Where to Get More Information 47

2 VBScript Tutorial 49
Introduction to VBScript 49
Variables 50
Constants 51
Named Constants 52
Operators and Expressions 53
Automatic Conversion 57
Flow Control 57
The `If…Then` Statement 58
The `Select Case` Statement 61
The `Do While` Loop 63
Terminating a Loop with `Exit Do` 65
Counting with the `For…Next` Statement 66
Processing Collections and Arrays with `For…Each` 67
VBScript Functions 68
Calling Functions and Subroutines 69
Documentation and Syntax 70
String-Manipulation Functions 71
Date and Time Functions 74
Interacting with the User 79
The MsgBox() Function 79
The InputBox() Function 82
Printing Simple Text Messages with `Wscript.Echo` 84
Advanced VBScript Topics 85
Error Handling 86
Procedures: Functions and Subroutines 87

Arrays 89
Variable Scope 91
Where to Go from Here 92

3 Scripting and Objects 93
Introduction to Objects 93
Classes and Instances 94
Containers and Collections 95
Object Naming 97
Using Objects with VBScript 98
Automation and Document Files 99
The Difference Between Properties and Methods 100
Nested Objects 101
Releasing Objects 102
Working with Collections 102
Using Objects with JScript 104
Case Sensitivity 104
Working with Collections 104
Using Objects with ActivePerl 106
Running Perl Scripts in WSH 106
The Perl Object Interface 107
Working with Collections 108
Using Objects with ActivePython 109
Working with Collections 110
Using the `WScript` Object 111
Retrieving Command-Line Arguments 113
Locating and Using Unusual Objects 115

4 File and Registry Access 123
Getting Real Work Done 123
Manipulating Files and Folders 124
`Scripting.FileSystemObject` 124
Working with File and Pathnames 130
The `Scripting.Drive` Object 135
The `Scripting.Folder` Object 139
The `Scripting.File` Object 144

Reading and Writing Files 149
The `TextStream` Object 150
Reading Text from Files 152
Writing Text to Files 154
Working with `Stdin` and `Stdout` 159
Reading Binary Files 163
Reading and Writing XML 167
Some XML Basics 168
Reading an XML File 176
Creating an XML or HTML File 179
Manipulating Programs and Shortcuts 181
The `WScript.Shell` Object 182
Running Programs 186
Creating and Modifying Shortcuts 193
Working with the Environment 196
Extracting Environment Information 198
Managing Environment Settings 199
Working with the Registry 201
Examining Registry Keys and Values 202
Saving Information in the Registry 203

5 Network and Printer Objects 207
Managing Network and Printer Connections 207
Retrieving Network User Information 212
Managing Drive Mappings 214
Listing Drive Mappings with `EnumNetworkDrives` 214
Adding Drive Mappings 218
Deleting Drive Mappings 219
Setting Up Mappings in a Script 220
Managing Network Printer Connections 221
Displaying Printer Information 222
Connecting to Network Printers 223
Redirecting DOS Session Printers 225
Deleting Printer Connections 226
Setting the Default Printer 228
Printing from Scripts 229

6 Messaging and Faxing Objects 231
Sending Email from Scripts with CDO 231
The CDO Object Model 232
The `CDO.Message` Object 235
Working with Fields 242
Fields for the `CDO.Message` Object 244
The CDO BodyParts Collection 246
The `CDO BodyPart` Object 247
The ADO `Stream` Object 250
The `CDO.Configuration` Object 250
Sending a Message with CDO 256
Constructing the Message 257
Adding Attachments 261
Including Images with an HTML Message 262
Specifying the Recipients and Subject 263
Specifying the Delivery Server 263
Sending the Message 265
Putting It All Together 265
Faxing from Scripts 271
Sending a Fax with a Script 274
Getting More Information About Faxing 277

7 Windows Management Instrumentation 279
Introduction to Windows Management Instrumentation 279
WMI Functions 280
Namespaces 281
Managing Windows Remotely 283
Making WMI Connections 287
WMI Object Hierarchy 288
Connecting with the `WbemScripting.SWbemLocator` Object 291
Connecting with a Moniker 292
Connecting to the Local Computer 294
Security and Authentication 294
Specifying Security Options 299
WMI Collections and Queries 301
`SWbemServices` 302
WQL Queries 303
`SWbemObjectSet` 305

SWbemObject 306
SWbemMethodSet and SWbemPropertySet 307
Scriptomatic 310
WMI Examples 312
Collecting System Information 312
Managing Printers 313
Monitoring Windows Service Packs and Hotfixes 313
Managing Services and Tasks 315
For More Information 317

8 Active Directory Scripting Interface 319
Managing the User Directory 319
Uses of the Active Directory Scripting Interface 320
Limitations of ADSI with Windows Script Host 321
ADSI Concepts 322
Multiple Inheritance 324
Creating ADSI Objects 325
Directory Security 328
Determining the Difference Between Containers and Leaves 330
ADSI Objects for the WinNT: Provider 332
IADs 333
IADsCollection and IADsContainer 336
Working with ADSI Collections 339
IADsComputer and IADsComputerOperations 340
IADsDomain 342
IADsFileService and IADsFileServiceOperations 345
IADsFileShare 347
IADsGroup 349
IADsMembers 350
IADsNamespaces 351
IADsPrintJob and IADsPrintJobOperations 351
IADsPrintQueue and IADsPrintQueueOperations 354
IADsService and IADsServiceOperations 357
IADsSession 361
IADsUser 362
IIS and Exchange 364
Managing Active Directory 364
X.500 and LDAP Terminology 364

Active Directory Objects 368
RootDSE 368
IADsO and IADsOU 369
Developing ADSI Scripts 370
EzAD Scriptomatic 372
For More Information 373

9 Deploying Scripts for Computer and Network Management 375

Using Scripts in the Real World 375
Designing Scripts for Other Users 376
Using WSF Files 377
WSF File Format Reference 379
Providing Online Help with WSF Files 384
Processing Command-Line Arguments 386
Enclosing More Than One Script 390
Putting It All Together 390
Deploying Scripts on a Network 394
Creating Simple Installation Programs with IExpress 395
Creating IExpress Install Scripts or Batch Files 398
Dealing with User Account Control 400
Providing an Uninstall Option 402
Writing Scripts to Manage Other Computers 403
Remote Scripting 405
Replicating Scripts to Multiple Computers 406
Scripting Security Issues 408
Script Signing 409
The Script Encoder 415
Setting Up Logon Scripts 416
User Profile Logon Scripts 416
Scripts for Logon, Logoff, and Other Events on Windows 7 and Vista 418
Group Policy Logon, Logoff, Startup, and Shutdown Scripts 418
Scheduling Scripts to Run Automatically 421
Writing Unattended Scripts 421
Sending Messages to the Event Log 423
Scheduling Scripts with the Task Scheduler 428

II The Command Line Environment

10 The CMD Command-Line 433

The Command Prompt 433

CMD Versus COMMAND 434

Running CMD 435

Opening a Command Prompt Window with Administrator Privileges 436

CMD Options 437

Disabling Command Extensions 439

Command-Line Processing 439

Stopping Runaway Programs 440

Console Program Input and Output 441

Using the Console Window 442

I/O Redirection and Pipes 443

Copy and Paste in Command Prompt Windows 447

Command Editing and the History List 448

Name Completion 450

Enabling Directory Name Completion 451

Multiple Commands on One Line 452

Grouping Commands with Parentheses 453

Arguments, Commas, and Quotes 454

Escaping Special Characters 454

Configuring the CMD Program 455

AutoRun 455

Environment Variable Substitution 456

The Search Path 456

Predefined and Virtual Environment Variables 459

Setting Default Environment Variables 461

Built-in Commands 462

Extended Commands 475

Listing Files with the `Dir` Command 476

Setting Variables with the `Set` Command 480

Conditional Processing with the `if` Command 482

Scanning for Files with the `for` Command 483

Getting More Information 488

11 Batch Files for Fun and Profit 491
Why Batch Files? 491
Creating and Using Batch Files 492
Batch File Programming 494
Displaying Information in Batch Files 495
Argument Substitution 496
Argument Editing 498
Conditional Processing with `If` 499
The Basic `If` Command 499
Checking for Files and Folders 500
Checking the Success of a Program 500
Performing Several Commands After `If` 501
Extended Testing 503
Processing Multiple Arguments 503
Working with Environment Variables 506
Environment Variable Editing 507
Processing Multiple Items with the `for` Command 508
Using Multiple Commands in a `for` Loop 510
Delayed Expansion 511
Using Batch File Subroutines 513
Prompting for Input 514
Useful Batch File Techniques 515
Processing Command-Line Options 515
Managing Network Mappings 518
Checking for Correct Arguments 519
Keeping Log Files 519

12 The MS-DOS Environment Under Windows 521
MS-DOS Programs on Windows 521
The Virtual DOS Machine 522
MS-DOS and `COMMAND.COM` 524
Configuring the MS-DOS Environment 525
Window and Memory Options 526
`CONFIG.NT` 532
`AUTOEXEC.NT` 535
MS-DOS Environment Variables 536
MS-DOS and Networking 536

Printing from MS-DOS 537
Print Redirection 538
Print Screen 538
Configuring Serial Communications with MS-DOS 539
Using Special-Purpose Devices for MS-DOS 539
Managing MS-DOS Programs 540
When Things Go Awry 540

13 Command-Line Utilities 543
Windows Command-Line Programs 543
The Essential Command Line 544
GUI Shortcuts 545
General-Purpose Shell Programs 547
`findstr` 547
`more` 552
`tree` 553
`xcopy` 554
File-Management Tools 557
`attrib` 557
`cacls` 559
Management Power Tools 563
`driverquery` 564
`runas` 565
`tasklist` 565
`taskkill` 568
`sc` 569
Networking Utilities 571
`ipconfig` 571
`net` 574
`netstat` 584
`nslookup` 586
`ping` 589
`tracert` 591
Getting More Utilities 592

III Introduction to Windows PowerShell

14 Windows PowerShell 593

Introduction to Windows PowerShell 593
- An Object-Oriented Command Shell 593
- Based on the .NET Framework 596
- An Extensible Environment 597

Obtaining Windows PowerShell 598
The PowerShell Environment 600
The PowerShell Command Prompt 601
- Command-Line Editing 602
- Copying and Pasting 603
- Pausing Output and Stopping a Runaway Program 604
- Command-Line Syntax 604

Cmdlets and Objects and Scripts, Oh My! 607
Getting Help 610
Prompting to Complete Commands 612
Aliases 612
- How to Get a Listing of Aliases 612
- How to Define a New Alias 613

Navigating Directories and Other Locations 613
PowerShell Security 615
- PowerShell Scripts and User Account Control 615
- Script Execution Policy 616

PowerShell Profiles 617

15 PowerShell Programming 621

The Windows PowerShell Programming Language 621
Windows PowerShell Syntax 622
Comments 622
Variables and Types 623
- Literal Values 625
- Object Methods and Properties 626
- Object Constructors 627
- String Interpolation 628
- Special Characters 629
- Here-Strings 629

Releasing Variables 630
Predefined Variables 630
Arrays 632
Constants 637
Expressions 638
Comparisons with Arrays 640
String Operators 643
The & (Execute) Operator 646
Operator Precedence 646
Assignment Operators 647
Statement Values 648
Casts 649
Passing by Reference 650
Hash Tables 650
Flow of Control 653
`if` 653
`while` 654
`do…while` and `do…until` 654
`for` 655
`foreach` 656
`switch` 657
`break` 660
`continue` 661
Program Blocks 661
Exception Handling 662
`trap` 662
`try/catch/finally` 663
throw 664
Defining Functions 664
Function Parameters 665
Function Scope 668
The Dot-Source Operator 668
Variable Scope 669
Pipeline Functions and Filters 671
Splatting 672

Using the .NET API 673
Calling Static Member Functions 673
Working with Strings 674
Working with Dates and Times 676
Converting Values 680
Mathematical Functions 680

16 Using PowerShell 683
Real-World PowerShell 683
Command-Line Techniques 685
Generating Objects 685
Filtering 686
Taking Actions 689
Formatting Cmdlet Output 690
The `-f` Operator 690
Working with Files and Folders 691
Seeing Whether a File Exists 697
Reading Text from Files 697
Writing Text to Files 698
Identifying Files by Size 698
Creating Useful Scripts 699
Comment Your Work! 700
Command-Line Processing 700
Writing Modules 701
Exception Handling as an Exit Strategy 702
Using Hash Tables 703
The PowerShell Integrated Scripting Environment 704
Starting the PowerShell ISE 705
Configuring the ISE 706
Creating and Editing Scripts 707
Running Scripts in the ISE 708
Setting Breakpoints and Single-Stepping 709
Interactively Examining and Changing Variables 710
Conditional Breakpoints 711
Remote and Background PowerShell 712
Where to Go from Here 712

IV Appendices

A VBScript Reference 713

VBScript 5.6 Language Features 713
Syntax 714
General Structure of a VBScript Program 714
Data Types and Variables 714
- Note on Dates and Times 716
- Variable Scope 716

Expressions and Operators 716
- Arithmetic Operators 717
- Comparison Operators 717
- Logical Operators 718

Program Statements 718
Functions 720
- Date Function Intervals 722

Predefined Special Values 722
VBA Features Omitted from VBScript 723

B CMD and Batch File Language Reference 725

Batch File Argument and for Variable Replacement 726
Environment Variable Expansion 727
Predefined Environment Variables 727
Command Formatting 729
Built-in Commands 730
- `For` Command Modifiers 733
- `set /a` Expression Operators 734

C Command Line Program Reference 735

Administrative Tools 736
Built-in and Batch File Commands 738
DOS Commands 739
File-Management Commands 740
Handy Programs 741
Networking Tools 741
Software Development Aids 742
TCP/IP Utilities 743
Windows GUI Programs 744

D Index of Patterns and Sample Scripts 747
Index of Patterns 747
Index of Sample Scripts and Batch Files 748

Index 753

E Automation Object Reference 1 (Online)
Collection and Dictionary Objects 1
Script Management and Utility Objects 2
File Access Objects 3
XML/HTML Processing Objects 6
Program Environment Objects 8
Network and Printer Objects 9
Messaging Objects 9
Windows Management Interface (WMI) Objects 11
Active Directory Scripting Interface Objects 13

F WSF and WSC File Format Reference 1 (Online)
XML Conformance 1
Structure of a WSF 2
Structure of a WSC File 3
Tag Syntax 3

G Creating Your Own Scriptable Objects 1 (Online)
Why Create Your Own Objects? 1
Programming Language Options 2
- Visual Basic 3
- C++ and C 3
- VBScript and JScript 4

Creating Objects with Windows Script Component Files 4
- WSC File Format 5
- XML Basics 7
- Understanding the Example 8

WSC File Format Reference 10
Creating a WSC 17
- Using the Windows Script Component Wizard 17
- Defining Properties and Methods 19
- Using Other Objects and Type Libraries 22

Defining Resources 22
Registering the Component 23
Testing 24
Using Scripted Objects from Other Programs 25
Deploying the Object to Other Computers 25
Creating a Practical Object 26

Introduction

Although this book has a brand new title, it is really an updated and revised second edition to *Windows XP: Under The Hood*. The first edition's automotive-themed title came about because of a certain nostalgia that I felt and I know many people share: Don't you long for the good-old days when you could pop the hood of your car and recognize what was underneath? When you could take a wrench and fix just about anything yourself? Cars aren't like that anymore; they've gotten so complex and intimidating that it's hard to imagine digging into one now.

Many of us have come to feel the same way about Windows. Windows has grown into a huge operating system with thousands of complex parts masked behind a slick but seemingly impenetrable graphical user interface (GUI).

This book is an attempt to reclaim those days when we could dig into our machines with confidence and satisfaction. Windows comes with powerful tools and interfaces that let you take control of every detail, if you're willing to roll up your sleeves and dive in.

Whether you're a Windows system administrator or a "power user" who's always on the lookout for more effective ways to use your computer, you're probably familiar with batch files, scripts, and command-line programs. Although they might seem unglamorous, they've been around longer than the PC itself, and sooner or later everyone who uses a computer for serious work runs into them. They might seem like something out of the past, but they've continued to evolve along with Windows. The automation tools provided with Windows are incredibly powerful and useful.

For most people, though, they remain mysterious and are seldom used. I wrote this book to help dispel the mystery. I have five aims in mind:

- To teach how to use the batch file and scripting languages provided with Windows.
- To show how to use command-line utilities and scripting objects as everyday tools.
- To provide an introduction to and reference for the hundreds of command-line programs and scripting objects provided with Windows.
- To provide an introduction to Windows PowerShell, Microsoft's newest command-line automation tool.
- To show you, above all, how you can *learn* to use these tools. No one book is going to solve all your Windows problems. This book teaches you how all these tools work and how to organize your scripting efforts, so you can go beyond canned solutions and create your own.

Although several books on the market are devoted to Windows Script Host, Windows PowerShell, Windows automation tools, and one or two cover Windows command-line utilities, this is the only book I know that combines all four in one volume.

In this book, I explicitly cover Windows 7, Vista, and XP. You can also use the techniques I show you with Windows Server operating systems and Windows 2000 Professional, if you still have that floating around.

Why Learn About This Stuff?

In the age of the GUI, you might wonder why you should spend time learning about scripts, batch files, and command-line programs. Aren't they part of the past, something we can leave behind with a big sigh of relief?

Well, obviously, I don't think so, or I wouldn't have spent months and months slaving away over a hot keyboard, in the dark, just for you. And in case guilt alone isn't enough to make you buy this book, I have some actual good reasons for you.

To begin, here are some important points about scripts and batch files:

- They let you make quick work of repetitive tasks. When you have a large number of files or items to process, or when you perform the same tasks day after day, automation can save you an amazing amount of time. Sure, you can point-and-click your way through running a file through several different programs or adding a user to your network, but when you have to do this job a few hundred times, the graphical approach is a nightmare.

- They encapsulate knowledge and serve as a form of documentation because they record in precise terms how to perform a job. If you write a script or batch file to perform some management function, years from now it can remind you or your successors what the job entails. This makes good business sense.
- They let you use the "insides" of application programs, such as Word and Excel, as tools to write your own programs.
- They let you write procedures that can manipulate files and settings not only on your own computer, but on others in your organization, over your network. Whether you have dozens or thousands of computers to manage, scripting functions can "push" changes to computers without requiring you to physically visit each one.
- They let you write procedures to "reset" a computer's environment to a standard, known configuration. Logon scripts, especially, can set up printers, mapped network drives, and Control Panel settings the same way every time a user logs on, thus eliminating support headaches and user confusion.

If that's the case for learning about scripting and batch files, then how about command-line utilities? Hear ye:

- Many Windows administration, maintenance, and repair functions don't appear anywhere in the Windows GUI. They're found in command-line programs only.
- Sometimes it's faster to type a few letters than to poke around the screen with a mouse!
- Because most command-line utilities are designed to act on data or text files in some particular useful way, you can often use command-line programs as building blocks to perform complex tasks such as sorting, extracting, and formatting information. Instead of writing a custom program, you sometimes use a series of command-line programs to get the job done with little effort. Think of command-line programs as the scissors and staplers on your computer desktop.

Although the Windows GUI has all the flash and gets all the attention, you can see that these behind-the-scenes tools are the real "meat" of the Windows operating system.

How This Book Is Organized

Although this book advances logically from beginning to end, it's written so you can jump in at any location, get the information you need quickly, and get out. You don't have to read it from start to finish, nor do you need to work through complex tutorials. (Even if you're familiar with the material, though, you should at least skim through the references because the batch file language and Windows Script Host program have evolved considerably over the years.)

This book is broken into four major parts. Here's the skinny on each one:

- Part I, "Scripting with Windows Script Host," covers the Windows Script Host tool, introduces the VBScript programming language, discusses the use of objects, and describes the process of writing and debugging scripts. It also provides a detailed reference for many of the scripting objects provided with Windows.
- Part II, "The Command-Line Environment," describes the Windows command language used to write batch files. The batch language has been enhanced considerably since its origin in MS-DOS, and it has become a much more useful way to automate the manipulation of files and directories. Part II also discusses the command-line environment, MS-DOS emulation, and the ways to alter the command environment through administrative tools. Finally, there is a guided tour of the 20 or so most important command-line programs provided with Windows, covering text file management, networking utilities, GUI shortcuts, and more.
- Part III, "Introduction to Windows PowerShell," introduces Windows PowerShell, Microsoft's newest and most peculiar command-line scripting environment. PowerShell is a powerful, and somewhat unusual, programming language. You can use it to perform general-purpose computing, to munch on files and all sorts of data, and to manage Windows workstations, servers and applications. You'll definitely find it worth investigating.
- Finally, Part IV, "Appendices," gives you concise references and indexes to the tools discussed in this book. Where appropriate, items include page references to the sections of the book where you can find information that is more detailed. There is also an index of sample scripts and batch files you can use as a starting point for your own projects. You can download these scripts and files from www.helpwin7.com/scripting. There you can also download some additional bonus appendixes we love but couldn't fit into this printed edition.

Within these sections, each chapter follows a common pattern. An introduction explains a particular type of tool or programming scheme, a reference section describes the tool in exhausting detail, and finally, a discussion shows how to use the tool to apply to real-world programs. I chose this structure because I want this book to serve both as a tutorial for readers who are new to these techniques and as a reference for readers who are familiar with the techniques, but just need a quick refresher.

I also want the material to be somewhat challenging. The early chapters on Windows Script Host and objects take more of a "tutorial" approach, but then the pace picks up. I hope I leave you with some questions unanswered and a few puzzles unsolved; because in your own pursuit of the answers, you learn more than you ever could from reading any book.

Conventions Used in This Book

To help you get the most from this book, special conventions and elements are used throughout.

Text Conventions

Various text conventions in this book identify terms and other special objects. These special conventions include the following:

Convention	Meaning
Italic	New terms or phrases when initially defined.
`Monospace`	Information that appears in code or onscreen or information you type.
Command sequences	All Windows book publishers struggle with how to represent command sequences when menus and dialog boxes are involved. In this book, we separate commands using a comma. Yeah, we know it's confusing, but this is traditionally how Que does it, and traditions die hard. For example, the instruction "choose Edit, Cut" means that you should open the Edit menu and choose Cut.
Key combinations	Key combinations are represented with a plus sign. For example, if the text calls for you to press Ctrl+Alt+Delete, you would press the Ctrl, Alt, and Delete keys at the same time.

In this book's reference lists, which describe the syntax and use of programming statements and objects, the following conventions are used:

Convention	Meaning
`boldface()`	Text and symbols in boldface are to be typed literally.
`italic`	Italics indicate names and values to be replaced by your own data.
`[options]`	Square brackets indicate optional items that are not required. The brackets are not to be typed in.
`{choice A | choice B}`	Curly brackets and the vertical bar (|) indicate items from which you make a choice of one alternative.
`item [, item...]`	Ellipses (...) indicate items that might be repeated as many times as desired.

Special Elements

Throughout this book, you find reference lists, patterns, tips, notes, cautions, cross-references, and sidebars. These items stand out from the rest of the text so you know they're of special interest.

Reference Lists

`Reference Lists`

Describe the syntax and usage of programming statements, object properties and methods, and command-line programs.

Patterns

Pattern

Patterns show how to solve a particular programming problem in a way that you can use in many situations. They provide a general-purpose way of going about some computing task.

Tips

Tip

Tips give you down-and-dirty advice on getting things done the quickest, safest, or most reliable way. Tips give you the expert's advantage.

Notes

Note

Notes are visual "heads-up" elements. Sometimes they just give you background information on a topic, but more often they point out special circumstances and potential pitfalls in some Windows features.

Cautions

Caution

Pay attention to cautions! They could save you precious hours in lost work.

Cross-References

Cross-references point you to other locations in this book (or other books in the Que family) that provide supplemental or supporting information. Cross-references appear as follows:

➔ For more information on Hollerith cards, see Chapter 1, "Windows Script Host," p. 9

Sidebars

Sidebar

Sidebars provide information that is ancillary to the topic being discussed. Read this information if you want to learn more details about an application or task.

This page intentionally left blank

3
Scripting and Objects

IN THIS CHAPTER

- This chapter introduces the concepts of objects, methods, and properties. It provides the background you need for the following seven chapters.
- Read this chapter to see how to use the objects provided with Windows Script Host with different scripting languages.
- To get the most out of this chapter, you should be familiar with at least one script programming language.
- The last section of the chapter shows how you can learn about the many undocumented objects provided with Windows.

Introduction to Objects

All the Windows Scripting languages that I discussed in Chapter 1, "Windows Script Host," provide the basic tools to control a script's execution and to manipulate strings, numbers, dates, and so on, but they don't necessarily provide a way of interacting with Windows, files, or application software. These functions are provided by *objects*—add-on components that extend a programming language's intrinsic capabilities. In this section, I discuss what objects are and introduce the terms you run into as you work with them. In the following sections, I discuss how objects are actually used in several programming languages.

In the most general sense, *objects* are little program packages that manipulate and communicate information. They're a software representation of something tangible, such as a file, a folder, a network connection, an email message, or an Excel document. Objects have properties and methods. *Properties* are data values that describe the attributes of the thing the object represents. *Methods* are actions—program subroutines—you can use to alter or manipulate whatever the object represents.

For example, a file on your hard disk has a size, creation date, and name. So, these are some of the properties you would expect a `File` object to have. You can rename, delete, read, and write a file, so a `File` object should provide methods to perform these tasks. An important aspect of objects is that they are self-contained and separate from

the program that uses them. How the object stores and manipulates its data internally is its own business. The object's author chooses which data and procedures to make accessible to the outside world. In programming jargon, we say that an object *exposes* properties and methods; these items compose its *interface*. Figure 3.1 shows the interface of a hypothetical `File` object.

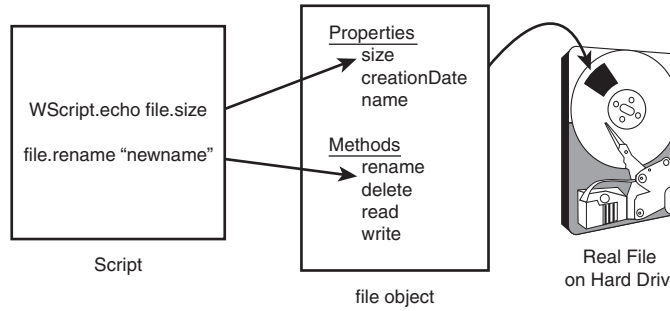

Figure 3.1 This hypothetical `File` object has an interface that can be used by other programs. How the `File` object actually stores its information and does its job are hidden.

Objects need a mechanism through which they can exchange property and method information with a program or script. Because each programming language has a unique way of storing and transferring data, objects and programs must use some agreed-upon, common way of exchanging data. For scripting and for most Windows applications, Microsoft uses what it calls the *Component Object Model* (COM). COM objects can be used by any compatible language, including VBScript, JScript, C, C++, C#, Visual Basic, Perl, and so on. COM objects can also be called *ActiveX Objects*, *Automation Objects*, or *OLE* objects, if they have certain additional features, but regardless of what they're called, the technology is based on COM.

In the next several chapters, you see objects that represent files, folders, network connections, user accounts, printers, Registry entries, Windows applications, email messages, and many more aspects of your computer and network. Windows comes with software to provide you with a wealth of objects. You can also download, buy, or create additional objects of your own devising.

Classes and Instances

Two other terms you're likely to run into while working with objects are *class* and *instance*. The distinction is the same as that between a blueprint for a house and the house itself.

The word *class* refers to the object's definition: its interface (the properties and methods it provides) and its implementation (the hidden programming inside that does the actual work). Hundreds of useful object classes are provided with Windows, and you can add or create more, as discussed in Appendix G, "Creating Your Own Scriptable Objects," which you can download at www.helpwin7.com/scripting.

When you use an object in a program, the class program creates one or more *instances* of the object. An instance is a parcel of computer memory set aside to hold the object's data. The class program then gives your program a *reference* to use when manipulating the object—some identifying value that the class program can use to determine which particular instance of the object your script or program is using. Figure 3.2 illustrates this point: Variables `file1` and `file2` are variables that reference two instances of a `File` object.

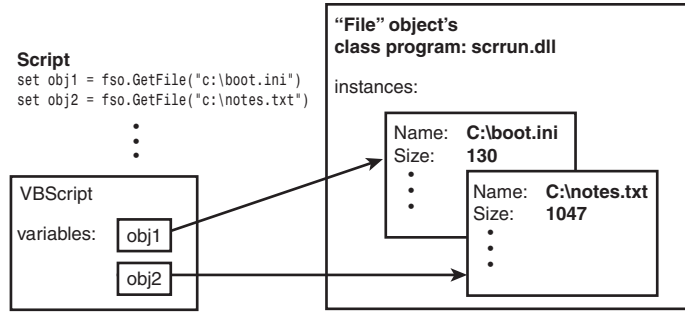

Figure 3.2 `File1` and `File2` refer to objects of the `File` class. This illustration shows two instances of the `File` object.

A reference is treated like any other variable in your program. Just as you can use functions such as `sqrt()` and `left()` to manipulate numeric and string values, you can use the object's methods and properties to manipulate an object reference.

Containers and Collections

As mentioned earlier, an *object* represents some real-world, tangible thing, such as a document or hard drive, and it has properties that represent the tangible thing's attributes. For example, an `apple` object might have attributes such as `color` and `tartness`. The actual data stored for the color might be a character string such as `"red"` or `"green"`. Tartness might be represented as number from `0` (sugary sweet) to `10` (brings tears to your eyes).

An object describing a file on a hard drive might have properties such as `name` (a character string) and `size` (a number). An object representing a hard drive might have properties describing the hard drive's size, volume name, and also the drive's contents.

Now, the contents of a hard drive could be represented as a list of filenames or an array of string values. However, it might be more useful if the hard drive could yield a list of file objects that you could then use to work with the files themselves. This is actually how many objects work. When appropriate, objects can return references to other objects. When an object needs to give you several other objects, it will give you a special object called a *collection*, which holds within it an arbitrary number of other objects. For example, a `Folder` object might represent a folder on your hard drive, and its `Files` property might yield a collection of `File` objects, which represent each of the files in the folder, as illustrated in Figure 3.3.

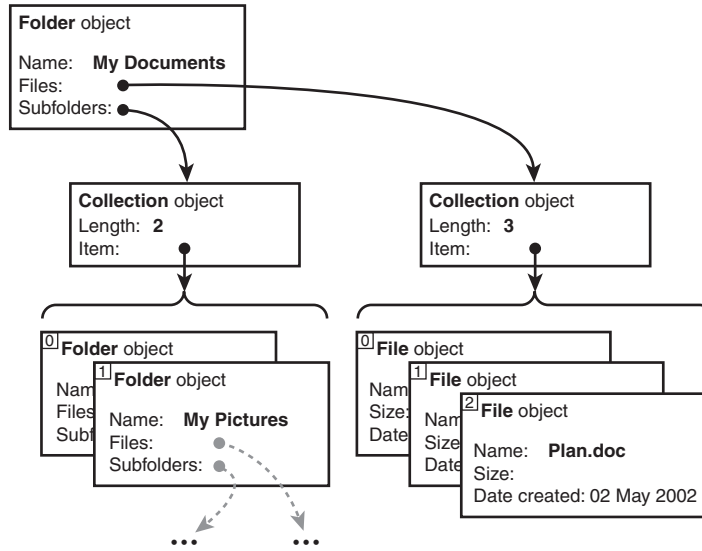

Figure 3.3 `File` and `Folder` objects can represent the contents of a hard drive. The contents of a folder can be represented by collections of `File` and `Folder` objects.

A collection object can actually hold *any* type of object inside it, and the collection object itself has properties and methods that let you count, extract, and work with these objects. This is a common thing to see in object programming: "container" objects that contain other objects of any arbitrary type.

Windows ActiveX objects use container objects that have two properties: `Item` and `Length`. The `Length` property indicates how many items are in the collection. The `Item` property retrieves one of the individual items. For some collections, you can extract

individual objects from the `Item` collection using `Item(0)`, `Item(1)`, and so on. For many collections, though, the `Item` property requires a name or other arcane bit of identifying information. Therefore, each scripting language provides a more general way of letting you examine all the objects in a collection. I discuss this in more detail later in the chapter.

Collections are pervasive in Windows script programming, and some languages have special means of working with them. I give examples of using collections in each of the scripting languages discussed later in the chapter.

Object Naming

Because objects are separate program components, scripts and other programs that use them need a way to locate them and tell Windows to activate them. In this section, I describe how this is done.

Each programmer who creates an object class gives it a name that, with any luck, is fairly self-explanatory. For example, `Scripting.FileSystemObject` is designed to be used by Windows Script Host (WSH) programs to view and manage hard drives, files, and folders. Each of the programming languages you can use with WSH has a way of creating an object instance given just this name. For example, in VBScript, the statement

```
set fsobj = CreateObject("Scripting.FileSystemObject")
```

does the job, whereas in Open Object REXX, the comparable statement is

```
fsobj  = .OLEObject~New("Scripting.FileSystemObject")
```

In either case, the statement causes the WSH interpreter to ask Windows to create an instance of the specified object. Windows looks up the object name in the Registry, finds the name of the program file that manages this object class (usually a file whose name ends in `.dll` or `.ocx`), and fires up the add-on program. The class program creates an instance of the object and gives your script a reference with which it can use and manipulate the object.

I show you how to do this in each WSH-compatible language later in this chapter. For almost all cases, this is all you need.

In the remainder of this chapter, I tell you how to use objects in VBScript and other languages. The next section on VBScript follows the tutorial style of Chapter 2, "VBScript Tutorial," whereas the descriptions for other languages assume more experience with programming.

Finally, at the end of the chapter, I tell you how to find useful objects not discussed in the later chapters of this book.

Using Objects with VBScript

To use objects in VBScript, you first need to create an instance of an object and store its reference in a VBScript variable. Then, the methods and properties of the object can be accessed using *variable.propertyname* or *variable.methodname*. This is easier to demonstrate than explain, so here's an example. This short script tells you whether your C: drive has a folder named `\windows`:

```
set fso = CreateObject("Scripting.FileSystemObject")
if fso.FolderExists("c:\windows") then
    WScript.echo "There is a folder named c:\windows"
end if
```

In the first line of the script, we create an instance of a `Scripting.FileSystemObject`. This is an object class provided with WSH that has handy properties and methods you can use when examining and manipulating disks and files.

Except for the word `set`, this looks just like a typical call to a function with the returned value being assigned to a variable. That's just what it is. `CreateObject` is a function that creates a new object instance. What's new is the word `set`, which VBScript requires you to use to indicate that an object reference is being stored rather than a regular value.

In general, the syntax to create an object instance in VBScript is

```
set variablename = CreateObject("objectname")
```

where *variablename* is the variable you want to use to hold the object reference and *objectname* is the type of object you want to create.

In the second line of the example, we use the `FolderExists` method to find out whether a specified folder exists. Remember that methods and properties are just like regular functions and subroutines; they just happen to "live" in a separate program that provides the object class. The presence of `fso.` before `FolderExists` tells VBScript that the `FolderExists` function is part of the object class to which `fso` refers, which in this example is `Scripting.FileSystemObject`.

Some properties and methods take arguments, as you saw with `FolderExists`. When they do, you have to use parentheses in the same way you would with any other VBScript function or subroutine call. If the method or property returns a function value, you must use parentheses:

```
variable = object.property("arguments", "in", "parens")
```

If a method doesn't return a value, you can omit the parentheses:

```
object.method "arguments", "without", "parens"
```

Does this look familiar? We used objects all through the VBScript tutorial in Chapter 2 in statements such as this:

```
WScript.echo "Today's date is", date
```

Now, you should recognize that `WScript` is an object reference and that `echo` is one of its methods. We never needed to use `CreateObject` to get `WScript` set up, though, because VBScript provides it automatically. The WScript object has several other handy methods and properties that I discuss later in this chapter.

As mentioned earlier, some properties and methods return another object as their values. For example, `Scripting.FileSystemObject` has a `GetFile` method that returns a `File` object. The `File` object can then be used to examine and manipulate the file. Here's a sample script that gives the size and creation date of the program file `\windows\notepad.exe`:

```
set fso = CreateObject("Scripting.FileSystemObject")
set file = fso.GetFile("c:\windows\notepad.exe")
WScript.echo "Notepad.exe was created on", file.DateCreated
WScript.echo "and is", file.Size, "bytes long"
```

The first line is the same as in the previous script, and it creates an instance of the helpful `Scripting.FileSystemObject`.

The second line asks the `FileSystemObject` to return a `File` object representing the file `c:\windows\notepad.exe`. Because I want to use this object several times, I saved it in the variable `file`, using the `set` keyword. (Although `file` is a reserved word in Visual Basic, it's not in VBScript, so it's available for use as a variable name.)

The next two lines use the `File` object's `DateCreated` and `Size` properties. Because these functions don't need arguments, there are no parentheses. The returned date/time and numeric values are printed by the `WScript.echo` method. On my computer, this prints the following:

```
Notepad.exe was created on 2/11/2008 6:56:09 PM
and is 151040 bytes long
```

Automation and Document Files

The `GetObject` function might be used to obtain an object that represents some already existing document file through a process Microsoft calls *Automation*. The `GetObject` function uses the name of the document file to find the appropriate object class server through the Windows standard file type/application association mechanism. You can see file type/associations on Windows XP in Windows Explorer by clicking Tools, Folder Options, File Types. On Windows 7 and Vista, you get there from Control Panel, Programs, Make a File Type Always Open in a Specific Program.

The following sample script uses the `GetObject` function to create a Word document object representing an existing file and print it:

```
set obj = GetObject("C:\docs\userlist.doc")  ' get object for existing document
obj.Printout                                 ' print the document
set obj = Nothing                            ' release the object
```

`GetObject` can also obtain a reference to an already existing object that was created by some other program, through a name called a *moniker*. Several preexisting objects can be used to manage networking, Windows, and Active Directory user accounts. We cover these in Chapter 7, "Windows Management Instrumentation," and Chapter 8, "Active Directory Scripting Interface."

The Difference Between Properties and Methods

I don't know about you, but for a long time I found the distinction between properties and methods to be confusing. Now, it's not crucially important to understand the difference, but if you're curious, I tell you how I finally came to an understanding of sorts.

If you look back in the preceding section, you see I mentioned the `FolderExists` method that is part of the object `FileSystemObject`. Why is `FolderExists` a method and not a property? It comes down to these main points:

- Properties relate directly to *aspects* of the object or, more precisely, of the thing the object represents.
- Properties act like variables: You just refer to them by their name.
- Every property returns a value of some sort. Retrieving a property's value doesn't change anything about the object or whatever it represents.
- Some properties let you assign new values to them. This changes the attribute of the object and the underlying thing it represents.
- Methods are the things the object's program can do for you.
- Methods act like functions and subroutines; they can have arguments passed to them.
- Methods don't have to return a value, but some do.
- Invoking a method can change something about the object or the real-world thing it represents.

Therefore, `FolderExists` is a method because it takes an argument (the name of the file it is to look up). Properties don't need arguments because they are intrinsic attributes of the object itself. As such, they don't need any additional information to return a value.

There is something else I should mention about properties: In many cases, you can both evaluate them (examine their values) and assign new values to them. They work just like variables in this regard. The difference is that when you assign a new value to a property, the object software makes a corresponding change in the actual thing the object represents. For example, assigning a new value to a `File` object's `Name` property changes the actual file's name, as shown here:

```
WScript.echo file.Name          ' evaluate the property
file.Name = "newname"           ' assign a new value to the property
```

Keep in mind, however, that some objects don't let you change a property's value. In this case, the object's documentation calls it a *read-only* property. And, a few specialized objects are designed so that changing the object's properties doesn't immediately change the underlying thing the object represents, until you use a special method that "commits" the change. Again, the object's documentation discusses this, and I point out a few objects of this sort in later chapters.

Nested Objects

One other thing I want to point out is that you don't necessarily need to save every object reference in a variable. In the previous example that displayed information about `Notepad.exe`, if I only wanted to see the creation date, I could have skipped the step of storing the `File` object in variable `file` and could have used these statements:

```
set fso = CreateObject("Scripting.FileSystemObject")
WScript.echo "Notepad.exe was created on",_
    fso.GetFile("c:\windows\notepad.exe").DateCreated
```

In this case, VBScript refers to `fso` to call the `GetFile` method, and the returned object is used to fetch the `DateCreated` property. It's not unusual to see several levels of objects this way; this is called a *nested object reference*.

When you're working with Microsoft Word objects, this is common. In scripts or Word macros, you might often see statements like these:

```
ActiveDocument.PageSetup.Orientation = wdOrientLandscape
ActiveDocument.PageSetup.TopMargin = InchesToPoints(0.5)
ActiveDocument.PageSetup.BottomMargin = InchesToPoints(0.5)
ActiveDocument.PageSetup.PageWidth = InchesToPoints(11)
```

In this example, the `ActiveDocument` object returns a `PageSetup` object, which has orientation and margin properties you can set. You could save yourself some extra keystrokes in creating this script by saving a reference to the `PageSetup` object, as follows:

```
set ps = ActiveDocument.PageSetup
ps.Orientation = wdOrientLandscape
ps.TopMargin = InchesToPoints(0.5)
ps.BottomMargin = InchesToPoints(0.5)

ps.PageWidth = InchesToPoints(11)
```

However, VBScript has a special program construct called the `With` statement that makes this even easier. The previous example could be rewritten this way:

```
with ActiveDocument.PageSetup
    .Orientation = wdOrientLandscape
    .TopMargin = InchesToPoints(0.5)
```

```
    .BottomMargin = InchesToPoints(0.5)
    .PageWidth = InchesToPoints(11)
end with
```

The `With` statement lets you specify an object reference that is taken as the "default" object between `With` and `End With`. Inside the `With` statement, you can refer to the default object's methods and properties by preceding them with a period but no variable name. Not only can this save you a lot of typing, but it's easier to read, and it lessens the workload on VBScript, thus speeding up your script.

Note

If you need to, you can refer to other objects inside the `With` statement by using the fully spelled-out *`object.method.etc`* syntax.

Releasing Objects

When you create an object, Windows activates the object's class server program to manage the object for you. In the case of `Scripting.FileSystemObject`, you usually create one of these objects at the beginning of your script and use it throughout. When your script completes, Windows releases the object you've created. The class server program takes care of freeing up its memory and other housekeeping chores. You don't have to worry about it at all.

However, if you use a script to create multiple objects, you might find that it's appropriate to explicitly release them when you are through using them. For instance, a script that creates multiple Word documents should tell Word to close each document when you're finished with it; then, the script should release the document object, lest you end up with hundreds of documents open at once.

You can explicitly tell an object you're finished with it by setting the variable that holds the object reference to the value `Nothing`. Later in this book, there are examples of this in some of the sample scripts.

Working with Collections

If you ask `Scripting.FileSystemObject` for the files or subfolders contained in a folder or drive, it might need to return multiple `File` or `Folder` objects. To manage this, it actually returns a single *collection* object that contains all the `File` or `Folder` objects inside it. You can then examine the contents of the collection to look at the individual items.

A collection object has a `Count` property that tells how many items are inside and an `Item` method that returns a specific item from the collection. This would lead you to expect that you could write a script like this to print the names of the files of the root folder on your hard drive:

```
set fso = CreateObject("Scripting.FileSystemObject")
set files = fso.GetFolder("c:\").Files
for i = 1 to files.Count
    WScript.echo files.Item(i).Name
next
```

However, this script doesn't work. With a folder collection, `Item` doesn't allow you to retrieve items by number. It requires you to specify the *name* of the particular object you want, and if you don't yet know the names, this isn't very useful.

To scan through collection objects that can't be referenced by number, each scripting language provides a way to scan through collections without knowing what they contain. VBScript, for example, provides a special version of the `For` loop called `For Each`.

Pattern

To scan through a collection object named `collection` in VBScript, use the `For Each` loop as follows:

```
for each objectvar in collection
   :statements using objectvar
next
```

The `For Each` loop runs through the statements once for each object in *`collection`*, and the variable *`objectvar`* is made to refer to each of the individual objects in turn. Using `For Each`, and using a variable named `file` to hold the individual file objects, our folder-listing script now works:

```
set fso = CreateObject("Scripting.FileSystemObject")
set files = fso.GetFolder("c:\").Files
for each file in files
    WScript.echo file.Name
next
```

You could even use the following shorter version:

```
set fso = CreateObject("Scripting.FileSystemObject")
for each file in fso.GetFolder("c:\").Files
    WScript.echo file.Name
next
```

Now, if you don't plan on writing scripts in any other languages, skip ahead to the section titled "Using the `WScript` Object" on p. 111 of this chapter for more information about the built-in `WScript` object.

Using Objects with JScript

JScript, like VBScript, is a strongly object-oriented language—it's expected that programmers will use objects to extend its power. JScript supplies 11 built-in object types, and programmers can create generic and structured object types in scripts. I don't discuss the intrinsic object types here because this book focuses on external scripting and Windows management objects.

External COM/ActiveX objects are created using the `new` statement, as follows:

```
variablename = new ActiveXObject("objectname");
```

Here, *`variablename`* is the declared variable that is to receive the new object reference.

After you have an object variable in hand, its methods and properties are accessed using *`variable.propertyname`* or *`variable.methodname`*. For example, this is a short script that tells whether your C: drive has a folder named `\windows`:

```
var fso;
fso = new ActiveXObject("Scripting.FileSystemObject");
if (fso.FolderExists("c:\windows"))
    WScript.echo("There is a folder named c:\\windows");
```

Parentheses must be used on all method calls, even if the return value is not used—VBScript tolerates a statement such as

```
WScript.echo "There is a folder named c:\windows"
```

but JScript does not.

Case Sensitivity

In the WSH environment, JScript programs have access to a predefined object named `WScript`, which provides several useful methods and properties pertaining to the script's environment, execution, and debugging. Because JScript is a case-sensitive language, to use this object you must type `WScript` with a capital *W*, exactly as written here.

However, the method and property names of ActiveX and COM objects are *not* case sensitive. For example, JScript permits you to type `WScript.echo` or `WScript.Echo`.

Working with Collections

If you are used to using JScript with Internet Explorer in browser or server-side scripts, you might be familiar with objects that return collections of objects. Many Internet Explorer objects let you scan through the object collection using JScript's `for…in` statement.

However, most other objects' collections do not work with `for...in` and you must use an `Enumerator` object to work with them. This is true of most objects you encounter in the WSH environment. JScript's `Enumerator` object gives you a way of accessing a collection by stepping forward or backward through the collection's list of objects.

To use a collection provided by a scripting or other ActiveX object, you must first convert it to an enumerator:

```
enumObject = new Enumerator(collectionObject);
```

The `Enumerator` object has no properties (in particular, no `Length` property; if you need to know the number of items, you must get it from the original collection's `Country` property). It does have an internal concept of its "position" in the collection and has methods that let you move the current position forward or back to the beginning. Its four methods are listed in Reference List 3.1.

REFERENCE LIST 3.1 Methods of the JScript `Enumerator` Object

`Item`
Returns the current item from the collection. The return value is an object of whatever type the collection holds. If the collection is empty or the current position is undefined, it returns `undefined`.

`AtEnd`
Returns a Boolean value: `True` if the current item is the last in the collection, if the current position is undefined, or if the collection is empty. Otherwise, `False` is returned.

`moveFirst`
Makes the first item in the collection the current item. If the collection is empty, `atEnd` is immediately `True`.

`moveNext`
Makes the next item in the collection the current item. If the collection is empty or the current item is already the last in the collection, `item` is `undefined`.

Although a newly created enumerator should be positioned on the first item automatically, it's a better style to use `moveFirst` before examining `atEnd` or `item`.

Pattern

To scan through a collection object named `obj`, use an enumerator in this way:

```
e = new Enumerator(obj);
for (e.moveFirst(); ! e.atEnd(); e.moveNext()) {
    x = e.item();
    ⋮statements using x
}
```

Here's an example. This script lists the names of all the files in the root folder on the C: drive:

```
var fso, e, file;

fso = new ActiveXObject("Scripting.FileSystemObject");

e = new Enumerator(fso.GetFolder("c:\\").files);
for (e.moveFirst(); ! e.atEnd(); e.moveNext()) {
    file = e.item();
    WScript.echo(file.name);
}
```

Now, if you plan on writing scripts only in JScript, you can skip ahead to the section titled "Using the `WScript` Object" for more information about the built-in `WScript` object.

Using Objects with ActivePerl

ActiveState's ActivePerl lets you run Perl scripts in the WSH environment. Perl's environment is already rich with file-management and network-communication tools, and if you're already a skilled Perl programmer, you might wonder what WSH can add. In other words, why use cscript or wscript to run Perl, when you could just run `perl.exe` directly?

The answer is that in the WSH environment, it's a simple matter to access COM, OLE (Automation), and ActiveX objects. The helpful `$WScript` object is predefined in the WSH environment. COM objects are the key to accessing network configuration, Active Directory, and Windows Management Instrumentation (WMI). Although you probably don't want to bother with the Windows script objects for file and directory management, the system-management tools make WSH worthwhile.

Running Perl Scripts in WSH

The ActivePerl installer creates two file associations for Perl files: `.pl` (Perl) is associated with `Perl.exe`, and `.pls` (`PerlScriptFile`) is associated with WSH.

If you use the familiar `.pl` filename extension for Perl programs that you want to run in the WSH environment, you have to use the command

```
cscript /engine:Perlscript myscript.pl
```

to fire them up. Because you might want to start scripts with the command line or from Explorer, your life is much easier if you use the extension `.pls` for programs meant to be used with WSH. This way, you can double-click the files in Explorer or use commands such as

```
start myscript.pls
myscript
cscript myscript.pls
```

to start script files; WSH knows what to do. You can also use PerlScript inside the structured `.WSF` files I discuss in Chapter 9, "Deploying Scripts for Computer and Network Management."

Here are some important things to remember when writing Perl scripts for use with WSH:

- You cannot use familiar Perl command-line switches such as `-w`. You need to directly set option values in the script.
- Any command-line arguments specified to cscript or wscript are not placed in the `ARGV` array. Instead, you must fetch command-line arguments from the `$WScript->Arguments` collection.

The Perl Object Interface

ActivePerl can interact with COM, ActiveX, and OLE (Automation) objects. The extended syntax for accessing methods is

```
$objectname->Method[(arguments[, ...])];
```

Here's an example:

```
$myobject->SomeMethod
$myobject->Anothermethod("argument", 47);
```

The syntax for accessing `Property` values is

```
$objectname->{Propertyname}
```

Here's an example:

```
value = $myobject->{Length};
$myobject->[color] = "Red";
```

Because the syntax for accessing methods and properties is different, you must take care to check the COM object's documentation carefully to determine whether a value you want to use is a property or method.

Caution

If you attempt to access an object property or method that does not exist or if you have misspelled the name, by default Perl does *not* generate an error. The result is simply undefined (`undef`). This makes it difficult for you to debug your script.

To avoid this pitfall, put

```
$^W = 1;
```

at the beginning of every script file. This causes Perl to print an error message if you attempt to reference an undefined method or property. I have found that the error message might not tell you clearly that the problem is an unrecognized property or method, but at least you get *some* warning.

To set a read/write property, you can simply assign a value to it, as in the following:

```
$file->{name} = "newname";
```

The standard `WScript` object, which I discuss in more detail later in the chapter, is predefined by the WSH environment and is available to a Perl script. For example, the current version of WSH can be printed with this script:

```
$WScript->Echo("The version is", $WScript->Version);
```

You could use conventional Perl I/O and write

```
print "The version is ", $WScript->Version;
```

instead. However, `print` writes to `stdout`, which is undefined when the script is run by WScript (the windowed version of WSH). A script using `print` works under Cscript, but not WScript. It's up to you, of course, but if you want your script to work equally well within either environment, use the `$WScript.Echo` method for output.

Note

Perl is a case-sensitive language, and object variable names such as `$WScript` must be typed exactly as shown here. However, an object's method and property names are *not* case sensitive.

To create an instance of an Automation, OLE, or ActiveX object, you can use either of two methods. The simplest is to use the `CreateObject` method provided with the built-in `$WScript` object, as in

```
$myobj = $WScript->CreateObject("Scripting.FileSystemObject");
```

You can also use the ActivePerl `Win32::OLE` extensions provided with ActivePerl:

```
use Win32::OLE;
$excel = Win32::OLE->new('Excel.Application')
    or die "OLE new failed";
```

For information on the OLE extensions, see the ActivePerl help file.

Working with Collections

Some of the COM objects you encounter in this book and elsewhere return *collection* objects, which are containers for a list of other objects. For example, the `Drive` property of `Scripting.FileSystemObject` returns a collection of `Drive` objects, and the `WScript.Arguments` property returns a collection of `Argument` objects. I discussed the methods and properties of the collection object earlier in the chapter.

Because the items of a collection object can't be obtained by an index value (at least, not directly), they must be "scanned" using an enumerator object. An enumerator gives you access to a collection object by maintaining the concept of a current location in the list, which you can then step through. To scan the list a second time, you must reset the current location to the beginning of the list and then step through the list again.

There are three ways to enumerate a collection. First, you can explicitly create an enumerator object, as in the following example:

```
$^W = 1;

$fso  = $WScript->CreateObject("Scripting.FileSystemObject");
$fls  = $fso->GetFolder("C:\\")->Files; # get collection of files in c:\
$n    = $fls->{Count};                  # just for kicks say how many there are
print $n, " files\n";

$enum = Win32::OLE::Enum->new($fls);    # create enumerator object
while (defined($file = $enum->Next)) {  # assign $file to each item in turn
    print $file->{Name}, "\n";          # print the file names
}
```

A second way uses the `in` function to hide the enumerator: The `in` operator returns `Win32::OLE::Enum->All(`*`obj`*`)`, which in turn returns a Perl array given a collection object. Instead of creating `$enum` and using a `while` loop in the previous example, I could have written the following:

```
foreach $file (in $fls) {
    print $file->{Name}, "\n";
}
```

Seeing this, you might guess that the third way is to use `in` to create an array of the subobjects, which you can then access explicitly:

```
@file = in($fls);
for ($i = 0; $i < $fls->{Count}; $i++) {
    print $file[i]->{Name}, "\n";
}
```

Of the three methods, the `foreach` method is the most straightforward, and it's the easiest on the eyes and fingers. Unless you really want to use an array, I recommend `foreach`.

Now, you might want to skip ahead to the section titled "Using the `WScript` Object" for more information about WSH built-in objects.

Using Objects with ActivePython

Many powerful CGI (Web-based) applications are written in Python, and ActiveState's ActivePython does a great job of integrating Python into the ASP scripting environment. This means it can also be used with WSH. Python, like Perl, has a rich set of

built-in and add-on functions that give you complete access to the Windows API, so the scripting utility objects described in this book aren't going to be terribly interesting to the Python programmer. Still, you might want to use the scripting objects in the interest of increasing your scripts' language portability, and because COM/ActiveX objects are the only way to get programmatic access to Automation servers such as Microsoft Word.

Unlike Perl, Python was designed from the ground up as an object-oriented language. Objects, properties, and methods are its bread and butter, so to speak. If you're coming to Python from a background in other languages, there are a few points of which you should be aware:

- The `WScript` object discussed throughout this chapter is predefined. Python is case sensitive, so the object must be referred to as `WScript`, exactly.
- Although Python is generally case sensitive, the names of COM object methods and properties are not.
- Perhaps the easiest way to create ActiveX/COM objects is with the `CreateObject` method provided by WScript. Here's an example:

```
fso = WScript.CreateObject("Scripting.FileSystemObject")
files = fso.GetFolder("C:\\").Files
```

- You cannot directly assign a value to a COM object property. You must use

```
object.SetValue("propertyname", newvalue)
```

  instead.
- Python automatically imports all predefined constants associated with an OLE object when you create an instance of the object. The constant values are created as properties of `win32com.client.constants`. For example, if you should create a Microsoft Word document, the value `win32com.client.constants.wdWindowStateMinimize` will be defined.

For more information about COM integration with Python, see the ActiveState documentation for the `win32com` package.

Working with Collections

Python automatically treats COM collection objects as enumerations. The easiest way to scan through the contents of an enumeration is with the `for…in` statement, as in this example:

```
fso = WScript.CreateObject("Scripting.FileSystemObject")
files = fso.GetFolder("c:\\").Files

for file in files:
    print file.name
```

Now, continue with the next section for more information about the built-in WScript object.

Using the WScript Object

WSH provides a built-in object named WScript for all scripts in all languages. We've used its Echo method in many of the examples in this book. WScript has several other methods and properties, as listed in Reference List 3.2, that you might find useful in writing scripts.

REFERENCE LIST 3.2 Properties and Methods of the WScript Object

Properties

Arguments
Returns a collection of WshArguments objects, representing the strings on the command line used to start WScript or Cscript. For example, if a script is started with the command

```
WScript myscript.vbs aaa bbb
```

or

```
myscript aaa bbb
```

then WScript.arguments.item(0) would yield "aaa" and WScript.arguments.item(1) would yield "bbb". WScript.arguments.length gives the number of arguments.

I discuss arguments in more detail in the next section.

BuildVersion
Returns a number identifying the current version of Windows Script Host. This number might vary between versions of Windows and as WSH is updated through Windows Update. I have seen WSH on Windows 7 return 0 for this property. I would not trust it to be usable.

FullName
Returns the full path and filename of the WSH program that is running your script (for example, c:\Windows\System32\cscript.exe).

Interactive
A Boolean value: True if the script is running in Interactive mode and False if in Batch mode. You might set this property using the //I or //B switch on the command line, or you might directly set the value in a script (for example, WScript.Interactive = False). In Batch mode, message and input boxes do not appear.

Name
Returns the name of the script host program (for example, `"Windows Script Host"`).

Path
Returns the name of the directory containing the script host program (for example, `"c:\Windows\System32"`).

ScriptFullName
Returns the full path and name of your script file (for example, `"c:\test\myscript.vbs"`).

ScriptName
Returns the name of your script file (for example, `"myscript.vbs"`).

StdErr, StdIn, and StdOut
These are file streams that can be used to read from the standard input or write to the standard output and error files. I discuss these in Chapter 4, "File and Registry Access." These properties are available with `cscript` only, not `wscript`.

Version
Returns the version of WSH (for example, `"Version 5.7"`).

Methods

CreateObject(*progid* [, *prefix*])
Similar to the built-in `CreateObject` function. With a *`prefix`* argument, it creates connected objects that can communicate events to the script. (Events are beyond the scope of this book.)

ConnectObject *object*, *prefix*
Connects an existing *`object`* to the script using event handler functions whose names begin with the string *`prefix`*.

DisconnectObject *object*
Disconnects the script from an object's events.

Echo *arg* [, *arg*]…
Displays any number of arguments of any type, formatted as strings and separated by spaces. Cscript writes them to the standard output, whereas WScript displays them in a pop-up message box.

GetObject(*filename* [, *progid*][, *prefix*])
Creates an object based on information stored in a file (for example, a document). If *`progid`* is not specified, it is determined from the file type. *`prefix`* might be specified to connect object events to the script.

`GetObject` can also obtain a reference to a preexisting object by specifying a special name called a *moniker*. This is illustrated extensively in Chapters 7 and 8.

Quit [*errorcode*]
Terminates the script. If a numeric value is specified, it is returned as the process's exit code—this can be useful when running scripts from batch files.

Sleep *msec*
Causes the script to pause for *msec* milliseconds. For example, `WScript.sleep 1000` pauses for one second.

Of the properties and methods listed, the most useful are the `Echo` and `Arguments` properties. Let's see how you can use arguments to control what a script does when you run it.

Retrieving Command-Line Arguments

The use of command-line arguments is a common way of specifying information to a script at the moment it's run. The most common use for this is to write scripts that manipulate files, user accounts, or computers. The script can be written in a generic way, so that you can specify the particular files, people, or what-have-you at the time you run the script. For example, a script to process a file could be written like this:

```
filename = "specialdocument.doc"
'statements to operate on the file named filename
⋮
```

However, if you wanted to use this script to work with a different file, you'd have to edit the script. If you want a more general-purpose script, write the script to get the filenames from its command line, so you can simply type something like this:

```
C:\> myscript some.doc another.doc
```

Then, the script will operate on the files whose names you typed, rather than on a file whose name is built in to the script.

Usually, each programming language has its own way of providing command-line arguments to a program, but in the WSH environment, there is only one way they are obtained—through the `WScript` object's `Arguments` property.

The `WScript.Arguments` property returns a collection of objects, one for each item listed on the script's command line. You can write a script to use these arguments this way, more or less:

```
for each filename in WScript.arguments
    ' statements to operate on the file named filename
    ⋮
next
```

Of course, you have to use whatever method of manipulating objects and collections is appropriate to the script language you're using (this example is in VBScript). With `script myscript.vbs`, the command line

```
C:\> myscript some.doc another.doc
```

sets up the `WScript.Arguments` collection with two items: `some.doc` and `another.doc`. In VBScript, the `for each` statement lets your script process them in turn.

If you don't specify any command-line arguments, though, this script does nothing at all. It's best to have a script tell the user how to use it properly in this case. Here's a scheme for writing command-line scripts that you might find to be handy.

Pattern

When a script uses command-line arguments to specify what files (or users, computers, or whatever) to work with, it should explain how to use the script if no arguments are specified:

```
if WScript.arguments.length = 0 then
    ' no arguments on the command line? Display usage information, then quit
    WScript.echo "This script processes the named files"
    WScript.echo "by doing etc etc etc to them".
    WScript.echo "Usage: myscript file [file ...]"
    WScript.quit
end if
for each filename in WScript.arguments
    ' commands to process filename go here
    ⋮
next
```

Alternatively, you might want your script to operate on a default file if no files are named on the command line. Such a script should use a subroutine to do the actual processing of the files, so the subroutine can be called with either the default file or with specified files. In VBScript, it looks like this:

```
if WScript.arguments.length = 0 then
    ' no arguments on command line -- process file "default.file"
    process "default.file"
else
    ' process each of the files named on the command line
    for each filename in WScript.arguments
        process filename
    next
end if

sub process (filename)
   ' statements to process filename
   ⋮
end sub
```

In Chapter 9, I show you how to use more powerful types of command-line processing.

Locating and Using Unusual Objects

Several powerful, commonly used objects provided with Windows are documented by Microsoft in the Windows Scripting reference documents, and I discuss most of these in Chapters 4–9. If you're new to scripting, these should be enough to get you started, so you might want to skip ahead to Chapter 4.

In addition to these standard objects, many developers and companies provide add-in objects for free or for sale. There is, however, a wealth of objects already on your computer; hundreds of COM objects are supplied with Windows, and hundreds more are added if you install applications such as Word, Excel, and Visio. Many are designed just for the use of specific application programs and are of no use to script authors. Others are general-purpose objects for use by scripts and compiled programs. How can you tell which objects are installed on your computer and of those, which are useful for scripting? To be honest, identifying useful objects is tricky business, but if you enjoy detective work, read on.

To get an idea of what I mean by "hundreds of objects," take a look at the Windows Registry.

Caution

Improper changes to the Windows Registry can make your computer nonfunctional. There is no `undo` command in the Registry Editor, so be very careful not to make *any* changes while examining the Registry.

To view the Registry on Windows 7 or Vista , click Start, type **`regedit`** into the search box, and press Enter. On XP, click Start, Run; type **`regedit`**; and press Enter. Expand the entry for `HKEY_CLASSES_ROOT` and scroll down past the `.`*`xxx`*`-format` entries to those that spell out names like "something dot something," as shown in Figure 3.4. Most of the entries from here down represent object classes; you can tell which ones do by the presence of a `CLSID` or `CurrVer` key under the object name.

In Figure 3.4, the `FaxControl.FaxControl.1` entry has a `CLSID` entry, so it is an object. A `CLSID` (or *class ID*) is a long, essentially random number that object authors use to give their object a unique "fingerprint." A `CurrVer` entry, such as the one found under `FaxControl.FaxControl`, is used when there's a chance more than one version of the class program might be installed on your computer. The `CurrVer` value tells Windows where to look to find the class information for the most recent version of the object. Find that entry, and you find the object's `CLSID`.

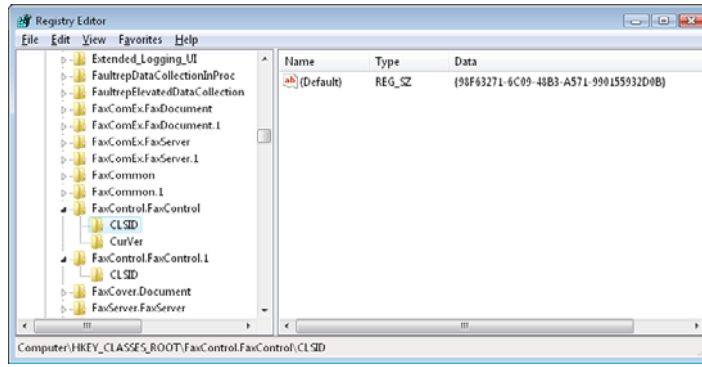

Figure 3.4 COM object classes are listed in the Registry under `HKEY_CLASSES_ROOT`, after the `.xxx` entries. Objects have an associated `CLSID` entry.

The first step in scouting out new and interesting objects is to peruse the Registry for names that sound interesting. For this example, I'll follow up on the `FaxControl.FaxControl.1` object from Figure 3.4.

When you've found a `CLSID` value for a potentially interesting object, locate the matching value under `My Computer\HKEY_CLASSES_ROOT\Clsid`, where you find the information Windows uses to locate and run the program file that actually manages the object.

Figure 3.5 shows the class information for `FaxControl.FaxControl.1`. The `InprocServer32` entry shows the actual program module (usually a DLL or OCX file) that manages the object. In this case, the program is `\WINDOWS\system32\Setup\fxsocm.dll`. The name of this object and the location of its DLL make it sound like it might be used for setting up the Fax service. But how?

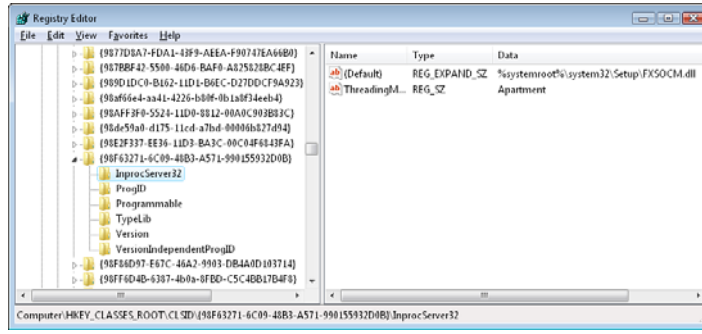

Figure 3.5 Class ID information for the `FaxControl.FaxControl.1` object.

The first thing you have to check is whether the object is even suitable for use in script programs; some aren't. The first test, then, is to see whether you can create an object in your chosen scripting language. Use the *server.object* name you found under HKEY_CLASSES_ROOT. In VBScript, it looks like this:

```
set obj = CreateObject("FaxControl.FaxControl.1")
WScript.echo "CreateObject worked!"
```

If this script produces an error message, the object can't be used in scripts. If it runs without an error message, as it did when I tested FaxControl.FaxControl.1, the object has passed the first hurdle.

Your next step should be to search the Internet for references to the object name (for example, search for FaxControl.FaxControl.1 or FaxControl.FaxControl). I've found that Google is a great place to start. If you see references to pages on msdn.microsoft.com, these might point to complete documentation for the object in question. Be sure to search Google's "Groups" section, too. Many programmers haunt the comp.*xxx* groups, and if you're lucky, you might find an archived discussion about the object. (Unfortunately, if you do a Google search for FaxControl.FaxControl, you will most likely find only references to this very discussion from this book or from its first edition titled *Windows XP Under the Hood*, but no documentation.)

If you can't find documentation online, Microsoft or the object's creator might have supplied a help file describing the object. See whether the Clsid Registry values list a help file ending in .hlp or .chm. If it does, at a command prompt type

```
start pathname\helpfile.xxx
```

where *pathname\helpfile.xxx* is the full path to the help file listed in the Registry. This might show you how the object works. In the case of FaxControl.FaxControl.1, there is no help file.

Note

If the help file has the .hlp extension and you're using Windows 7, Vista or Windows Server 2008, you have to install the old Windows Help viewing program before you can open the .hlp file. Go to www.microsoft.com and search for "download winhlp32.exe".

If no help file is named, don't give up. Because COM objects are designed to be used by many programming languages, they can—if their developer wanted them to—provide a list of methods, properties, and their arguments to any program that asks. If your mystery object has this feature, you might be able to burrow into the object's program file to find its usage information.

The easiest way to do this is with an *object browser*, a program that's designed to do just this sort of burrowing. Microsoft provides one with many of its applications. If you have Microsoft Word, Excel, or PowerPoint, the Object Browser is included as part of the Macro Editor. Start the application and click Tools, Macro, Visual Basic Editor. Then, click View, Object Browser. If you have the full developer's version of Visual Basic installed, run it and click View, Object Browser.

To view information for a questionable class, you need to tell Word (or Visual Basic, and so on) to look into the class's program file. To do this, click Tools, References. Click Browse and locate the DLL or OCX file you found in the Registry. Click Open, and the library appears as a checked item in the Available References list, as shown in Figure 3.6.

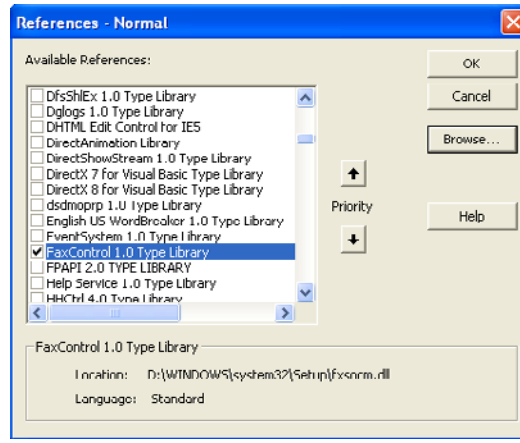

Figure 3.6 Selecting an object class type library to view in the Object Browser.

When the object type is listed and checked under Available References, click OK.

Then, select the class name from the library list in the upper-left corner of the Object Browser window, as shown in Figure 3.7. Choose object types in the left panel; the browser displays the object's methods, procedures, and predefined constants in the right panel under Members. You can select the objects in this list one by one, and in the bottom panel, the browser displays the method or procedure's arguments, if any, and any explanatory text it can dig up.

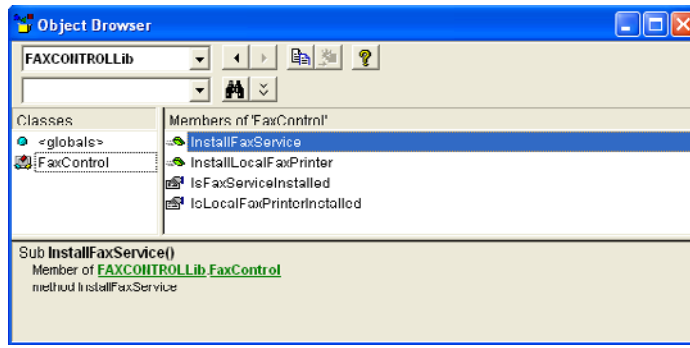

Figure 3.7 Viewing a class's type information in the Object Browser.

If you don't have any of the applications I've mentioned, another tool, called the OLE/COM Object Viewer, is provided with the Windows XP and Windows 2000 Resource Kits which you can download from www.microsoft.com. You can also download this tool directly from www.microsoft.com by searching for "OLE/COM Object Viewer."

The OLE/COM Object Viewer is much more difficult to use than the Object Browser. Here are some tips for using this viewer:

- Try to find the object of interest under Object Classes, All Objects. I've found that not all the objects I'm interested in are listed there. For example, `Scripting.Dictionary` is present, whereas `Scripting.FileSystemObject` is missing. If you can't find it there, look under Type Libraries.
- Double-click the library or object to view its class information. This information is designed for COM object programmers, not end users, so it's going to be tough to understand.
- `Typedef` items list some predefined constant values used by all the objects provided by the server.
- `Coclass` items define the objects the class server can create. If you view the contents of a coclass item, you find the class's predefined constants, properties, and methods.

Although either object browser can show you what sorts of values the methods and properties expect and return, it can't tell you what these values mean, so you have to experiment to find out if and how you can use them. In the case of `FaxControl.FaxControl.1`, the Object Browser showed two properties and two methods, as listed in Reference List 3.3.

REFERENCE LIST 3.3 Properties and Methods of the *FaxControl.FaxControl* Object

Properties:

`IsFaxServiceInstalled`
Returns a Boolean value.

`IsLocalFaxPrinterInstalled`
Returns Boolean value.

Methods:

`InstallFaxService`
Returns no value and takes no arguments.

`InstallLocalFaxPrinter`
Returns no value and takes no arguments.

This sounds pretty straightforward. There are no arguments to supply to these methods, so there's no detective work required there. Also, the names sound pretty self-explanatory. This object lets you know whether the Fax service and the Fax Printer are installed, and it can install them. But does it work?

Here's a sample script I wrote to check:

```
set obj = CreateObject("FaxControl.FaxControl.1")
WScript.echo "IsFaxServiceInstalled =", obj.IsFaxServiceInstalled
WScript.echo "IsLocalFaxPrinterInstalled =", obj.IsLocalFaxPrinterInstalled
```

The results when I ran it on a Windows XP computer that had a modem but did not have the fax service set up were

```
IsFaxServiceInstalled = 0
IsLocalFaxPrinterInstalled = 0
```

When I ran the script

```
set obj = CreateObject("FaxControl.FaxControl.1")
obj.InstallFaxService
```

Windows Setup asked for my Windows XP CD disk and installed the Fax service and printer. The first script's output became this:

```
IsFaxServiceInstalled = -1
IsLocalFaxPrinterInstalled = -1
```

Here, `-1` means `True` (any nonzero value means True), so the object does the job you expect it to. Here's a script that can automatically be sure a user's Windows XP system has the Fax service as well as a fax and printer installed:

```
set obj = CreateObject("FaxControl.FaxControl.1")

if not obj.IsFaxServiceInstalled then
```

```
    WScript.echo "Installing Fax Service..."
    obj.InstallFaxServic
elseif not obj.IsLocalFaxPrinterInstalled then
    WScript.echo "Reinstalling Fax Printer..."
    obj.InstallLocalFaxPrinter
else
    WScript.echo "Fax printer is ready to go."
end if
```

This is a good example of the functionality you can find by digging into the many objects that come with Windows, and it shows the kinds of scripts you can write to manage Windows computers so other users don't have to poke around with the Control Panel themselves.

(On Windows Vista, this script works only on the Business, Enterprise and Ultimate editions. The Home editions don't come with built-in faxing support, but, all versions of Windows 7 include the Faxing service.)

This page intentionally left blank

Index

Symbols & Numerics

& character, 54-55, 75

& operators
PowerShell, 646

+ operators, 54-55

64-bit installers, creating with IExpress, 398

A

AccountDisabled property (IADsUser object), 362

active connections, listing, 585

Active Directory
LDAP, 364-367
managing, 364
objects, 368
IADsO, 369
IADsO object, 369-370
IADsOU, 369
IADsOU object, 369-370
RootDSE, 368
RootDSE object, 368-369
X.500, 364-367

ActivePerl objects, 106
collections, 108-109
Perl Object Interface, 107-108
running Perl scripts in Windows Script Host, 106-107

ActivePython objects, 109-110

ActiveState Website, 15

Add method (BodyPart collections, 247

AddAttachment method (CDO.Message objects), 238-239

AddBodyPart method (BodyPart objects), 248

adding
attachments to messages (CDO), 261
drive mappings, 218-219
images to HTML messages (CDO), 262-263

AddPrinterConnection methods (WSHNetwork objects)
arguments
LocalName arguments, 208
Password arguments, 209
RemoteName arguments, 209
UpdateProfile arguments, 209
UserName arguments, 209
DOS session printers, redirecting, 225-226
scripts, printing from, 229-230

AddRelatedBodyPart method (CDO.Message objects), 239
constant values, 241

AddWindowsPrinterConnection methods (WSHNetwork objects), 209
arguments, PrinterPath arguments, 209, 224
network printers, connecting to, 223-224

administrative tools (Windows XP), 736-738

Administrator account, ADSI security, 328

ADO (ActiveX Data Objects), 242, 250

ADSI (Active Directory Scripting Interface), 319-320
collections, 339-340
common uses for, 320
containers, 330-332
directories, security, 328-330
Exchange (Microsoft), 364
EzAD Scriptomatic tool, 372
IIS (Internet Information Services), 364
LDAP provider, 364, 366-367

IADsO object, 369-370
IADsOU object, 369-370
RootDSE object, 368-369
limitations of, 321
multiple inheritance, 324
objects, 322-324
creating, 325-328
directory security, 328, 330
IADs object, 333-336
IADsCollection object, 336-338
IADsComputer object, 340
IADsComputerOperations object, 340
IADsContainer object, 336-338
IADsDomain object, 342-344
IADsFileService object, 345-346
IADsFileServiceOperations object, 345-346
IADsFileShare object, 347-348
IADsGroup object, 349-350
IADsMembers object, 350
IADsNamespaces object, 351
IADsPrintJob object, 351-352, 354
IADsPrintJobOperations object, 351-354
IADsPrintQueue object, 354-357
IADsPrintQueueOperations object, 354-357
IADsPrintServiceOperations object, 357, 360
IADsResource object, 346
IADsService object, 357-360
IADsServiceOperations object, 360-361
IADsSession object, 361-362
IADsUser object, 362-363
leaves, 330
Microsoft documentation, 332
multiple inheritance, 324-325
WinNT provider, 332
scripts, developing, 370-371
WSH (Windows Script Host), 321

AdsPath property (IADs object), 334

aliases, 450, 612

altering script variable values, Script Debugger (Windows), 45

AppActivate method (WScript.Shell object), 183-184

appendChild methods (IXMLDOMNode object), 174

applets, Control Panel, 546-547

argument substitution, 496-497

arguments, 31-32
batch file command-line, 726
delayed expansion, 511-513
editing, 498-499
validating, 519
checking (batch files), 519
CMD command-line processing, 454
CMD shell, separating, 454
command-line arguments
extracting named arguments, 387
Named collection, 386-387
processing, 386
processing named arguments, 386
processing unnamed arguments, 389
retrieving, 113-114
Unnamed collection, 389
Force arguments (WSHNetwork objects), 210-211
LocalName arguments (WSHNetwork objects), 208-210
multiple arguments, processing, 503-506
multiple, batch file processing, 503-506
Name arguments (WSHNetwork objects), 210-211
Named collection (command-line arguments)
Count method, 386
Exists method, 387
Items property, 386
Length property, 386
Password arguments (WSHNetwork objects), 209-210
PrinterPath arguments (WSHNetwork objects), 209, 224
RemoteName arguments (WSHNetwork objects), 209-210
substituting (batch files), 496
unnamed arguments, processing for WSF files, 389-390
Unnamed collection (command-line arguments)
Count method, 389
Items property, 389
Length property, 389

UpdateProfile arguments (WSHNetwork objects), 209-211
UserName arguments (WSHNetwork objects), 209-210

Arguments properties (WshShortcut object), 193

Arguments properties (WshUrlShortcut object), 193

arithmetic operators (VBScript variables), 54-55, 717

arithmetic operators (VBScript), 55

arrays, 89-91
PowerShell, 632-636
comparisons, 640-643
values, extracting, 636-637
VBScript, 89-91

assigning
drives, pushd command (CMD), 518
logon scripts with Group Policy, 418-421
paths, 134
random drive letters in batch files, 518
user profile logon scripts, 416, 418

assignment operators (PowerShell), 648

assoc command (CMD), 462

async properties (DOMDocument object), 170

AtEndOfLine properties (TextStream object), 151

AtEndOfStream properties (TextStream object), 151, 162

attachments, adding to messages (CDO), 261

Attachments property (CDO.Message objects), 236

attrib command (file management tools), 557-558
attributes, setting/clearing, 558-559
finding hidden files, 558

attrib command-line tool, 557-559

attribute names (LDAP), 365

attributes
file attribute values, 141
changing, 143-144
testing, 142
folder attribute values, 141
changing, 143-144
testing, 142
multiple attributes, testing, 143
setting/clearing (attrib command), 558-559

attributes properties (IXMLDOMNode object), 173, 175

Attributes properties (Scripting.File object), 145

authentication, WMI, 295-296
impersonation, 297-298
privileges, 298-299

AUTOEXEC.NT file, configuring NTVDM, 535-536

AutoGenerateTextBody property (CDO.Message objects), 236

automatic conversion (VBScript), 57

automatic scripting
messages, sending to Event Log, 423-425
messages, printing, 425
results, summarizing, 425-427
unattended scripts, writing, 421-423
scheduling, 421-431
unattended scripts
controlling logged information, 423
creating, 421-423

automatically assigned addresses, resetting, 573

automatically running scripts, 40

automation (objects), GetObject function, 99-100

AutoRun settings (CMD), 455

AutoUnlockInterval property (IADsDomain object), 343

AvailableSpace properties (Scripting.Drive object), 136

B

backups
creating with xcopy, 555
making (xcopy command-line programs), 555

unattended backups (xcopy command-line programs), 556
BannerPage property (IADsPrintQueue object), 354
batch files, 491
argument substitution, 496-497
arguments
checking, 519
editing, 498-499
expressions, 726
substituting, 496
validating, 519
command-line options, processing, 515-517
commands, 494-495, 738-739
conditional processing
extended if command, 501-503
if command, 499-500
creating, 492-494
delayed expansion, 511-513
echoing, 495-496
environment variables, 506-508
exit status, checking, 501
for command, 508-511
delayed expansion, 511-512
for loops, 510
if command (CMD), conditional processing, 499-501
information, displaying, 495-496
input, prompting for, 514-515
installers, creating with IExpress, 398-400
keeping log files, 519-520
log files, maintaining, 519-520
multiple arguments, processing, 503-506
network mappings, 518
deleting previous mappings, 518
managing, 518
UNC pathnames, 518
numerical calculations, performing, 481
privileges, 493
programming, 494-495
prompting for input, 514
random drive letter, assigning, 518
running scripts from, 39
script files, running, 39
storing, 492
subroutines, 513-514
UNC pathnames, 518
versus scripting, 13
versus Windows scripts, 13
BCC property (CDO.Message objects), 236
Big-Endian format, 365
binary files
BMP image data, reading, 164-167
MP3 tag data, reading, 166-167
reading, 163-164
bitwise mathematics, 143
BMP image data, reading, 164-165
BodyPart collections (CDO email components)
CDO.Message objects, 246
methods
Add method, 247
Delete method, 247
DeleteAll method, 247
properties
Count property, 246
Item property, 246
BodyPart objects (CDO email components), 232, 247
methods
AddBodyPart method, 248
GetEncodedContentStream() method, 248
SaveToFile method, 248
properties
BodyPart property, 247
Charset property, 247
ContentMediaType property, 247
ContentTransferEncoding property, 248
Fields property, 248
Filename property, 248
Parent property, 248
BodyPart property (CDO.Message objects), 236
BodyParts property (BodyPart objects), 247
Boolean value constants (VBScripts), 52
break command, PowerShell, 660-661
breakpoints, 45

breakpoints, setting in ISE, 709
BuildPath method (Scripting.FileSystemObject object), 125, 130
built-in commands (CMD)
@command, 462
assoc command, 462
call command, 463
cd command, 463
chdir command, 463
cls command, 463
command extensions, 475-476
copy command, 463
date command, 465
del command, 465-466
dir command, 466
listing files, 476-480
echo commands, 466
endlocal command, 467, 472
erase command, 467
exit/B command, 467
for command
delayed expansion, 511-512
for loop, 510
numerical for loops, 486
parsing text, 487-488
processing directories, 486
processing multiple items, 508-509
scanning files, 483-484
variables, 485-486
ftype command, 467-468
goto command, 468, 502
if command, 468, 499, 502
conditional processing, 482-483, 499-501
extended testing, 503
md command, 468-469
mkdir command, 469
move command, 469
path command, 470
pause command, 470
popd command, 470
prompt command, 470
pushd command, 470
assigning drives, 518
rd command, 471
rem command, 471
rename command, 471
rmdir command, 471
set commands, 471
batch files, performing numerical calculations, 481
setting variables, 480
setlocal command, 472
shift command, 472
start command, 473-474
time command, 474
title command, 474
type command, 474
ver command, 475
verify on/off command, 475
vol command, 475
Windows XP, 738-739
built-in functions (VBScripts), 720
BusinessCategory property (IADsOU object), 369

C

cacls command (file management tools), 559-561
file/folder privacy, 563
permissions
checking, 562
granting, 562
cacls command-line tool, 559-563
call command (CMD), 463, 513-514
Call Stack window, viewing, 46-47
case insensitive searching (findstr command-line program), 549
case sensitivity, Jscript objects, 104
casts, 625, 649-650
CC property (CDO.Message objects), 236
cd command (CMD), 463
CDO (Collaboration Data Objects), 232
email
routing through SMTP servers, 264
sending email attachments, 261
sending from scripts, 232-235
specifying delivery servers, 263
specifying subjects/recipients, 263
email components
BodyPart collections, 246-247
BodyPart objects, 232, 247-250
CDO.Configuration, 233

CDO.Message objects, 232-246
Field collections, 243-246, 249-250
Field objects, 233, 243
Fields collections, 242-243
messages
attachments, adding, 261
creating, 257
delivery server, specifying, 263-265
HTML, sending, 259
images, including with HTML messages, 262-263
multiformat, sending, 260
program output, sending, 258
recipients, specifying, 263
sending, 256, 265-266, 268, 270
sending multiformat messages, 260
sending program output messages, 258
sending text file messages, 258
sending text string messages, 257-258
VBScript constant definitions
Website, 241
Web pages, sending, 259-260

CDO.Configuration (CDO email components), 233

CDO.Configuration object, 250
LoadFrom method, cdoConfigSource constant, 253
methods, 251-256
properties, 251

CDO.Message objects (CDO email components), 232-235
BodyPart collections, 246
Field collection values, 244-246, 249-250
methods
AddAttachment method, 238-239
AddRelatedBodyPart method, 239-241
CreateMHTMLBody method, 240-242
Send method, 240
properties
Attachments property, 236
AutoGenerateTextBody property, 236
BCC property, 236
BodyPart property, 236
CC property, 236
Configuration property, 236
DSNOptions property, 236, 240-241
Fields property, 237
From property, 237
HTMLBody property, 237
HTMLBodyPart property, 237
MDNRequested property, 237
MIMEFormatted property, 237
Organization property, 238
ReplyTo property, 238
Sender property, 238
Subject property, 238
TextBody property, 238
TextBodyPart property, 238
To property, 238

cdoConfigSource constants, 253

cdoProtocolsAuthentication constants, 254

cdoSendUsing constants, 254

cdoTimeZoneId constants, 254-256

certificates, obtaining for code signing, 410-411

ChangePassword method (IADsUser object), 363

ChangeStartMode methods (Win32_Service objects), 316

changing
file/folder attributes, 143-144
folder attribute values, 143-144
PATH, 457-458

Charset property (BodyPart objects), 247

chdir command (CMD), 463

checking
arguments (batch files), 519
files/folders, if command (CMD), 500
free space (drives), 138-139
permissions (cacls command), 562

child nodes, 169

childNodes properties (DOMDocument object), 171

childNodes properties (IXMLDOMNode object), 173

choosing Window script languages, 16

choosing a scripting language, 16
CIM (Common Information Model), 281-283
class names (ADSI objects), 325
Class property (IADs object), 334
classes, 95
classes (objects), 95
clearing DNS cache, ipconfig command (Windows XP networking utilities), 573
Close methods (TextStream object), 152
cls command (CMD), 463
CMD shell
arguments, 438-439
separating, 454
AutoRun settings, 455
batch files
argument editing, 498-499
arguments, validating, 519
command-line options, processing, 515-517
delayed expansion, 511-513
environment variables, 506-508
input, prompting for, 514-515
log files, maintaining, 519-520
multiple items, processing with for command, 508-511
random drive letter, assigning, 518
subroutines, 513-514
UNC pathnames, 518
built-in commands, 462-475, 738-739
command-line processing, 439-440
arguments, 454
commas, 454
console program input/output, 441-447
console windows, 442
copying/pasting in command prompt windows, 447-448
editing commands, 448-449
grouping commands with parentheses, 453-454
History list, 449
multiple commands, 452-453
name completion, 450
quotes, 454
runaway programs, stopping, 440
running, 435, 438
special characters, 454
command-line programs, killing, 440
commands
editing, 448-450
extensions, 439, 475-476
multiple, typing on one line, 452-453
configuring, 455
AutoRun, 455
environment variables, 456-461
console programs, 441-442
copying and pasting, 447-448
reading text, 191-193
console window, 442
dir command, 476-480
directory name completion, 451-452
elevated Command Prompt, opening, 436-437
enviroment variable substitution, 456
environment variables
default, setting, 461
PATH, 456-458
system-wide, 459-461
extended commands, 475-476
extensions, disabling, 439
for command, 483-486
text, parsing, 487-488
variables, 485
if command, 482-483
name completion, 450-451
network mappings, deleting, 518-519
options, 437-438
redirection, 443-447
running, 435
set command, 481-482
shortcut, creating, 436
special characters, escaping, 455
versus COMMAND shell, 434
cmdlets, PowerShell, 607, 609-610, 690-691
code signing, 409-410, 412, 415
certificate, obtaining, 410-411
scripts, signing, 412-413
signed scripts, requiring, 414
collaboration data objects. *See* CDO
collection objects, 96-97
ActivePerl, 108-109

ActivePython, 110
Count properties, 103
For Each loops, 103
Item methods, 103
JScript, 104-106
SWbemObjectSet, 305-306

collection system information, example WMI script, 312-313

collections, 96, 102-103, 339-340
ADSI (Active Directory Scripting Interface) collections, 339
Environment collection (Windows Scripting Host)
Count method, 198
extracting information, 198-199
Item properties, 197
Length properties, 197
managing settings, 199
Remove method, 198
for JScript, 104-106
IXMLDOMNamedNodeMap properties, 176
Perl, 108-109
Python, 110
VBScript, 67

Column properties (TextStream object), 151

columnar listings, creating, 478

COM (Component Object Model), 94

command extensions (CMD), 475-476

command options (Cscript), 33-36

command options (Wscript), 33, 35

command prompt window
arguments, 454
assigning drives, 518
checking arguments, 519
command-line processing, 435, 438-440
commas, 454
console program input/output redirection, 443-447
console window, 442
creating, 492-494
delayed expansion, 511-512
deleting network mappings, 518
editing commands, 448-449
environment variables, 506-508
for loops, 510
grouping commands with parentheses, 453-454
History list, 449
keeping log files, 519-520
managing network mappings, 518
processing command-line options, 515-517
processing multiple items, 503-509
prompting for input, 514
subroutines, 513-514

command shells
configuring, 706-707
scripts, editing, 707-708
objects, generating, 685-686
obtaining, 598-600
profiles, 617-618
running on remote computers, 712
security, 615-617
text, reading from files, 697-698
text, writing to files, 698
PowerShell
aliases, 612
cmdlets, 607, 609-610
command-line syntax, 604-607
command-line editing, 602-603
copying and pasting, 603-604
get-help command, 610-611

command-line arguments
named arguments
extracting, 387
processing, 386
processing for WSF files, 386-388
Named collection
Count method, 386
Exists method, 387
Items property, 386
Length property, 386
processing, 386
retrieving, 113-114
unnamed arguments, processing, 389
Unnamed collection
Count method, 389
Items property, 389
Length property, 389

command-line processing (CMD), 439-440
arguments, 454

commands
editing, 448-449
grouping with parentheses, 453-454
commas, 454
console programs
copying/pasting in command prompt windows, 447-448
input/ouput, 441-442
input/ouput redirection, 443-447
console windows, 442
History list, 449
multiple commands, 452-453
name completion, 450
quotes, 454
PowerShell, 700
exception handling, 702-703
hash tables, 703-704
modules, writing, 701
runaway programs, stopping, 440
running, 435, 438
special characters, 454

command-line programs, 543
findstr, 547-548
adding/removing information, 550
case insensitive searching, 549
literal string matching, 549
matching text with wildcards, 550-551
positional searching, 549
searching multiple files, 549
GUI shortcuts, 545
more, 552-553
running, 188-191
tree, 553
xcopy, 554
copying subdirectories, 554
copying updated files, 555-556
making backups, 555
unattended backups, 556

COMMAND.COM shell, 524
file-management tools, 557
attrib, 557-559
cacls, 559-563
general purpose shell programs
findstr, 547-552
more, 552-553
tree, 553-554
xcopy, 554-557
GUI shortcuts, 545
Control Panel, 546-547
management power tools
driverquery, 564
runas, 565
sc, 569-571
taskkill, 568-569
tasklist, 565-567
networking tools
ipconfig, 571-573
net, 574-583
netstat, 584-586
nslookup, 586-589
ping, 589-590
tracert, 591-592
versus CMD, 434

commands
assoc command (CMD), 462
attrib command (file management tools), 557-558
finding hidden files, 558
setting/clearing attributes, 558-559
batch file commands, 738-739
built-in, 730-733
for command modifiers, 733-734
set /a command operators, 734
built-in commands (CMD), 462, 465, 469, 473, 475
@command, 462
assoc command, 462
call command, 463
cd command, 463
chdir command, 463
cls command, 463
command extensions, 439, 475-476
copy command, 463
date command, 465
del command, 465-466
dir command, 466, 476-480
echo commands, 466
endlocal command, 467, 472
erase command, 467
exit/B command, 467
for command, 483-488, 508-512
ftype command, 467-468
goto command, 468, 502
if command, 468, 482-483, 499-503
md command, 468-469
mkdir command, 469
move command, 469
path command, 470
pause command, 470

popd command, 470
prompt command, 470
pushd command, 470, 518
rd command, 471
rem command, 471
rename command, 471
rmdir command, 471
set commands, 471, 480-481
setlocal command, 472
shift command, 472
start command, 473-474
time command, 474
title command, 474
type command, 474
ver command, 475
verify on/off command, 475
vol command, 475
cacls command (file management tools), 559-561
checking permissions, 562
file/folder privacy, 563
granting permissions, 562
call command (CMD), 463
cd command (CMD), 463
chdir command (CMD), 463
cls command (CMD), 463
CMD shell
dir, 476-480
for, 483-488
if, 482-483
set, 481-482
copy command (CMD), 463
date command (CMD), 465
del command (CMD), 465-466
dir command (CMD), 466
listing files, 476-480
DOS commands (Windows XP), 739-740
driverquery command (management power tools), 564
echo commands (CMD), 466
editing, 448-449
editing in CMD shell, 448-450
endlocal command (CMD), 467, 472
erase command (CMD), 467
exit/B command (CMD), 467
extended (CMD shell), 475-476
file management commands, 740-741
for command (CMD)
delayed expansion, 511-512
for loop, 510
numerical for loops, 486
parsing text, 487-488
processing directories, 486
processing multiple items, 508-509
scanning files, 483-484
variables, 485-486
ftype command (CMD), 467-468
goto command (CMD), 468, 502
grouping with parentheses, 453-454
if command (CMD), 468, 499, 502
conditional processing, 482-483, 499-501
extended testing, 503
input/output redirection formats, 729
ipconfig command (networking utilities), 571
examining/clearing DNS cache, 573
listing IP address information, 571-572
resetting automatically assigned addresses, 573
md command (CMD), 468-469
mkdir command (CMD), 469
move command (CMD), 469
multiple command formats, 729
net command (networking utilities), 574
net continue command (networking utilities), 574
net file command (networking utilities), 574
net help command (networking utilities), 575
net helpmsg command (networking utilities), 575
net localgroup command (networking utilities), 575
net pause command (networking utilities), 575
net print command (networking utilities), 575-576
net send command (networking utilities), 576
net session command (networking utilities), 576-577
net share command (networking utilities), 577

net start command (networking utilities), 578
net statistics command (networking utilities), 578
net stop command (networking utilities), 578
net use command (networking utilities), 579-581
net user command (networking utilities), 581, 583
net view command (networking utilities), 583-584
netstat command (networking utilities), 584
constant monitoring, 586
listing active connections, 585
listing open ports, 586
listing statistics, 586
nslookup command (networking utilities), 586
finding hostname IP addresses, 587-589
testing DNS servers, 589
path command (CMD), 470
pause command (CMD), 470
ping command (networking utilities), 589-590
popd command (CMD), 470
prompt command (CMD), 470
pushd command (CMD), 470
rd command (CMD), 471
rem command (CMD), 471
rename command (CMD), 471
rmdir command (CMD), 471
sc command (management power tools), 569
sc queryex command, 569-570
starting/stopping services, 570
sc queryex command (management power tools)
listing installed services, 569-570
set commands (CMD), 471
batch files, performing numerical calculations, 481
setting variables, 480
set/a commands, expression operators, 734
setlocal command (CMD), 472
shift command (CMD), 472
start command (CMD), 473-474
taskkill command (management power tools)
killing processes by program name, 569
killing processes with PID numbers, 568
killing user processes, 568
tasklist command (console programs), 441
tasklist command (management power tools), 565-567
TCP/IP, 743
time command (CMD), 474
title command (CMD), 474
tracert command (networking utilities), 591-592
type command (CMD), 474
VBScriptm, Wscript.Echo, 84-85
ver command (CMD), 475
verify on/off command (CMD), 475
vol command (CMD), 475

commas, CMD command-line processing, 454

comments, PowerShell, 622, 700

comparing
CMD shell and COMMAND shell, 434
properties and methods, 100-101
scripting and batch files, 13
scripting and compiled languages, 13

comparison operators (VBScript), 55-56, 717

comparisons, performing with arrays (PowerShell), 640-643

Compatibility tab (Properties dialog box), 532

compiled languages versus scripting languages, 13

completing PowerShell commands, 612

complex text files, creating, 157-159

Computer property (IADsSession object), 361

ComputerName properties (WSHNetwork objects), 208

ComputerPath property (IADsSession object), 361

conditional breakpoints, setting in ISE, 711

conditional processing
if command, 482-483
if command (CMD), 482-483, 499
checking for files and folders, 500
checking program success, 500-501

conditional processing (batch files)
extended if command, 501-503
if command, 499-500
exist option, 500

conditional statements (VBScript), 57
If-Then statements, 58-59
variations, 59
variations, If-End if statements, 59
variations, If-Then-Else statements, 59-60
Select Case statements, 61-62

ConenctTime property (IADsSession object), 361

CONFIG.NT (NTVDM), 532-535

Configuration property (CDO.Message objects), 236

configuring
AUTOEXEC.NT (NTVDM), 535
CMD, 455
AutoRun, 455
environment variable substitution, 456-458
environment variables, 459-461
drive mappings, 220-221
environment variables (NTVDM), 536
Font tab (NTVDM), 528
ISE, 706-707
Memory tab (NTVDM), 528-529
Miscellaneous Settings tab (NTVDM), 530-532
NTVDM, 525-526
AUTOEXEC.NT file, 535-536
Compatibility tab, 532
CONFIG.NT file, 532-535
environment variables, 536
Font tab, 528
Memory tab, 528
Miscellaneous Settings tab, 530-531
Properties tab, 526-528
Screen tab, 530
serial communications, 539

confirming
drives existence, 137-138
target drives, 137-138

connecting to network printers, 223-225

ConnectServer method, parameters, 291-292

console programs, 441-442
command prompt windows, copying/pasting in, 447-448
copying and pasting, 447-448
filters, 445
full-screen mode, 442
I/O redirection, 443-447
input/output, 441-442
input/output redirection, 443-447
standard error, 445
tasklist command, 441
text, reading, 191-193

console windows, 442

constant values
AddRelatedBodyPart method constant values, 241
CreateMHTMLBody method constant values, 242
DSNOptions property constant values, 240-241

constants
cdoConfigSource, 253
cdoProtocolsAuthentication constants, 254
cdoSendUsing constants, 254
cdoTimeZoneId constants, 254-256
PowerShell, 637
VBScript, 51-52, 722

constructors, PowerShell, 627

container objects, 96

containers, 96, 330-332

ContentMediaType property (BodyPart objects), 247

ContentTransferEncoding property (BodyPart objects), 248

continue command, PowerShell, 661

Continue method (IADsServiceOperations object), 360

ContinueService methods (Win32_Service objects), 316

Control Panel, running with COMMAND.COM shell, 546-547

Control panel (Windows XP), applets, 546-547

controlling logged information (unattended scripts), 423

converting strings to other types (VBScript), 74

copy command (CMD), 463

Copy method (Scripting.Folder object), 141, 145

CopyFile method (Scripting.FileSystemObject object), 125

CopyFolder method (Scripting.FileSystemObject object), 125, 135

CopyHere method (IADsCollection object), 337

copying
- folders, 135
- scripts to multiple computers, 406-408
- subdirectories (xcopy command-line programs), 554
- updated files (xcopy command-line programs), 555-556

CPL files, 546

Create method (IADsCollection object), 337-338

createCDataSection methods (DOMDocument object), 171

createComment methods (DOMDocument object), 171

createDocumentType methods (DOMDocument object), 171

createElement methods (DOMDocument object), 171

CreateMHTMLBody method (CDO.Message objects), 240
- constant values, 242

createProcessingInstruction methods (DOMDocument object), 171

CreateShortcut method (WScript.Shell object), 184

CreateTextFile method (Scripting.FileSystemObject object), 126

createTextNode methods (DOMDocument object), 172

CreatFolder method (Scripting.FileSystemObject object), 125, 132-134

CreatFullPath method (Scripting.FileSystemObject object), 133-134

creating
- ADSI objects, 325-328
- batch files, 492-494
- columnar listings, 478
- complex text files, 157-159
- folders, 132-135
- fully qualified pathnames, 130-131
- functions, 87
- HTML (Hypertext Markup Language), 179-181
- installation programs with IExpress, 395-398
  - *batch files, 398-400*
  - *UAC, 400-401*
  - *uninstall option, providing, 402*
- messages, CDO (Collaboration Data Objects), 257
- script files, 24-25
- script shortcuts, 39
- scripts to manage other computers, 404-405
- shortcuts, 183-186, 193-196
- text files, 157, 159
  - *inserting tabs, 157*
  - *Unix-compatible, 159*
- unattended scripts, 421-423
- user-friendly scripts, 376-377
- WSF files, 377-378, 390, 394
- XML files, 179-181

Cscript, 28-29
- command options, 33-36
- script files
  - *running, 30*
  - *running from batch files, 40*

saving, output redirection, 33
viewing, pipe mechanisms, 33
security, 41

CSLID (objects), 116

CurrentDirectory properties (WScript.Shell object), 182

currentTime property (RootDSE object), 368

CurrentUserCount property (IADsFileShare object), 347

D

Datatype property (IADsPrintQueue object), 354

date and time functions (VBScript), 75-78

date command (CMD), 465

Date value constants (VBScripts), 52

Date() functions (VBScript), 75-77, 79, 722

Date/Time constants (VBScript), 52

DateAdd functions (VBScripts), 75

DateAdd() function, 75-76

DateCreated properties (Scripting.File object), 145

DateCreated properties (Scripting.Folder object), 140

DateDiff() function, 76

DateDiff() functions (VBScripts), 76

DateLastAccessed properties (Scripting.File object), 140, 145

DCOM (Distributed COM)
security, WMI (Windows Management Interface), 294-300
Windows remote management, 283
on domain networks, 283-284
on workgroup networks, 284-287

debugging
scripts, 42-45
Call Stack windows, viewing, 46-47
Script Debugger (Windows), 43-45
tracing, 43
Wscript.Echo command, 84
VBScript, 42

default environment variables, 727-728

default printer, setting with WScript.Network object, 228

DefaultContainer property (IADsNamespaces object), 351

DefaultJobPriority property (IADsPrintQueue object), 354

defaultNamingContext property (RootDSE object), 368

defining
JavaScripts, 14
JScripts, 14
methods, 93, 100
objects, 10-11, 93
Perl, 15
properties, 93, 100
Python, 15
Ruby, 15
scripts, 10
VBScripts, 14
Windows Script Host, 11-12

del command (CMD), 465-466

delayed expansion, 511-513

Delete method (BodyPart collections), 247

Delete method (IADsCollection object), 338

Delete method (IADsContainer object), 338

Delete method (Scripting.Folder object), 141

Delete method (SWbemServices objects), 302, 307

Delete methods (Scripting.File object), 146

DeleteAll method (BodyPart collections), 247

DeleteFile method (Scripting.FileSystemObject object), 126, 132

DeleteFolder method (Scripting.FileSystemObject object), 126

deleting
drive mappings, 219-220
files, 132
network mappings with batch files, 518-519
network printer connections, 226-228
printer connections, 226-228

delivery server, specifying in messages (CDO), 263, 265

Dependencies property (IADsService object), 358

deploying scripts on network, 394

Description properties
ADsUser object, 362
IADsFileShare object, 348
IADsGroup object, 349
IADsO object, 369
IADsOU object, 369
IADsPrintJob object, 352
IADsPrintQueue object, 354
WshShortcut object, 194
WshUrlShortcut object, 194

DesktopInteract properties (Win32_Service objects), 315

developing ADSI (Active Directory Scripting Interface) scripts, 370-371

dir command (CMD), 466, 476-480
files, listing, 476-477
creating columnar listings, 478
listing hidden files, 480
paginating lists, 477
printing directory listings, 478
retrieving filename listings, 479
searching files, 477
sorting listings, 479

directories
Active Directory
LDAP, 364-367
managing, 364
objects, 368
X.500, 364-367
containers versus leaves, 330-331
PowerShell, 692-696
printing, 478
processing, for command (CMD), 486
user directories
managing, 319-321
security, 328, 330

directory name completion (CMD shell), 451-452

disabling
batch file echoing, 495
CMD extensions, 439
command extensions (CMD), 439
unsigned scripts, 408

displaying
information in batch files, 495-496
network user information, Wscript.Network objects, 212-214
printer information, 222-223
script function values, Script Debugger (Windows), 45
script properties, 23
text, Wscript.Echo command, 84

DisplayName properties (Win32_Service objects), 315

DisplayName property (IADsService object), 358

distinguishing containers from leaves, 331-332

Division property (IADsComputer object), 340

DN (distinguished name), 364

DNS cache, examining/clearing, 573

DNS servers, testing, 589

dnsHostName property (RootDSE object), 368

Do While statement, 63-65

documentation, ADSI (Active Directory Scripting Interface), 332

documentElement properties (DOMDocument object), 171

documenting functions (VBScripts), 70

domain networks
logon scripts, assigning through Group Policy, 418-421
remote management with WMI, 283-284

DOMDocument object (Windows Scripting Host), 169
methods
createCDataSection methods, 171
createComment methods, 171
createDocumentType method, 171
createElement method, 171
createProcessingInstruction method, 171
createTextNode method, 172
getElementsByTagName method, 172
load method, 172
loadXML method, 172
Save method, 172
selectNodes method, 172
selectSingleNode method, 172
properties
async properties, 170
childNodes properties, 171
documentElement properties, 171
parseError properties, 171
xml properties, 171

DOS commands (Windows XP), 739-740

DOS printer sessions, redirecting, 225-226

dot-sourcing, PowerShell, 668

downloading
pvk2pvc.exe, 410
Scriptomatic, 310
signtool.exe, 410

drive mappings
adding, 218-219
configuring, 220-221
deleting, 219-220
listing, 214-218
managing, 214
scripts, 220-221

Drive properties (Scripting.Folder object), 140

DriveExists method (Scripting.FileSystemObject object), 126

DriveLetter properties (Scripting.Drive object), 136

driverquery command (management power tools), 564

drives
assigning, 518
confirming, 137-138
existence, confirming, 137-138
free space, checking, 138-139
locating, 138

Drives property (Scripting.FileSystemObject object), 125

DriveType properties (Scripting.Drive object), 136

DSNOptions property (CDO.Message objects), 236
constant values, 240-241

DTD (document type definition), 168

DTMF (Distributed Management Task Force), 281

dynamic environment variables, 728

dynamic methods, 306

dynamic properties, 306

E

echo commands (CMD), 466

echoing, 495-496

editing
arguments (batch files), 498-499
batch file arguments, 498-499
CMD commands, 448-450
commands, 448-449
environment variables, 507-508

editing tools, 27

elements (XML), 168

elevated command prompt, opening, 436-437

email
CDO (Collaboation Data Objects)
email components
BodyPart collections, 246-247
BodyPart objects, 232, 247-250
CDO.Configuration, 233
CDO.Message objects, 232-246
Field collections, 243-246, 249-250
Field objects, 233, 243
Fields collections, 242-243

messages
attachments, adding, 261
delivery server, specifying, 263, 265
HTML, sending, 259
images, including, 262-263
multiformat, sending, 260
program output, sending, 258
recipients, specifying, 263
sending, 265, 267-268, 270
subject, specifying, 263
text strings, sending, 257-258
web pages, sending, 259-260
messaging objects, CDO, 232-256
routing through SMTP servers, CDO (Collaboration Data Objects), 264
sending
CDO (Collaboration Data Objects), 232-235
from scripts, 231-232
HTML, 234
MIME (Multipart Internet mail Extensions), 234

enabling directory name completion (CMD), 451-452

enclosing multiple scripts in WSF files, 390

encoded scripts, 20

encryption
Script Encoder, 415
WMI, 295-296

endlocal command (CMD), 467, 472

enforcing script signing, 414

Enumerator objects (Jscript), 105

EnumNetworkDrives() methods(WSHNetwork objects), 209, 214-217

EnumPrinterConnections() methods(WSHNetwork objects), 209, 222-223

Environment collection (Windows Scripting Host)
extracting information, 198-199
managing settings, 199
methods
Count method, 198
Remove method, 198
properties
Item properties, 197
Length properties, 197

Environment properties (WScript.Shell object), 182-183, 196-197

environment variable substitution (CMD), 456
PATH
changing, 457-458
search paths, 456-457

environment variables, 506-507, 536, 727
batch files, 506-508
default, 461, 727-728
delayed expansion, 511-513
dynamic, 728
editing, 507-508
expressions, 727
for batch files
managing, 199
NTVDM, configuring, 536
PATH, 456-457
changing, 457-461
managing, 200-201
predefined, 727-728
retrieving, 198-199
system-wide, 459-461

erase command (CMD), 467

error handling, VBScript, 86-87

error status of batch files, verifying, 501

ErrorControl property (IADsService object), 358-359

escaping special characters (CMD shell), 454-455

Event Log messages, sending, 423-424
printing messages, 425
results, summarizing, 425, 427

examining DNS cache, ipconfig command (Windows XP networking utilities), 573

example scripts
WMI
printers, managing, 313
system information, collecting, 312-313

tasks, managing, 315-317
Windows service packs, monitoring, 314-315
WSF files, 390-394
exception handling (PowerShell), 702-703
throw command, 664
trap command, 662
Exchange (Microsoft), ADSI (Active Directory Scripting Interface), 364
Exec method (WScript.Shell object), 184
ExecMethod method (SWbemServices objects), 302
ExecQuery method (SWbemServices objects), 302-305
exist option (if command), batch file conditional processing, 500
Exists method (Named collection), 387
Exit Do statement, 65-66
exit do statements (VBScripts), 64-65
Exit For statements (VBScripts), 66
exit status, verifying, 501
exit/B command (CMD), 467
ExitCode properties (WshScriptExec object), 188
ExpandEnvironmentStrings method (WScript.Shell object), 184
expressions, 54
automatic conversions, 57
PowerShell, 638-639
VBScript, syntax, 716
extended commands (CMD shell), 475-476
extended if command, 501-503
extending built-in functions, 88
Extensible Markup Language. ***See*** **XML**
extensions
CMD shell, disabling, 439
for script files, 19-20
extracting
environment variables, 198-199
information, Environment collection (Windows Scripting Host), 198-199
named arguments (command-line arguments), 387
EzAD Scriptomatic tool, 372

F

FAXCOMEx.FaxDocument object
methods, 273-274
properties, 271-273
faxes, sending from scripts, 271, 274-277
FaxNumber property (IADsO object), 369
FaxNumber property (IADsOU object), 369
Field collections (CDO email components)
BodyPart objects, 249-250
CDO.Message objects, 244-246
methods
Update method, 243
Field objects (CDO email components), 233, 243
Fields collection, 242-246
Fields property (BodyPart objects), 248
Fields property (CDO.Configuration objects), 251
nntpauthenticate field, cdoProtocolAuthentication constants, 254
sendusing field, cdoSendUsing constants, 254
smtpauthenticate field, cdoProtocolAuthentication constants, 254
timezoneid field, cdoTimeZoneId constants, 254-256
values, 252-253

Fields property (CDO.Message objects), 237

file attribute values, 141
changing, 143-144
testing, 142

file management commands, 740-741

file management tools (Windows XP)
attrib command, 557-558
finding hidden files, 558
setting/clearing attributes, 558-559
cacls command, 559-561
checking permissions, 562
file/folder privacy, 563
granting permissions, 562

file-management tools
running with COMMAND.COM shell, 557
attrib, 557-559
cacls, 559-563

FileExists method (Scripting.FileSystemObject object), 126

filename listings, retrieving, 479

Filename property (BodyPart objects), 248

files
attribute values, 141-143
attributes, changing, 143-144
batch files
assigning drives, 518
checking arguments, 519
creating, 492-494
delayed expansion, 511-512
deleting network mappings, 518
displaying information, 495-496
editing arguments, 498-499
environment variables, 506-508
for loops, 510
if command (CMD) conditional processing, 499-501
keeping log files, 519-520
managing network mappings, 518
performing numerical calculations, 481
processing command-line options, 515-517
processing multiple arguments, 503-506
processing multiple items, 508-509
programming, 494-495
prompting for input, 514
subroutines, 513-514
substituting arguments, 496
versus Windows scripts, 13
binary files, reading, 163-167
checking for, if command (CMD), 500
deleting, 132
hidden files, listing, 558, 480
listing, 476-480
dir command (CMD), 476-480
multiple files, searching (findstr command-line programs), 549
privacy (cacls command), 563
reading, 149
reading text from, 152-153
renaming, 132, 146
scanning for, 146-149, 483-484
searching, dir command (CMD), 477
stdin files, 159-161
stdout files, 159-162
text files
creating, 157-159
inserting tabs, 157
writing Unix-compatible text files, 159
updated files, copying (xcopy command-line programs), 555-556
writing text to, 154-157
WSF files
creating, 377-378, 390, 394
extracting named arguments (command-line arguments), 387
formats, 379-383
processing command-line arguments, 386
processing named arguments (command-line arguments), 386
processing unnamed arguments (command-line arguments), 389
providing online help, 384-385
XML tags, 379-383

Files properties (Scripting.Folder object), 140

FileSystem properties (Scripting.Drive object), 137

FileSystemObject object, 130

Filter property
IADsCollection object, 336
IADsContainer object, 336
IADsDomain object, 343
IADsMembers object, 350
IADsOU object, 369

filters, 161, 445

finding
hidden files, 558
hostname IP addresses, 587-589

findstr (command-line programs), 547-548
adding/removing information, 550
case insensitive searching, 549
literal string matching, 549
multiple files, searching, 549
positional searching, 549
text, matching with wildcards, 550-551

firstChild properties (IXMLDOMNode object), 173

flow control (VBScript), 57
Do While statement, 63-65
Exit Do statement, 65-66
For…Each statement, 68
For…Next statement, 66-67
If…Then statement, 58-61
PowerShell, 653-661
Select Case statement, 61-63

folder attribute values, 141
changing, 143-144
testing, 142

FolderExists method (Scripting.FileSystemObject object), 126

folders
attribute values, 141-144
checking for, 500
copying, 135
creating, 132-135
privacy (cacls command), 563

Font tab (NTVDM), configuring, 528

for command (CMD), 483-484, 486
delayed expansion, 511-512
directories, processing, 486
files, scanning, 483-484
for loop, 510
multiple items, processing, 508-509
numerical for loops, 486
PowerShell, 655-656
text, parsing, 487-488
variables, 485-486

For Each loops (collection objects), 103

For…Each statement, 68

For…Next statement, 66-67

Force arguments (WSHNetwork objects), 210-211

forcedos compatibility program (Windows XP), 524-525

formatting
commands, 729
WQL queries, 304

for…in statements (Jscript), 105

For…Next statements (VBScripts), 66-68

free space (drives), checking, 138-139

FreeSpace properties (Scripting.Drive object), 137

From property (CDO.Message objects), 237

fso. *See* FileSystemObject

ftype command (CMD), 467-468

full-screen mode (console programs), 442

FullName properties (WshShortcut object), 194

FullName properties (WshUrlShortcut object), 194

FullName property (IADsUser object), 362

fully qualified pathnames, 130
creating, 130-131

functions
built-in functions, extending, 88
creating, 87
GetObject function, 99-100

PowerShell
dot-sourcing, 668
pipeline functions, 671-672
Print Screen function (Windows XP), 538
VBScript, 720-722
built-in functions, 720
calling, 69
creating, 87-88
date and time, 75-78, 722
documenting, 70
InputBox(), 82-84
MsgBox(), 79-82
string-manipulation functions, 71-74
Time() functions, 75
Ucase functions, 68
syntax, 70
WMI, 281

G

general-purpose shell programs, running with COMMAND.COM shell
findstr, 547-552
more, 552-553
tree, 553-554
xcopy, 554-557

Get method (IADs object), 334-335

get-help command, PowerShell, 610-611

GetAbsolutePathName method (Scripting.FileSystemObject object), 126

GetBaseName method (Scripting.FileSystemObject object), 127, 131-132

GetDrive method (Scripting.FileSystemObject object), 127

GetDriveName method (Scripting.FileSystemObject object), 127

getElementsByTagName methods (DOMDocument object), 172

GetEncodedContentStream() method (BodyPart objects), 248

GetEx method (IADs object), 335

GetExtensionName method (Scripting.FileSystemObject object), 127

GetFile method (Scripting.FileSystemObject object), 127

GetFileName method (Scripting.FileSystemObject object), 127

GetFolder method (Scripting.FileSystemObject object), 127

GetInfo method (IADs object), 335

GetInfoEx method (IADs object), 335

getNamedItem methods (IXMLDOMNamedNode object), 176

GetObject function, 99-100

GetObject method (IADsCollection object), 338

GetParentFolderName method (Scripting.FileSystemObject object), 127

GetSpecialFolder method (Scripting.FileSystemObject object), 128

GetTempName() method (Scripting.FileSystemObject object), 128

global scope, 91

goto command (CMD), 468, 502

granting permissions (cacls command), 562

graphical user interface, 434

Group Policy logon scripts, assigning on domain networks, 418-421

grouping
CMD commands with parentheses, 453-454
commands with parentheses, 453-454

Groups property (IADsUser object), 362

GUI programs, 744-745

GUI shortcuts (command-line programs), 545-547

GUID property (IADs object), 334

H

hasChildNodes methods (IXMLDOMNode object), 174

hash tables, PowerShell, 650-653, 703-704

help, providing for WSF files, 384-385

here-strings, PowerShell, 629

hidden files
finding, attrib command (file management tools), 558
listing, dir command (CMD), 480

Hints property (IADsCollection object), 337

History list (CMD), 449

HomeDirectory property (IADsUser object), 362

host (WSH), 11-12

HostComputer property
IADsFileShare object, 348
IADsPrintQueue object, 354
IADsService object, 358

HostPrintQueue property (IADsPrintJob object), 352

hotfixes, monitoring, 313-315

Hotkey properties (WshShortcut object), 194

HTML (Hypertext Markup Language)
creating, 179-181
email, sending, 234
reading/writing, 167, 172, 176-177

HTMLBody property (CDO.Message objects), 237

HTMLBodyPart property (CDO.Message objects), 237

I

I/O redirection, 443-447

IADs object (ADSI), 333-336
methods
Get method, 334-335
GetEx method, 335
GetInfo method, 335
GetInfoEx method, 335
Put method, 335
PutEx method, 335-336
SetInfo method, 336
properties
AdsPath property, 334
Class property, 334
GUID property, 334
Name property, 334
Parent property, 334
Schema property, 334

IADsCollection object (ADSI), 336-338
methods
CopyHere method, 337
Create method, 337-338
Delete method, 338
GetObject method, 338
MoveHere method, 338
properties
Count property, 336
Filter property, 336
Hints property, 337

IADsComputer object (ADSI), 340-342
properties
Division property, 340
OperatingSystem property, 340
OperatingSystemVersion property, 340
Owner property, 340
Processor property, 340
ProcessorCount property, 340

IADsComputerOperations object (ADSI), 340-342

IADsContainer object (ADSI), 336-338
collections, 339-340
methods
CopyHere method, 337
Create method, 337-338
Delete method, 338
GetObject method, 338
MoveHere method, 338
properties
Count property, 336

Filter property, 336
Hints property, 337

IADsDomain object (ADSI), 342-344
methods, SetInfo method, 344
properties
AutoUnlockInterval property, 343
Filter property, 343
IsWorkgroup property, 343
LockoutObservationInterval property, 343
MaxBadPasswordsAllowed property, 343
MaxPasswordAge property, 343
MinPasswordAge property, 343
MinPasswordLength property, 343
PasswordAttributes property, 343-344
PasswordHistoryLength property, 344

IADsFileService object (ADSI), 347
properties
MaxUserCount property, 345
Resources property, 345
Sessions property, 346

IADsFileServiceOperations object (ADSI), 347
properties
MaxUserCount property, 345
Resources property, 345
Sessions property, 346

IADsFileShare object (ADSI), properties
CurrentUserCount property, 347
Description property, 348
HostComputer property, 348
MaxUserCount property, 348
Name property, 348
Path property, 348

IADsGroup object (ADSI), 349-350
properties
Description property, 349
IsMember() property, 349
Member() property, 349
Methods property, 349
Name property, 349
Remove property, 349

IADsMembers object (ADSI), 350

IADsNamespaces object (ADSI), 351

IADsO object (Active Directory), properties
Count property, 369
Description property, 369
FaxNumber property, 369
Filter property, 369
LocalityName property, 369
Name property, 369
Parent property, 369
PostalAddress property, 370
SeeAlso property, 370
TelephoneNumber property, 370

IADsOU object (Active Directory), properties
BusinessCategory property, 369
Count property, 369
Description property, 369
FaxNumber property, 369
Filter property, 369
LocalityName property, 369
Name property, 369
Parent property, 369
PostalAddress property, 370
SeeAlso property, 370
TelephoneNumber property, 370

IADsPrintJob object (ADSI), 351-354

IADsPrintJobOperations object (ADSI), 351-354

IADsPrintQueue object (ADSI), 354-357
properties
BannerPage property, 354
Datatype property, 354
DefaultJobPriority property, 354
Description property, 354
HostComputer property, 354
Location property, 354
Model property, 354
Name property, 355
PrintDevices property, 355
PrinterPath property, 355
PrintProcessor property, 355
Priority property, 355
Starttime property, 355
UntilTime property, 355

IADsPrintQueueOperations object (ADSI), 354-357

IADsResource object (ADSI), 346

IADsService object (ADSI), 357-361
methods, SetInfo method, 359
properties
Dependencies property, 358
DisplayName property, 358
ErrorControl property, 358-359
HostComputer property, 358
LoadOrderGroup property, 358
Name property, 358
Path property, 358
ServiceAccountName property, 358
ServiceAccountPath property, 358
ServiceType property, 358
StartType property, 359
StartupParameters property, 359
Version property, 359

IADsServiceOperations object (ADSI), 357-361
methods
Continue method, 360
Pause method, 360
SetPassword method, 360
Start method, 361
Stop method, 361
properties, Status property, 360

IADsSession object (ADSI), 361-362
properties
Computer property, 361
ComputerPath property, 361
ConnectTime property, 361
IdleTime property, 361
User property, 361
UserPath property, 361

IADsUser object (ADSI)
methods
ChangePassword method, 363
SetInfo method, 363
SetPassword method, 363
properties
AccountDisabled property, 362
Description property, 362
FullName property, 362
Groups property, 362
HomeDirectory property, 362
IsAccountLocked property, 362
LastLogin property, 363
LastLogoff property, 363
Profile property, 363

IconLocation properties (WshShortcut object), 194

IdleTime property (IADsSession object), 361

IExpress, creating installation programs, 395-398
batch files, 398-400
UAC, 400-401
uninstall option, providing, 402

if command (CMD), 468, 482-483, 499-502
conditional processing, 482-483, 499
checking for files and folders, 500
checking program success, 500-501
extended testing, 503
PowerShell, 653

If…Then statement, 58-61

If…End if statements (VBScripts), 59

IF…Then statements (VBScripts), 58-59
variations, 59
If…End If statements, 59
If…Then…Else statements, 59
If…Then…ElseIf statements, 60

If…Then…Else statements (VBScripts), 59-60

IIS (Internet Information Services), ADSI (Active Directory Scripting Interface), 364, 374

images, adding to HTML messages (CDO), 262-263

impersonation, WMI options, 295-298

information
extracting, Environment collection (Windows Scripting Host), 198-199
saving, Windows Registry, 203-205

InputBox() functions (VBScripts), 82-84

insertBefore methods (IXMLDOMNode object), 174

inserting tabs into text files, 157

installation programs
creating with IExpress, 395-398
batch files, 398-400
UAC, 400-401
uninstall option, providing, 402

installed services, listing, 569

installing code-signing certificate, 411

instances (objects), 95

Instances method (SWbemServices objects), 307

InstancesOf method (SWbemServices objects), 303

InStr function (VBScript string-manipulation functions), 71-72

InStr() function, 71-72

InStrRev() function, 71-72

interface, 94

interpreters, 14

InterrogateService methods (Win32_Service objects), 316

IP addresses
hostname IP addresses, finding, 587-589
listing information, 571-572

ipconfig command (networking utilities), 571
automatically assigned addresses, resetting, 573
DNS cache, examining/clearing, 573
IP address, listing information, 571-572

ipconfig command-line tool, 571-573

IsAccountLocked property (IADsUser object), 362

ISE (Integrated Scripting Environment), 705
breakpoints, setting, 709
conditional breakpoints, setting, 711
configuring, 706-707
scripts, editing, 707-708

IsMember property (IADsGroup object), 349

IsReady properties (Scripting.Drive object), 137-138

IsRootFolder properties (Scripting.Folder object), 140

IsWorkgroup property (IADsDomain object), 343

Item method (collection objects), 103

item methods (IXMLDOMNamedNode object), 176

Item properties
Environment collection, 197
BodyPart collections, 246
Fields collections, 243
Named collection, 386
Unnamed collection, 389

IXMLDOCNode object (Windows Scripting Host), 176

IXMLDOMNamedNode object (Windows Scripting Host), 176

IXMLDOMNode object (Windows Scripting Host), 173
methods
appendChild method, 174
hasChildNodes method, 174
insertBefore method, 174
removeChild method, 174
repalceChild method, 174
selectNodes method, 174
selectSingleNode method, 174
setAttribute method, 175
nodeType values, 175
properties
attributes properties, 173, 175
childNodes properties, 173
firstChild properties, 173
lastChild properties, 173
nextSiblings properties, 173
nodeName properties, 173
nodeType properties, 173
nodeTypeString properties, 174
nodeValue properties, 174
ownerDocument properties, 174
previousSibling properties, 174
xml properties, 174

J-K

JavaScript, defining, 14

joining strings
& character, 54-55, 75
+ operators, 54-55

JScript, 13
defining, 14
interpreters, 14
objects, 104
collections, 104-106
Enumerator objects, 105
for…in statements, 105
WScript, 104

JSE extensions (script files), 20

keys (Registry), 202

killing
command-line programs, 440
processes, PID numbers (taskkill command), 568-569
user processes (taskkill command), 568

L

languages
interpreters
JScript language interpreter, 14
VBScript language interpreter, 14
script files
extensions, 19-20
WSC (Windows Script Component) files, 23
WSF (Windows Script Files), 21-22
WSH (Windows Script Host) settings, 23
script languages
ActivePerl, 106-109
ActivePython, 109-110
choosing, 16
JavaScripts, defining, 14
JScripts, 13-14, 104-106
Perl, 15
Python, defining, 15
Ruby, defining, 15
VBScripts, 13, 49, 98-99, 102, 713
VBScripts, constants, 51-53, 714
VBScripts, debugging, 42
VBScripts, defining, 14
VBScripts, functions, 68-84, 720, 722
VBScripts, interpreters, 14
VBScripts, omitted VBA features, 723-724
VBScripts, procedures, 87-89
VBScripts, program structures, 714
VBScripts, sample scripts, 16-19
VBScripts, statements, 57-59, 61-68, 718
VBScripts, syntax, 714
VBScripts, variables, 50-57, 89-92, 714, 716-718
WScript, 111-114

lastChild properties (IXMLDOMNode object), 173

LastLogin property (IADsUser object), 363

LastLogoff property (IADsUser object), 363

LDAP, 364-370
attribute names, 365
DNs, 365
RDNs, 365
Website, 367

leaves, 330
distinguishing from containers, 331-332
versus containers, 330-331

Left() function, 72-73

Length properties (Environment collection), 197

length properties (IXMLDOMNamedNode object), 176

Length property (Named collection), 386

Length property (Unnamed collection), 389

limitations of ADSI, 321

Line properties (TextStream object), 151

listing
active connections, 585
drive mappings, 214-218

files, 476-480
hidden files, 480
installed services, 569-570
IP address information, 571-572
open ports, 586
statistics, 586

literal strings, matching (findstr command-line programs), 549

literal values, PowerShell, 625-626

literals, 51

Little-Endian format, 365

Load methods (CDO.Configuration objects), 251

load methods (DOMDocument object), 172

LoadFrom method (CDO.Configuration objects), cdoConfigSource constants, 253

LoadOrderGroup property (IADsService object), 358

loadXML methods (DOMDocument object), 172

local computers, connecting to with WMI, 294

LocalityName property (IADsO object), 369

LocalityName property (IADsOU object), 369

LocalName arguments (WSHNetwork objects), 208, 210

locating drives, 138

Location property (IADsPrintQueue object), 354

LockCount property (IADsResource object), 346

LockoutObservationInterval property (IADsDomain object), 343

log files
keeping (batch files), 519-520
maintaining, 519-520
messaging, 425
printing, 425

LogEvent method (WScript.Shell object), 184

logged information, controlling on unattended scripts, 423

logical operators (VBScript variables), 54, 56, 718

logoff program (Windows XP), 741

logon scripts, 213, 416
assigning through Group Policy, 418-421
group policy scripts, 420
user profile logon scripts, 416-418
writing, 214

looping statements, 57
Do While, 63-65
Exit Do, 65-66
For…Each, 68
For…Next, 66-67
If…Then, 58-61

loops
For Each loops (collection objects), 103
numerical for loops, 486

M

macros, 450

management power tools (Windows XP), 563
driverquery command, 564
running with COMMAND.COM shell
driverquery, 564
runas, 565
sc, 569-571
taskkill, 568-569
tasklist, 565-567
sc command, 569
sc queryex command, 569-570
starting/stopping services, 570
taskkill command, 568
killing processes by program name, 569
killing processes with PID numbers, 568
killing user processes, 568
tasklist command, 565-567

managing
Active Directory, 364
computers, creating scripts, 404-405

Environment settings, Environment collection (Windows Scripting Host), 199
environment variables, 199
PATH, 200-201
MS-DOS programs, 540
network connections, 207-211
drive mappings, adding, 218-219
drive mappings, configuring, 220-221
drive mappings, deleting, 219-220
drive mappings, listing, 214-218
Windows Script Host, 207-208
network mappings (batch files), deleting previous mappings, 518
printers
connections, 207-208, 223-228
DOS printer sessions, redirecting, 225-226
example WMI script, 313
information, displaying, 222-223
WMI (Windows Management Interface), 313
services, WMI (Windows Management Interface), 315-317
tasks, example WMI script, 315-317
user directories, 319-321

MapNetworkDrive methods(WSHNetwork objects)
arguments, 210
drive mappings
adding, 218-219
scripts, 220-221

mapping printers in NTVDM, 538

markup tags, XML (Extensible Markup Language), 168-169

matching
literal strings (findstr command-line programs), 549
text with wildcards (findstr command-line programs), 550-551

MaxBadPasswordsAllowed property (IADsDomain object), 343

MaxPasswordAge property (IADsDomain object), 343

MaxUserCount property (IADsFileService object), 345

MaxUserCount property (IADsFileServiceOperations object), 345

MaxUserCount property (IADsFileShare object), 348

md command (CMD), 468-469

MDNRequested property (CDO.Message objects), 237

Member() property (IADsGroup object), 349

members, 626

Memory tab (NTVDM), configuring, 528-529

messages
attachments, adding, 261
CDO, sending, 265-270
creating, 257
delivery server, specifying, 263-265
Event log, sending to, 423-425
HTML
images, including, 262-263
sending, 259
multiformat, sending, 260
program output messages, sending, 258
recipients, specifying, 263
sending, 256, 265-270
subject, specifying, 263
text file messages, sending, 258
text string messages, sending, 257-258
web pages, sending, 259-260

messaging objects, CDO, 232, 234-235
BodyPart object, 247-250
BodyParts collection, 246-247
CDO.Configuration object, 250-256
CDO.Message object, 236-242
Fields, 242-246

methods, 93
AddPrinterConnection (WSHNetwork objects), 208
redirecting, 225-226
LocalName arguments, 208
Password arguments, 209
printing from scripts, 229-230
RemoteName arguments, 209
UpdateProfile arguments, 209
UserName arguments, 209
AddWindowsPrinterConnection (WSHNetwork objects), 209

connection to network printers, 223-224
PrinterPath arguments, 209, 224
AppActivate method (WScript.Shell object), 183-184
appendChild methods (IXMLDOMNode object), 174
BodyPart collections methods, 247
BodyPart object methods
AddBodyPart methods, 248
GetEncodedContentStream() methods, 248
SaveToFile methods, 248
BuildPath method (Scripting.FileSystemObject object), 125, 130
CDO.Message object methods
AddAttachment method, 238-239
AddRelatedBodyPart method, 239-241
CreateMHTMLBody method, 240-242
Send method, 240
ChangePassword method (IADsUser object), 363
Close method (TextStream object), 152
ConnectServer method, parameters, 291-292
Continue method (IADsServiceOperations object), 360
Copy
Scripting.Folder object, 141
Scripting.File object, 145
CopyFile method (Scripting.FileSystemObject object), 125
CopyFolder method (Scripting.FileSystemObject object), 125, 135
CopyHere method (IADsCollection object), 337
Count method
Environment collection, 198
Named collection, 386
Create method (IADsContainer object), 337-338
createCDataSection method (DOMDocument object), 171
createComment method (DOMDocument object), 171
createDocumentType method (DOMDocument object), 171
createElement method (DOMDocument object), 171
CreateFolder method (Scripting.FileSystemObject object), 125, 132-134
CreateFullPath method (Scripting.FileSystemObject object), 133-134
createProcessingInstruction method (DOMDocument object), 171
CreateShortcut method (WScript.Shell object), 184
CreateTextFile method (Scripting.FileSystemObject object), 126
createTextNode method (DOMDocument object), 172
defining, 93, 100
Delete method
IADsCollection object, 338
IADsContainer object, 338
Scripting.File object, 146
DeleteFile method (Scripting.FileSystemObject object), 126, 132
DeleteFolder method (Scripting.FileSystemObject object), 126
DriveExists method (Scripting.FileSystemObject object), 126
dynamic methods, 306
EnumNetworkDrives() (WSHNetwork objects), 209
listing drive mappings, 214-217
EnumPrinterConnections() (WSHNetwork objects), 209
displaying printer information, 222-223
Exec method (WScript.Shell object), 184
Exists method (Named collection), 387
ExpandEnvironmentStrings method (WScript.Shell object), 184

FileExists method (Scripting.FileSystemObject object), 126
FolderExists method (Scripting.FileSystemObject object), 126
Get method (IADs object), 334-335
GetAbsolutePathName method (Scripting.FileSystemObject object), 126
GetBaseName method (Scripting.FileSystemObject object), 127, 131-132
GetDrive method (Scripting.FileSystemObject object), 127
GetDriveName method (Scripting.FileSystemObject object), 127
getElementsByTagName method (DOMDocument object), 172
GetEx method (IADs object), 335
GetExtensionName method (Scripting.FileSystemObject object), 127
GetFile method (Scripting.FileSystemObject object), 127
GetFileName method (Scripting.FileSystemObject object), 127
GetFolder method (Scripting.FileSystemObject object), 127
GetInfo method (IADs object), 335
GetInfoEx method (IADs object), 335
getNamedItem methods (IXMLDOMNamedNode object), 176
GetObject method (IADsContainer object), 338
GetParentFolderName method (Scripting.FileSystemObject object), 127
GetSpecialFolder method (Scripting.FileSystemObject object), 128
GetTempName() method (Scripting.FileSystemObject object), 128
hasChildNodes methods (IXMLDOMNode object), 174
insertBefore methods (IXMLDOMNode object), 174
Item methods (collection objects), 103
item methods (IXMLDOMNamedNode object), 176
load method (DOMDocument object), 172
Load methods (CDO.Configuration objects), 251
LoadFrom method (CDO.Configuration method), cdoConfigSource constants, 253
loadXML method (DOMDocument object), 172
LogEvent method (WScript.Shell object), 184
MapNetworkDrive (WSHNetwork objects), 210
  adding drive mappings, 218-219
  drive mappings, 220-221
  LocalName arguments, 210
  Password arguments, 210
  RemoteName arguments, 210
  UpdateProfile arguments, 210
  UserName arguments, 210
Move method (Scripting.File object), 146
MoveFile method (Scripting.FileSystemObject object), 129, 132
MoveFolder method (Scripting.FileSystemObject object), 129
MoveHere method (IADsCollection object), 338
OpenAsTextStream method (Scripting.File object), 146
OpenTextFile method (Scripting.FileSystemObject object), 129
Pause method
  IADsPrintJobOperations object, 353
  IADsPrintQueueOperations object, 356
  IADsServiceOperations object, 360
Popup method (WScript.Shell object), 184-185

PowerShell, 626-627
Purge method (IADsPrintQueueOperations object), 356
Put method (IADs object), 335
PutEx method (IADs object), 335-336
Read method (TextStream object), 152, 163, 166
ReadAll method (TextStream object), 152
ReadLine method (TextStream object), 152-153
RegDelete method (WScript.Shell object), 186
RegRead method (WScript.Shell object), 186, 202
RegWrite method (WScript.Shell object), 186, 203
Remove methods (Environment collection), 198
removeChild methods (IXMLDOMNode object), 174
RemoveNetworkDrive (WSHNetwork objects), 210
deleting drive mappings, 219-220
Force arguments, 210
Name arguments, 210
UpdateProfile arguments, 210
RemovePrinterConnection (WSHNetwork objects), 211
deleting printer connections, 226-228
Force arguments, 211
Name arguments, 211
UpdateProfile arguments, 211
replaceChild methods (IXMLDOMNode object), 174
Resume method
IADsPrintJobOperations object, 353
IADsPrintQueueOperations object, 356
Run method (WScript.Shell object), 186
Save method
DOMDocument object, 172
WshShortcut object, 194
WshUrlShortcut object, 194
selectNodes method
DOMDocument object, 172
IXMLDOMNode object, 174
selectSingleNode method
DOMDocument object, 172
IXMLDOMNode object, 174
SendKeys method (WScript.Shell object), 186
setAttribute methods (IXMLDOMNode object), 175
SetDefaultPrinter (WSHNetwork objects), 211
setting default printers, 228
SetInfo method
IADs object, 336
IADsDomain object, 344
IADsUser object, 363
SetPassword method
IADsServiceOperations object), 360
IADsUser object, 363
Skip method (TextStream object), 152
SkipLine method (TextStream object), 152
Start method (IADsServiceOperations object), 361
static methods, 306
Stop method (IADsServiceOperations object), 361
SWbemObjects objects methods, 307
SWbemServices objects methods
Delete method, 302
ExecMethod method, 302
ExecQuery method, 302-305
InstancesOf method, 303
Terminate method (WshScriptExec object), 189
versus properties, 100-101
Win32_Service objects methods, 316
Write method (TextStream object), 152
WriteBlankLines method (TextStream object), 152, 159
WriteLine method (TextStream object), 152-156, 159

Methods properties (SWbemServices objects), 306

Methods property (IADsGroup object), 349

Microsoft Developer's Network Website, 50, 232
Microsoft documentation, ADSI (Active Directory Scripting Interface) objects, 332
Microsoft Exchange. See Exchange (Microsoft)
Microsoft TechNet website, 50
Microsoft Website, 70
Mid() function, 72-73
MIME (Multipart Internet Mail Extensions), 234
MIMEFormatted property (CDO.Message objects), 237
MinPasswordAge property (IADsDomain object), 343
MinPasswordLength property (IADsDomain object), 343
Miscellaneous Settings tab (NTVDM), configuring, 530-532
Miscellaneous Settings tab (Properties dialog box), 530-531
mkdir command (CMD), 469
Model property (IADsPrintQueue object), 354
modifiers, for command, 733-734
modifying shortcuts, 193-196
modules (PowerShell), writing, 701
monikers, 100
security options, specifying, 300-301
WMI (Windows Management Interface), connecting with, 292-293
monitoring
Hotfixes, WMI (Windows Management Interface), 313-314
Windows service packs
example WMI script, 314-315
WMI (Windows Management Interface), 313-314
more (command-line programs), 552-553
more command, 552-553
more program (Windows XP), 741
move command (CMD), 469
Move methods (Scripting.File object), 146
MoveFile method (Scripting.FileSystemObject object), 129, 132
MoveFolder method (Scripting.FileSystemObject object), 129
MoveHere method (IADsCollection object), 338
MoveHere method (IADsContainer object), 338
MP3 tag data, reading, 166-167
MS-DOS
configuring, 525-526
managing programs, 540
NTVDM (Windows NT Virtual DOS Machine), 522-523
configuring, 525-536
configuring serial communications, 539
hardware support, 539
networking, 536-537
printing, 537
printing redirection, 538
Properties dialog box
Compatibility tab, 532
Font tab, 528
Memory tab, 528
Miscellaneous Settings tab, 530-531
Program tab, 526-528
Screen tab, 530
troubleshooting, 540-541
MsgBox functions (VBScripts), 69-70, 79-82
documenting, 70
MSXML2.DOMDocument object, 169
multidimensional PowerShell arrays, 634
multiformat messages, sending, 260
multipart internet mail extensions. *See* MIME
multiple arguments, processing in batch files, 503-506
multiple attributes, testing, 143

multiple commands (CMD shell), typing on one line, 452-453

multiple files, searching (findstr command-line programs), 549

multiple inheritance, 324-325

multiple scripts, enclosing in WSF files, 390

multiple workstations, replicating scripts to, 406-408

N

Name arguments (WSHNetwork objects), 210-211

name completion (CMD shell), 450-451

Name properties
Scripting.File object, 145-146
Scripting.Folder object, 140
Win32_Service objects, 315
IADs object, 334
IADsFileShare object, 348
IADsGroup object, 349
IADsOU object, 369
IADsPrintQueue object, 355
IADsResource object, 346
IADsService object, 358

named arguments (command-line arguments)
extracting, 387
processing, 386-388

Named collection (command-line arguments), 386-387

named constants, 52-53

namespaces, 281-283

namingContext property (RootDSE object), 368

nested objects, 101-102

net command (networking utilities), 574

net command-line tool, 574-583

net continue command (networking utilities), 574

net file command (networking utilities), 574

.NET framework, PowerShell, 596-597
dates and times, 677-679
mathematical functions, 680
static member functions, calling, 673
strings, 674-676

net help command (networking utilities), 575

net helpmsg command (networking utilities), 575

net localgroup command (networking utilities), 575

net pause command (networking utilities), 575

net print command (networking utilities), 575-576

net send command (networking utilities), 576

net session command (networking utilities), 576-577

net share command (networking utilities), 577

net start command (networking utilities), 578

net statistics command (networking utilities), 578

net stop command (networking utilities), 578

net use command (networking utilities), 579-581

net user command (networking utilities), 581, 583

net view command (networking utilities), 583-584

netstat command (networking utilities), 584
active connections, listing, 585
constant monitoring, 586
open ports, listing, 586
statistics, listing, 586

network management, 207-211
domain networks
logon scripts, assigning through Group Policy, 418-421
remote management with WMI, 283-284
drive mappings
adding, 218-219

configuring, 220-221
deleting, 219-220
listing, 214-218
multiple computers, replicating scripts to, 406-408
workgroup networks, remote management with WMI, 284-287

network mappings (batch files)
deleting, 518
managing, 518

network user information, retrieving, 212-214

networking utilities (Windows XP), 741-742
ipconfig command, 571
examining/clearing DNS cache, 573
listing IP address information, 571-572
resetting automatically assigned addresses, 573
net command, 574
net continue command, 574
net file command, 574
net help command, 575
net helpmsg command, 575
net localgroup command, 575
net pause command, 575
net print command, 575-576
net send command, 576
net session command, 576-577
net share command, 577
net start command, 578
net statistics command, 578
net stop command, 578
net use command, 579-581
net user command, 581-583
net view command, 583-584
netstat command, 584
constant monitoring, 586
listing active connections, 585
listing open ports, 586
listing statistics, 586
nslookup command, 586
finding hostname IP addresses, 587-589
testing DNS servers, 589
ping command, 589-590
tracert command, 591-592

networks, deploying scripts on, 394

newsgroups, VBScript newsgroup Websites, 92

nextSiblings properties (IXMLDOMNode object), 173

nntpauthenticate fields (CDO.configuration objects), cdoProtocolAuthentication constants, 254

node types, IXMLDOMNode object, 175

nodeName properties (IXMLDOMNode object), 173

nodes, 169

nodeType properties (IXMLDOMNode object), 173

nodeType values (IXMLDOMNode object), 175

nodeTypeString properties (IXMLDOMNode object), 174

NodeValue properties (IXMLDOCNode object), 176

nodeValue properties (IXMLDOMNode object), 174

non-standard objects, 115-120

Notify property (IADsPrintJob object), 352

NotifyPath property (IADsPrintJob object), 352

Now() functions (VBScript), 75

nslookup command (networking utilities), 586
DNS servers, testing, 589
hostname IP addresses, finding, 587-589

nslookup command-line tool, 586-589

NTVDM (Windows NT Virtual DOS Machine), 522-523
applications, terminating, 540
AUTOEXEC.NT file, 535-536
CONFIG.NT file, 532-535
configuring, 525-526
AUTOEXEC.NT, 535
CONFIG.NT, 532-535
environment variables, 536
Font tab, 528

Memory tab, 528-529
Miscellaneous Settings tab, 530-532
Program tab, 526-528
Screen Settings tab, 530
drive letters, mapping, 537
environment variables, 536
MS-DOS hardware support, 539
networking, 536-537
printing, 537
printing redirection, 538
Properties dialog box
Compatibility tab, 532
Font tab, 528
Memory tab, 528
Miscellaneous Settings tab, 530-531
Program tab, 526-528
Screen tab, 530
serial communications, configuring, 539

numeric constants (VBScripts), 51

numerical calculations, performing for batch files, 481

numerical for loops, 486

O

object browsers, 118
classes, viewing, 118
OLE/COM Object Viewer, 119

objects, 10, 93-94
IXMLDOMNode
methods, 174-175
node types, 175
properties, 173-174
Active Directory objects, 368
IADsO object, 369-370
IADsOU object, 369-370
RootDSE object, 368-369
ActivePerl, 106
collections, 108-109
Perl Object Interface, 107-108
running Perl scripts in Windows Script Host, 106-107
ActivePython, 109-110
collections, 110
ADO, Stream object, 250
ADSI, 322, 324
class names, 325
creating, 325-328
leaves, 330
multiple inheritance, 324
RootDSE, 368-369
automation, GetObject function, 99-100
CDO, messaging objects, 232-250
classes, 95-96
CLSID, 116
collection objects, 96-97
ActivePerl, 108-109
ActivePython, 110
Count properties, 103
For Each loops, 103
Item methods, 103
JScript, 104-106
collections, 96
Com (Common Object Model) objects, 94
containers, 96
defining, 10-11, 93
DOMDocument objects (Windows Scripting Host), 169-170
async properties, 170
childNodes properties, 171
createCDataSection methods, 171
createComment methods, 171
createDocumentType method, 171
createElement method, 171
createProcessingInstruction method, 171
createTextNode method, 172
documentElement properties, 171
getElementsByTagName method, 172
load method, 172
loadXML method, 172
parseError properties, 171
Save method, 172
selectNodes method, 172
selectSingleNode method, 172
xml properties, 171
fax objects, FAXCOMEx.FaxDocument object, 271-274
IADs object (ADSI), 333
AdsPath property, 334
Class property, 334
Get method, 334-335
GetEx method, 335
GetInfo method, 335
GetInfoEx method, 335
GUID property, 334

Name property, 334
Parent property, 334
Put method, 335
PutEx method, 335-336
Schema property, 334
SetInfo method, 336
IADsCollection object (ADSI)
CopyHere method, 337
Count property, 336
Create method, 337-338
Delete method, 338
Filter property, 336
GetObject method, 338
Hints property, 337
MoveHere method, 338
IADsComputer object (ADSI), 340
IADsComputerOperations object (ADSI), 340
IADsContainer object (ADSI), 336
CopyHere method, 337
Count property, 336
Create method, 337-338
Delete method, 338
Filter property, 336
GetObject method, 338
Hints property, 337
MoveHere method, 338
IADsDomain object (ADSI), 342
AutoUnlockInterval property, 343
Filter property, 343
IsWorkgroup property, 343
LockoutObservationInterval property, 343
MaxBadPasswordsAllowed property, 343
MaxPasswordAge property, 343
MinPasswordAge property, 343
MinPasswordLength property, 343
PasswordAttributes property, 343-344
PasswordHistoryLength property, 344
SetInfo method, 344
IADsFileService object (ADSI)
MaxUserCount property, 345
Resources property, 345
Sessions property, 346
IADsFileServiceOperations object (ADSI), 345
MaxUserCount property, 345
Resources property, 345
Sessions property, 346
IADsFileShare object (ADSI)
CurrentUserCount property, 347
Description property, 348
HostComputer property, 348
MaxUserCount property, 348
Name property, 348
Path property, 348
IADsGroup object (ADSI), 349-350
IADsMembers object (ADSI), 350
IADsNamespaces object (ADSI), 351
IADsO object (Active Directory)
Count property, 369
Description property, 369
FaxNumber property, 369
Filter property, 369
LocalityName property, 369
Name property, 369
Parent property, 369
PostalAddress property, 370
SeeAlso property, 370
TelephoneNumber property, 370
IADsOU object (Active Directory)
BusinessCategory property, 369
Count property, 369
Description property, 369
FaxNumber property, 369
Filter property, 369
LocalityName property, 369
Name property, 369
Parent property, 369
PostalAddress property, 370
SeeAlso property, 370
TelephoneNumber property, 370
IADsPrintJob object (ADSI), 351-354
IADsPrintJobOperations object (ADSI), 351-352, 354
IADsPrintQueue object (ADSI), 356-357
BannerPage property, 354
Datatype property, 354
DefaultJobPriority property, 354
Description property, 354
HostComputer property, 354
Location property, 354
Model property, 354
Name property, 355
PrintDevices property, 355
PrinterPath property, 355
PrintProcessor property, 355

Priority property, 355
Starttime property, 355
UntilTime property, 355
IADsPrintQueueOperations object (ADSI), 354-357
IADsResource object (ADSI), 346
IADsService object (ADSI), 357, 360
Dependencies property, 358
DisplayName property, 358
ErrorControl property, 358-359
HostComputer property, 358
LoadOrderGroup property, 358
Name property, 358
Path property, 358
ServiceAccountName property, 358
ServiceAccountPath property, 358
ServiceType property, 358
SetInfo method, 359
StartType property, 359
Startup Parameters property, 359
Version property, 359
IADsServiceOperations object (ADSI), 357
Continue method, 360
Pause method, 360
SetPassword method, 360
Start method, 361
Status property, 360
Stop method, 361
IADsSession object (ADSI), 361-362
IADsUser object (ADSI)
AccountDisabled property, 362
ChangePassword method, 363
Description property, 362
FullName property, 362
Groups property, 362
HomeDirectory property, 362
IsAccountLocked property, 362
LastLogin property, 363
LastLogoff property, 363
Profile property, 363
SetInfo method, 363
SetPassword method, 363
instances, 95
IXMLDOCNode objects (Windows Scripting Host), 176
IXMLDOMNamedNode objects (Windows Scripting Host), 176
IXMLDOMNode objects (Windows Scripting Host)
appendChild method, 174
attributes properties, 173, 175
childNodes properties, 173
firstChild properties, 173
hasChildNodes method, 174
insertBefore method, 174
lastChild properties, 173
nextSiblings properties, 173
nodeName properties, 173
nodeType properties, 173
nodeType values, 175
nodeTypeString properties, 174
nodeValue properties, 174
ownerDocument properties, 174
previousSibling properties, 174
removeChild method, 174
replaceChild method, 174
selectNodes method, 174
selectSingleNode method, 174
setAttribute method, 175
xml properties, 174
JScript
case sensitivity, 104
collections, 104-106
Enumerator objects, 105
WScript, 104
messaging objects, CDO.Configuration object, 250-256
methods
AddPrinterConnection, 208, 225-226, 229-230
AddWindowsPrinterConnection, 209, 223-224
defining, 93, 100
EnumNetworkDrives, 209, 214-217
EnumPrinterConnections, 209, 222-223
MapNetworkDrive, 210, 218-221
RemoveNetworkDrive, 210, 219-220
RemovePrinterConnection, 211, 226-228
SetDefaultPrinter, 211, 228
versus properties, 100-101
monikers, 292-293, 300-301
MSXML2.DOMDocument, 169
naming, 97

nested objects, 101-102
non-standard, 115-120
object browsers, 118-119
OLE/COM Object Viewer, 119
Perl, 107-109
PowerShell, generating, 685-686
properties
defining, 93, 100
read-only properties, 101
Python, 110
releasing, 102
RootDSE object (Active Directory)
currentTime property, 368
defaultNamingContext property, 368
dnsHostTime property, 368
namingContext property, 368
rootDomainNamingContext property, 368
serverName property, 368
supportedLDAPVersion property, 369
Scripting.Drive object (Windows Scripting Host), 135-136
Scripting.Drive objects (Windows Scripting Host)
AvailableSpace properties, 136
DriveLetter properties, 136
DriveType properties, 136
FileSystem properties, 137
FreeSpace properties, 137
IsReady properties, 137-138
Path properties, 137
RootFolder properties, 137
SerialNumber properties, 137
ShareName properties, 137
TotalSize properties, 137
VolumeName properties, 137
Scripting.File objects (Windows Scripting Host)
Attributes properties, 145
copy method, 145
DateCreated properties, 145
DateLastAccessed properties, 145
DateLastModified properties, 145
Delete method, 146
Drive properties, 145
Move method, 146
Name properties, 145-146
OpenAsTextStream method, 146
ParentFolder properties, 145
Path properties, 145
ShortName properties, 145
ShortPath properties, 145
Type properties, 145
Scripting.FileSystemObject, 124
methods, 125-130
Scripting.FileSystemObject (Windows Scripting Host), 124
BuildPath method, 125, 130
CopyFile method, 125
CopyFolder method, 125, 135
CreateFolder method, 125, 132-134
CreateFullPath method, 133-134
CreateTextFile method, 126
DeleteFile method, 126, 132
DeleteFolder method, 126
DriveExists method, 126
Drives property, 125
FileExists method, 126
FolderExists method, 126
GetAbsolutePathName method, 126
GetBaseName method, 127, 131-132
GetDrive method, 127
GetDriveName method, 127
GetExtensionName method, 127
GetFile method, 127
GetFileName method, 127
GetFolder method, 127
GetParentFolderName method, 127
GetSpecialFolder method, 128
GetTempName() method, 128
MoveFile method, 129, 132
MoveFolder method, 129
OpenTextFile method, 129
Scripting.Folder object (Windows Scripting Host), 139
Attributes properties, 140
copy method, 141
DateCreated properties, 140
DateLastAccessed properties, 140
DateLastModified properties, 140
Delete method, 141
Drive properties, 140
file attribute values, 141-144
Files properties, 140
folder attribute values, 141-144
IsRootFolder properties, 140
multiple attributes, 143
Name properties, 140
ParentFolder properties, 140
Path properties, 141

ShortName properties, 141
ShortPath properties, 141
Size properties, 141
SubFolders properties, 141
Type properties, 141
SWbemObjectSet collection object, 305-306
SWbemServices, 300-301
methods, 302-303
properties, 302
TextStream object (Windows Scripting Host), 150
AtEndOfLine properties, 151
AtEndOfStream properties, 151, 162
Close methods, 152
Column properties, 151
Line properties, 151
Read methods, 152, 163, 166
ReadAll methods, 152
ReadLine methods, 152-153
Skip methods, 152
SkipLine methods, 152
Write methods, 152
WriteBlankLines methods, 152, 159
WriteLine methods, 152-156, 159
VBScript, 98-100
collections, 67, 102-103
nested objects, 101-102
releasing, 102
Win32_Service
methods, 316
properties, 315-316
Windows Registry, viewing, 115
WMI, 288-289, 291
SWbemObject, 306-307
WbemScripting.SWbemLocator object, 291-292
WScript, 111-112
command-line arguments, retrieving, 113-114
methods, 112-113
properties, 111-112
WScript.Network
default printer, setting, 228
DOS printer sessions, redirecting, 225-226
drive mappings, adding, 218-219
drive mappings, configuring, 220-221
drive mappings, deleting, 219-220
drive mappings, listing, 214, 216-218
network connections, managing, 207
network user information, retrieving, 212, 214
printer information, displaying, 222-223
printers connections, deleting, 226-228
WScript.Shell object (Windows Scripting Host)
AppActivate method, 183-184
CreateShortcut method, 184
CurrentDirectory properties, 182
Environment properties, 182-183, 196-197
Exec method, 184
ExpandEnvironmentStrings method, 184
LogEvent method, 184
Popup method, 184-185
RegDelete method, 186
RegRead method, 186, 202
RegWrite method, 186, 203
Run method, 186
SendKeys method, 186
SpecialFolders properties, 183
WSH
FileSystemObject, 130
Scripting.Drive, 135-137
Scripting.File, 145-146
Scripting.Folder, 139-141
TextStream, 150-152
WScript.Shell, 182-186
WSHNetwork objects
AddPrinterConnection method, 208, 225-226, 229-230
AddWindowsPrinterConnection method, 209, 223-224
ComputerName properties, 208
EnumNetworkDrives() method, 209, 214-217
EnumPrinterConnections() method, 209, 222-223
MapNetworkDrive method, 210, 218-221
RemoveNetworkDrive method, 210, 219-220
RemovePrinterConnection method, 211, 226-228
SetDefaultPrinter method, 211, 228

UserDomain properties, 208
UserName properties, 208
WshScriptExec object (Windows Scripting Host)
ExitCode properties, 188
ProcessID properties, 189
Status properties, 189
StdErr properties, 189
StdIn properties, 189
StdOut properties, 189
Terminate method, 189
WshShortcut object (Windows Scripting Host)
Arguments properties, 193
Description properties, 194
FullName properties, 194
Hotkey properties, 194
IconLocation properties, 194
Save methods, 194
TargetPath properties, 194
WindowStyle properties, 194
WorkingDirectory properties, 194
WshUrlShortcut object (Windows Scripting Host)
Arguments properties, 193
Description properties, 194
FullName properties, 194
Hotkey properties, 194
IconLocation properties, 194
Save methods, 194
TargetPath properties, 194
WindowStyle properties, 194
WorkingDirectory properties, 194

obtaining
code-signing certificate, 410-411
PowerShell, 598-600

OLE/COM Object Viewer, 119

online help, providing for WSF files, 384-385

Open Object REXX, 15

open ports, listing, 586

OpenAsTextStream methods (Scripting.File object), 146

opening elevated Command Prompt, 436-437

OpenTextFile method (Scripting.FileSystemObject object), 129

OperatingSystem property (IADsComputer object), 340

OperatingSystemVersion property (IADsComputer object), 340

operators
& operator (PowerShell), 646
PowerShell
assignment operators, 648
precedence, 646-647
splat operators, 672-673
set /a command, 734
sting operators (PowerShell), 643-646
VBScript, 53-55, 717-718
arithmetic, 55
automatic conversion, 57
comparison, 55
logical, 56
precedence, 54

options for CMD shell, 437-438

Organization property (CDO.Message objects), 238

output redirection, 33, 445-447

Owner property (IADsComputer object), 340

ownerDocument properties (IXMLDOMNode object), 174

P

PagesPrinted property (IADsPrintJobOperations object), 353

paginating file lists, dir command (CMD), 477

parameters, passing, 134

Parent property (BodyPart objects), 248

Parent property (IADs object), 334

Parent property (IADsO object), 369

Parent property (IADsOU object), 369

ParentFolder properties (Scripting.File object), 145

ParentFolder properties (Scripting.Folder object), 140

parentheses, grouping commands, 453-454

parseError properties (DOMDocument object), 171

parsing text, 487-488

passing information to scripts, 31-32

passing parameters, 134

Password arguments (WSHNetwork objects), 209-210

PasswordAttributes property (IADsDomain object), 343-344

PasswordHistoryLength property (IADsDomain object), 344

PATH, 456-457
- adding scripts to, 37
- changing, 457-458
- for all users, adding scripts to, 37-38
- for single user, adding scripts to, 37-38
- search paths, 456-457
- special handling, 200-201

path command (CMD), 470

Path properties
- Scripting.Drive object, 137
- Scripting.File object, 145
- Scripting.Folder object, 141
- SWbemServices objects, 307
- IADsFileShare object, 348
- IADsResource object, 346
- IADsService object, 358

PathName properties (Win32_Service objects), 315

pathnames, 130, 518

paths, assigning, 134

pause command (CMD), 470

Pause method
- IADsPrintJobOperations object, 353
- IADsPrintQueueOperations object, 356
- IADsServiceOperations object, 360

PauseService methods (Win32_Service objects), 316

performing numerical calculations for batch files, 481

Perl, 15
- defining, 15
- objects, 107-109
- scripts, running, 106-107
- website, 15

Perl Object Interface, 107-108

permission control (DCOM security), WMI (Windows Management Interface), 298-299

permissions
- checking (cacls command), 562
- granting (cacls command), 562
- managing with cacls command-line tool, 562-563

PID numbers, killing processes (taskkill command), 568

ping command-line tool, 589-590

pipe mechanisms, 33

pipeline functions, PowerShell, 671-672

popd command (CMD), 470

Popup method (WScript.Shell object), 184-185

ports, listing, 586

Position property (IADsPrintJobOperations object), 353

positional searching (findstr command-line programs), 549

PostalAddress property (IADsOU object), 370

PowerShell
- & operators, 646
- aliases, 612
- arrays, 632, 634-636
  - *comparisons, 640-643*
  - *values, extracting, 636-637*
- casts, 649-650
- cmdlets, 607, 609-610, 690-691
- command-line editing, 602-603
- command-line processing, 700
- command-line syntax, 604-607
- commands, completing, 612
- comments, 622, 700
- constants, 637

constructors, 627
copying and pasting, 603-604
directories, 613-615, 692-696
exception handling, 702-703
throw command, 664
trap command, 662
expressions, 638-639
files
text, reading, 697-698
text, writing, 698
filtering, 686-689
flow-of-control commands
break, 660-661
continue, 661
do, 654
for, 655-656
foreach, 656-657
if, 653
switch, 657-660
while, 654
functions
dot-sourcing, 668
parameters, 665
scope, 668
get-help command, 610-611
hash tables, 650, 652-653, 703-704
here-strings, 629
ISE, 705
breakpoints, setting, 709
conditional breakpoints, 711
configuring, 706-707
scripts, editing, 707-708
literal values, 625-626
methods, 626-627
modules, writing, 701
.Net platform, 596-597
dates and times, 677-679
mathematical functions, 680
static member functions, calling, 673
strings, 674-676
objects, generating, 685-686
obtaining, 598-600
operators
assignment operators, 648
precedence, 646-647
splat operators, 672-673
pipeline functions, 671-672
profiles, 617-618
running on remote computers, 712
security, 615-617
string operators, 643-646
strings, 628
variables, 623-624
predefined, 630-632
releasing, 630
scope, 665-669

precedence (VBScript variable operators), 54

predefined environment variables, 459-461, 727-728

predefined PowerShell variables, 630-632

previousSibling properties (IXMLDOMNode object), 174

print redirection (NTVDM), 538

Print Screen function (Windows XP), 538

PrintDevices property (IADsPrintQueue object), 355

PrinterPath arguments (WSHNetwork objects), 209, 224

PrinterPath property (IADsPrintQueue object), 355

printers
connecting to, 223-225
connections, deleting, 226-228
default printer, setting, 228
DOS printer sessions, redirecting, 225-226
information, displaying, 222-223
managing
example WMI script, 313
WMI (Windows Management Interface), 313
network connections, deleting, 226-228
network printers, connecting to, 223-224

printing, 229-230
directory listings, 478
Event Log results, 425
log files, 425
NTVDM, 537-538
with Wscript.Echo command, 84-85

PrintJobs property (IADsPrintQueueOperations object), 356

PrintProcessor property (IADsPrintQueue object), 355
Priority property (IADsPrintJob object), 352
Priority property (IADsPrintQueue object), 355
private folders, creating, 563
private scope, 91
privileges
for batch files, 493
WMI options, 298-299
procedures (VBScripts), 87
functions
creating, 87
extending built-in functions, 88
subroutines, 89
process environment, 197
processes, killing with taskkill, 568-569
ProcessID properties (WshScriptExec object), 189
processing
batch files, command-line options, 515-517
command-line arguments, 386
conditional processing, if command (CMD), 482-483, 499-501
directories, 486
multiple arguments, batch files, 503-506
named arguments (command-line arguments), 386
unnamed arguments (command-line arguments), 389
Processor property (IADsComputer object), 340
ProcessorCount property (IADsComputer object), 340
Profile property (IADsUser object), 363
profiles, PowerShell, 617-618
program output, sending, 258
program statements, VBScript, 718-719
Program tab (NTVDM), configuring, 526-528
programming, batch files, 494-495
programs
command-line, running, 188-191
running, 183-186
Windows programs, running, 187-188
prompt command (CMD), 470
prompting user input, 162, 514-515
properties, 93
AccountDisabled property (IADsUser object), 362
AdsPath property (IADs object), 334
Arguments property (WshShortcut object), 193
Arguments property (WshUrlShortcut object), 193
AtEndOfLine property (TextStream object), 151
AtEndOfStream property (TextStream object), 151, 162
Attributes (Scripting.Folder object), 140
attributes property (IXMLDOMNode object), 173, 175
Attributes property (Scripting.File object), 145
AutoUnlockInterval property (IADsDomain object), 343
AvailableSpace (Scripting.Drive object), 136
BannerPage property (IADsPrintQueue object), 354
BodyPart collections properties
Count property, 246
Item property, 246
BodyPart object properties
BodyPart property, 247
Charset property, 247
ContentMediaType property, 247
ContentTransferEncoding property, 248
Fields property, 248
Filename property, 248
Parent property, 248
BusinessCategory property (IADsOU object), 369
CDO.Message object properties
Attachments property, 236
AutoGenerateTextBody property, 236

BCC property, 236
BodyPart property, 236
CC property, 236
Configuration property, 236
DSNOptions property, 236, 240-241
Fields property, 237
From property, 237
HTMLBody property, 237
HTMLBodyPart property, 237
MDNRequested property, 237
MIMEFormatted property, 237
Organization property, 238
ReplyTo property, 238
Sender property, 238
Subject property, 238
TextBody property, 238
TextBodyPart property, 238
To property, 238
childNodes property (DOMDocument object), 171
childNodes property (IXMLDOMNode object), 173
Class property (IADs object), 334
Column property (TextStream object), 151
Computer property (IADsSession object), 361
ComputerName properties (WSHNetwork objects), 208
ComputerPath property (IADsSession object), 361
ConnectTime property (IADsSession object), 361
Count method (Unnamed collection), 389
Count properties (collection objects), 103
Count property (IADsCollection object), 336
Count property (IADsContainer object), 336
Count property (IADsMembers object), 350
Count property (IADsO object), 369
Count property (IADsOU object), 369
CurrentDirectory property (WScript.Shell object), 182
currentTime property (RootDSE object), 368
CurrentUserCount property (IADsFileShare object), 347
Datatype property (IADsPrintQueue object), 354
DateCreated (Scripting.Folder object), 140
DateCreated property (Scripting.File object), 145
DateLastAccessed (Scripting.Folder object), 140
DateLastAccessed property (Scripting.File object), 145
DateLastModified (Scripting.Folder object), 140
DateLastModified property (Scripting.File object), 145
DefaultContainer property (IADsNamespaces object), 351
DefaultJobPriority property (IADsPrintQueue object), 354
defaultNamingContext property (RootDSE object), 368
defining, 93, 100
Dependencies property (IADsService object), 358
Description property (IADsFileShare object), 348
Description property (IADsGroup object), 349
Description property (IADsO object), 369
Description property (IADsOU object), 369
Description property (IADsPrintJob object), 352
Description property (IADsPrintQueue object), 354
Description property (IADsUser object), 362
Description property (WshShortcut object), 194
Description property (WshUrlShortcut object), 194
DisplayName property (IADsService object), 358
Division property (IADsComputer object), 340
dnsHostTime property (RootDSE object), 368

documentElement property (DOMDocument object), 171
Drive (Scripting.Folder object), 140
Drive property (Scripting.File object), 145
DriveLetter (Scripting.Drive object), 136
Drives property (Scripting.FileSystemObject object), 125
DriveType (Scripting.Drive object), 136
dynamic properties, 306
Environment property (WScript.Shell object), 182-183, 196-197
ErrorControl property (IADsService object), 358-359
ExitCode property (WshScriptExec object), 188
FaxNumber property (IADsO object), 369
FaxNumber property (IADsOU object), 369
Fields collections properties, Item property, 243
Fields property (CDO.Configuration objects), 251
nntpauthenticate field, 254
sendusing field, 254
smtpauthenticate field, 254
timezoneid field, 254-256
values, 252-253
Files (Scripting.Folder object), 140
FileSystem (Scripting.Drive object), 137
Filter property (IADsCollection object), 336
Filter property (IADsContainer object), 336
Filter property (IADsDomain object), 343
Filter property (IADsMembers object), 350
Filter property (IADsO object), 369
Filter property (IADsOU object), 369
firstChild property (IXMLDOMNode object), 173
FreeSpace (Scripting.Drive object), 137
FullName property (IADsUser object), 362
FullName property (WshShortcut object), 194
FullName property (WshUrlShortcut object), 194
Groups property (IADsUser object), 362
GUID property (IADs object), 334
Hints property (IADsCollection object), 337
Hints property (IADsContainer object), 337
HomeDirectory property (IADsUser object), 362
HostComputer property (IADsFileShare object), 348
HostComputer property (IADsPrintQueue object), 354
HostComputer property (IADsService object), 358
HostPrintQueue property (IADsPrintJob object), 352
Hotkey property (WshShortcut object), 194
Hotkey property (WshUrlShortcut object), 194
IconLocation property (WshShortcut object), 194
IconLocation property (WshUrlShortcut object), 194
IdleTime property (IADsSession object), 361
IsAccountLocked property (IADsUser object), 362
IsMember() property (IADsGroup object), 349
IsReady (Scripting.Drive object), 137-138
IsRootFolder (Scripting.Folder object), 140
IsWorkgroup property (IADsDomain object), 343
Item property (Environment collection), 197
Items property (Named collection), 386
Items property (Unnamed collection), 389
lastChild property (IXMLDOMNode object), 173

LastLogin property (IADsUser object), 363
LastLogoff property (IADsUser object), 363
Length property (Environment collection), 197
length property (IXMLDOMNamedNode object), 176
Length property (Named collection), 386
Length property (Unnamed collection), 389
Line property (TextStream object), 151
LoadOrderGroup property (IADsService object), 358
LocalityName property (IADsO object), 369
LocalityName property (IADsOU object), 369
Location property (IADsPrintQueue object), 354
LockCount property (IADsResource object), 346
LockoutObservationInterval property (IADsDomain object), 343
MaxBadPasswordsAllowed property (IADsDomain object), 343
MaxPasswordAge property (IADsDomain object), 343
MaxUserCount property (IADsFileService object), 345
MaxUserCount property (IADsFileServiceOperations object), 345
MaxUserCount property (IADsFileShare object), 348
Member() property (IADsGroup object), 349
Methods property (IADsGroup object), 349
MinPasswordAge property (IADsDomain object), 343
MinPasswordLength property (IADsDomain object), 343
Model property (IADsPrintQueue object), 354
Name (Scripting.Folder object), 140
Name property (IADs object), 334
Name property (IADsFileShare object), 348
Name property (IADsGroup object), 349
Name property (IADsO object), 369
Name property (IADsOU object), 369
Name property (IADsPrintQueue object), 355
Name property (IADsResource object), 346
Name property (IADsService object), 358
Name property (Scripting.File object), 145-146
namingContext property (RootDSE object), 368
nextSiblings property (IXMLDOMNode object), 173
NodeName property (IXMLDOCNode object), 176
nodeName property (IXMLDOMNode object), 173
nodeType property (IXMLDOMNode object), 173
nodeTypeString property (IXMLDOMNode object), 174
NodeValue property (IXMLDOCNode object), 176
nodeValue property (IXMLDOMNode object), 174
Notify property (IADsPrintJob object), 352
NotifyPath property (IADsPrintJob object), 352
of DOMdocument object, 170-171
OperatingSystem property (IADsComputer object), 340
OperatingSystemVersion property (IADsComputer object), 340
Owner property (IADsComputer object), 340
ownerDocument property (IXMLDOMNode object), 174
PagesPrinted property (IADsPrintJobOperations object), 353
Parent property (IADs object), 334
Parent property (IADsO object), 369
Parent property (IADsOU object), 369

ParentFolder (Scripting.Folder object), 140
ParentFolder property (Scripting.File object), 145
parseError property (DOMDocument object), 171
PasswordAttributes property (IADsDomain object), 343-344
PasswordHistoryLength property (IADsDomain object), 344
Path (Scripting.Drive object), 137
Path (Scripting.Folder object), 141
Path property (IADsFileShare object), 348
Path property (IADsResource object), 346
Path property (IADsService object), 358
Path property (Scripting.File object), 145
Position property (IADsPrintJobOperations object), 353
PostalAddress property (IADsO object), 370
PostalAddress property (IADsOU object), 370
previousSibling property (IXMLDOMNode object), 174
PrintDevices property (IADsPrintQueue object), 355
PrinterPath property (IADsPrintQueue object), 355
PrintJobs property (IADsPrintQueueOperations object), 356
PrintProcessor property (IADsPrintQueue object), 355
Priority property (IADsPrintJob object), 352
Priority property (IADsPrintQueue object), 355
ProcessID property (WshScriptExec object), 189
Processor property (IADsComputer object), 340
ProcessorCount property (IADsComputer object), 340
Profile property (IADsUser object), 363
read-only properties, 101
Remove property (IADsGroup object), 349
Resources property (IADsFileService object), 345
Resources property (IADsFileServiceOperations object), 345
rootDomainNamingContext property (RootDSE object), 368
RootFolder (Scripting.Drive object), 137
Schema property (IADs object), 334
SeeAlso property (IADsOU object), 370
SerialNumber (Scripting.Drive object), 137
serverName property (RootDSE object), 368
ServiceAccountName property (IADsService object), 358
ServiceAccountPath property (IADsService object), 358
ServiceType property (IADsService object), 358
Sessions property (IADsFileService object), 346
Sessions property (IADsFileServiceOperations object), 346
ShareName (Scripting.Drive object), 137
ShortName property (Scripting.File object), 145
ShortPath (Scripting.Folder object), 141
ShortPath property (Scripting.File object), 145
Size (Scripting.Folder object), 141
Size property (IADsPrintJob object), 352
SpecialFolders property (WScript.Shell object), 183
specified property (IXMLDOCNode object), 176
StartTime property (IADsPrintJob object), 352
Starttime property (IADsPrintQueue object), 355
StartType property (IADsService object), 359

Startup Parameters property (IADsService object), 359
static properties, 306
Status property (IADsPrintJobOperations object), 353
Status property (IADsPrintQueueOperations object), 356
Status property (IADsServiceOperations object), 360
Status property (WshScriptExec object), 189
StdErr property (WshScriptExec object), 189
StdIn property (WshScriptExec object), 189
StdOut property (WshScriptExec object), 189
SubFolders (Scripting.Folder object), 141
supportLDAPVersion property (RootDSE object), 369
SWbemObjects objects properties
Methods properties, 306
Path properties, 307
Property properties, 307
SWbemServices objects properties, Security property, 302
TargetPath property (WshShortcut object), 194
TargetPath property (WshUrlShortcut object), 194
TelephoneNumber property (IADsO object), 370
TelephoneNumber property (IADsOU object), 370
TimeElapsed property (IADsPrintJobOperations object), 353
TimeSubmitted property (IADsPrintJob object), 352
TotalPages property (IADsPrintJob object), 352
TotalSize (Scripting.Drive object), 137
Type (Scripting.Folder object), 141
Type property (Scripting.File object), 145
UntilTime property (IADsPrintJob object), 352
UntilTime property (IADsPrintQueue object), 355
User property (IADsPrintJob object), 352
User property (IADsResource object), 346
User property (IADsSession object), 361
UserDomain properties (WSHNetwork objects), 208
UserName properties (WSHNetwork objects), 208
UserPath property (IADsPrintJob object), 352
UserPath property (IADsResource object), 346
UserPath property (IADsSession object), 361
Version property (IADsService object), 359
versus methods, 100-101
VolumeName (Scripting.Drive object), 137
Win32_Service objects properties
DesktopInteract properties, 315
DisplayName properties, 315
Name properties, 315
PathName properties, 315
Started properties, 315
StartMode properties, 316
StartName properties, 316
State properties, 316
Status properties, 316
WindowStyle property (WshShortcut object), 194
WindowStyle property (WshUrlShortcut object), 194
WorkingDirectory property (WshShortcut object), 194
WorkingDirectory property (WshUrlShortcut object), 194
xml property (DOMDocument object), 171
xml property (IXMLDOMNode object), 174

Property properties (SWbemServices objects), 307

providers (ADSI)
LDAP, 364, 366-367
IADsO object, 369-370
IADsOU object, 369-370
RootDSE object, 368-369
supported objects, 327-328
WinNT, 332-333
IADs object, 333-336
IADsCollection object, 336-338
IADsComputer object, 340-342
IADsComputerOperations object, 340-342
IADsContainter object, 336-338
IADsDomain object, 342-344
IADsFileService object, 345-347
IADsFileServiceOperations object, 345-347
IADsFileShare object, 347-348
IADsGroup object, 349-350
IADsMembers object, 350
IADsNamespaces object, 351
IADsPrintJob object, 351-354
IADsPrintJobOperations object, 351-354
IADsPrintQueue object, 354-357
IADsPrintQueueOperations object, 354-357
IADsService object, 357-361
IADsServiceOperations object, 357-361
IADsSession object, 361-362
IADsUser object, 362-363

providing WSF file online help, 384-385

Purge method (IADsPrintQueueOperations object), 356

pushd command (CMD), 470, 518

Put method
IADs object, 335
SWbemServices objects, 307

PutEx method (IADs object), 335-336

pvk2pvc.exe, downloading, 410

Python, 15
defining, 15
objects, 110

Q-R

queries, WQL, 303, 305

quotes, CMD command-line processing, 454

rd command (CMD), 471

RDNs (relative distinguished names), 365

Read methods (TextStream object), 152, 163, 166

read-only properties, 101

ReadAll methods (TextStream object), 152

reading
binary files, 163
BMP image data, 164-167
MP3 tag data, 166-167
files, 149
Registry values, 202-203
text
from console programs, 191-193
from files, 152-153
XML files, 176-178

ReadLine methods (TextStream object), 152-153

recipients, specifying in messages (CDO), 263

recording information to Event Log, 423
messages, printing, 425
results, summarizing, 425-427

recursion, 134-135

redirecting DOS session printers, 225-226

redirection, 443-447, 729

references, 95

RegDelete method (WScript.Shell object), 186

Registry, 201
keys, 202
saving information in, 203-205
unsigned scripts, disabling, 408-409
values, reading, 202-203
viewing, 115

RegRead method (WScript.Shell object), 186, 202

regular expressions, 552

RegWrite method (WScript.Shell object), 186, 203

releasing
objects, 102
PowerShell variables, 630

releasing objects, 102

rem command (CMD), 471

remote management with WMI, 283
on domain networks, 283-284
on workgroup networks, 284-287

remote-management scripts, writing, 403-405

RemoteName arguments (WSHNetwork objects), 209-210

Remove methods (Environment collection), 198

Remove property (IADsGroup object), 349

removeChild methods (IXMLDOMNode object), 174

RemoveNetworkDrive methods(WSHNetwork objects)
arguments, 210
drive mappings, deleting, 219-220

RemovePrinterConnection methods(WSHNetwork objects)
arguments, 211
printer connections, deleting, 226-228

rename command (CMD), 471

renaming files, 132, 146

replaceChild methods (IXMLDOMNode object), 174

replicating scripts to multiple computers, 406-408

ReplyTo property (CDO.Message objects), 238

requiring signed scripts, 414

resetting automatically assigned addresses, 573

Resources property (IADsFileService object), 345

results from scripts, saving, 32-33

Resume method (IADsPrintJobOperations object), 353

Resume method (IADsPrintQueueOperations object), 356

retrieving
filename listings, 479
network user information, 212-214

retrieving
command-line arguments, 113-114
environment variables, 198-199

REXX, 15

Right() function, 72-73

rmdir command (CMD), 471

rootDomainNamingContext property (RootDSE object), 368

RootDSE object (ADSI), 368-369

RootFolder properties (Scripting.Drive object), 137

routing email through SMTP servers, 264

Ruby, 15-16

Run method (WScript.Shell object), 186

runas command-line tool, 565

runaway command-line programs, stopping, 440

running
CMD shell, 435
command-line programs, 188-191
Perl scripts in Windows Script Host, 106-107
programs, 183-186
script files, 29-30
scripts
automatically, 40
from batch files, 39
Perl, 106-107
with shortcut icon, 38-39
Windows programs, 187-188

S

sample scripts, 748-752

sample scripts (VBScripts), 16-19
Websites, 50

Save methods (DOMDocument object), 172

Save methods (WshShortcut object), 194

Save methods (WshUrlShortcut object), 194

SaveToFile method (BodyPart objects), 248

saving
information in Registry, 203-205
script files, output redirection, 33
script results, 32-33

sc command (management power tools)
sc queryex command, listing installed services, 569-570
services, starting/stopping, 570

scanning for files, 146-149

Scheduled Task Wizard, 430

scheduling
automatic scripts, 421-431
scripts with Task Scheduler, 428-430

Schema property (IADs object), 334

Screen Settings tab (NTVDM), configuring, 530

Screen tab (Properties dialog box), 530

Script Debugger (Windows), 43
function keys, 44
script function values, displaying, 45

script editing tools, 27

Script Encoder, 21, 415

script files
creating, 24-25
extensions, 19-20
JSE, 20
VBE, 20
running, 29-30
Script Encoder, 21
Windows Script Component files, 23
Windows Script files, 21-22
WSF (Windows Script Files), 21-23
WSH (Windows Script Host) settings, 23

scripting, 10-11
example script, 16-19
versus batch files, 13

scripting languages, 13
encoding tools, 20
JScript, 14, 104-106
Perl, 15
collections, 108-109
objects, 107-108
scripts, running in WSH, 106-107
Python, 15, 110
REXX, 15
Ruby, 15-16
selecting, 16
VBScript, 14, 49-50
arrays, 89-91
automatic conversion, 57
constants, 51-53
error handling, 86-87
flow control, 57-68
functions, 69-84
objects, 98-103
operators, 53-56
procedures, 87-89
variable scope, 91-92
variables, 50-51
Wscript.Echo command, 84-85
versus compiled languages, 13

Scripting.Drive object (Windows Scripting Host), 135
properties
AvailableSpace properties, 136
DriveLetter properties, 136
DriveType properties, 136
FileSystem properties, 137
FreeSpace properties, 137
IsReady properties, 137-138
Path properties, 137
RootFolder properties, 137
SerialNumber properties, 137
ShareName properties, 137
TotalSize properties, 137
VolumeName properties, 137

Scripting.File object (Windows Scripting Host)
methods
Copy method, 145
Delete method, 146
Move method, 146
OpenAsTextStream method, 146
properties
Attributes properties, 145
DateCreated properties, 145
DateLastAccessed properties, 145
DateLastModified properties, 145
Drive properties, 145
Name properties, 145-146
ParentFolder properties, 145
Path properties, 145
ShortName properties, 145
ShortPath properties, 145
Type properties, 145

Scripting.FileSystemObject object (Windows Scripting Host), 124
methods
BuildPath method, 125, 130
CopyFile method, 125
CopyFolder method, 125, 135
CreateFolder method, 125, 132-134
CreateFullPath method, 133-134
CreateTextFile method, 126
DeleteFile method, 126, 132
DeleteFolder method, 126
DriveExists method, 126
FileExists method, 126
FolderExists method, 126
GetAbsolutePathName method, 126
GetBaseName method, 127, 131-132
GetDrive method, 127
GetDriveName method, 127
GetExtensionName method, 127
GetFile method, 127
GetFileName method, 127
GetFolder method, 127
GetParentFolderName method, 127
GetSpecialFolder method, 128
GetTempName method, 128
MoveFile method, 129, 132
MoveFolder method, 129
OpenTextFile method, 129
properties, Drives property, 125

Scripting.Folder object (Windows Scripting Host), 139, 144
file attribute values, 141
changing, 143-144
testing, 142
folder attribute values, 141
changing, 143-144
testing, 142
methods, 141
multiple attributes, testing, 143
properties
Attributes properties, 140
DateCreated properties, 140
DateLastAccessed properties, 140
DateLastModified properties, 140
Drive properties, 140
Files properties, 140
IsRootFolder properties, 140
Name properties, 140
ParentFolder properties, 140
Path properties, 141
ShortName properties, 141
ShortPath properties, 141
Size properties, 141
SubFolders properties, 141
Type properties, 141

Scripting.Folder object, 139

Scriptomatic, 310-311

scripts
ActivePerl, 106-109
ActivePython, objects, 109-110
adding to Windows' PATH list, 37-38
ADSI, developing, 370-371
automatic scripts
creating unattended scripts, 421-423
scheduling, 421-431
breakpoints, 45
creating, 26
debugging, 42-45
Call Stack window, viewing, 46-47
Wscript.Echo command, 84
defining, 10
deploying on network, 394
drives
checking free space, 138
confirming existence, 137-138
mappings, 220-221
encoded, 20
example scripts, WSF, 390-394

files
reading, 149
scanning for, 146-149
writing, 149
installers, creating with IExpress, 398-400
interpreters
JScript language interpreters, 14
VBScript language interpreters, 14
logon scripts
group policy scripts, 420
user profile logon scripts, 416-418
multiple computers, copying to, 406-408
multiple scripts, enclosing with WSF files, 390
networks, deploying on, 394
output redirection, 33
passing information to, 31-32
Perl, running in Windows Script Host, 106-107
printing from, 229-230
properties, displaying, 23
remote management, writing, 403-405
remote scripts, 405
replicating to multiple computers, 406-408
results, saving, 32-33
running, 29
automatically, 40
from batch files, 39
with shortcut icon, 38-39
scheduling with Task Scheduler, 428-430
Script Encoder, 21
security, 40-41, 408-409
code signing, 409-410, 412
Script Encoder, 415
security policies, 414
Trust Policy control, 42
Trust Policy control, 42
shortcuts
creating, 39, 183-186, 193-195
modifying, 193-195
signing, 412-413
stdin files, 159-161
stdout files, 159-162
unattended scripts
controlling logged information, 423
creating, 421-423
user-friendly scripts, creating, 376-377
versus batch files, 13
viewing
Call Stack window (Windows XP), 46
WMI (Windows Management Interface) scripts, 312
writing for other users, 376-377
Wscript objects, 111-114

searching
case insensitive searching (findstr command-line program), 549
files, 477
multiple files, 549

security, 40-41, 414
ADSI, 328-330
code signing, 409
certificate, obtaining, 410-411
scripts, signing, 412-413
signed scripts, requiring, 414
DCOM security, 294-300
file/folder privacy (cacls command), 563
permissions, granting (cacls command), 562
PowerShell, 615-617
Script Encoder, 415
script files, 40-42
Trust Policy control, 42
WMI
impersonation, 297-298
monikers, specifying security options, 300
privileges, 298-299

Security property (SWbemServices objects), 302

SeeAlso property (IADsOU object), 370

Select Case statement, 61-63

select queries (WQL), 304

selecting scripting language, 16

selectNodes methods
DOMDocument object, 172
IXMLDOMNode object, 174

selectSingleNode methods
DOMDocument object, 172
IXMLDOMNode object, 174

Send method (CDO.Message objects), 240

Sender property (CDO.Message objects), 238

sending
email
CDO (Collaboration Data Objects), 232-235, 261
HTML, 234
MIME (Multipart Internet Mail Extensions), 234
faxes from scripts, 271-277
HTML files, 259-260
messages to Event log, 423
printing messages, 425
results, summarizing, 425-427
multiformat messages, 260
output to network printers, 229-230
program output messages, 258
text file messages, 258
text string messages, 257-258
Web pages, 259-260

SendKeys method (WScript.Shell object), 186

sendusing fields (CDO.configuration objects), cdoSendUsing constants, 254

separating CMD arguments, 454

serial communications, configuring (NTVDM), 539

SerialNumber properties (Scripting.Drive object), 137

serverName property (RootDSE object), 368

servers
DNS server, testing, 589
email delivery servers, specifying, 263
SMTP servers, routing email, 264

service packs, monitoring, 314-315

ServiceAccountName property (IADsService object), 358

ServiceAccountPath property (IADsService object), 358

services
managing, 315-317
starting/stopping, 570-571

ServiceType property (IADsService object), 358

Sessions property (IADsFileService object), 346

Sessions property (IADsFileServiceOperations object), 346

set commands (CMD), 471
batch files, performing numerical calculations, 481
variables, setting, 480

set/a command expression operators, 734

setAttribute methods (IXMLDOMNode object), 175

SetDefaultPrinter methods(WSHNetwork objects), 211, 228

SetInfo method
IADs object, 336
IADsDomain object, 344
IADsService object, 359
IADsUser object, 363

setlocal command (CMD), 472

SetPassword method (IADsServiceOperations object), 360

setting
attributes (attrib command), 558-559
default environment variables, 461
default printers, 228
variables, 480

ShareName properties (Scripting.Drive object), 137

sharing scripts on multiple computers, 406-408

shells, 434

shift command (CMD), 472

shortcuts
creating, 38-39, 183-186, 193-196
modifying, 193-195
to CMD shell, creating, 436

ShortName properties
Scripting.File object, 145
Scripting.Folder object, 141

ShortPath properties
Scripting.File object, 145
Scripting.Folder object, 141
shutdown scripts, assigning through Group Policy, 418-421
siblings, 169
signed scripts
code signing, 409-413
requiring, 414
signtool.exe, downloading, 410
single-stepping, ISE, 709
Size properties (Scripting.Folder object), 141
Size property (IADsPrintJob object), 352
Skip methods (TextStream object), 152
SkipLine methods (TextStream object), 152
SMTP servers, routing email, 264
software development aids, 742
software interrupts, 522
sort program (Windows XP), 741
sorting listings, dir command (CMD), 479
special characters, CMD command-line processing, 454
special-purpose devices, running through NTVDM, 539-540
SpecialFolders properties (WScript.Shell object), 183
specified properties (IXMLDOCNode object), 176
specifying
delivery server for messages (CDO), 263-265
email delivery servers, 263
email subjects/recipients, 263
moniker security options, 300-301
splat operators (PowerShell), 672-673
standard error, 445
standard input/output, 160
filters, 161
user input, prompting for, 162
start command (CMD), 473-474
Start method (IADsServiceOperations object), 361
Started properties (Win32_Service objects), 315
starting/stopping services, sc command (management power tools), 570
starting/stopping services, 570-571
StartMode properties (Win32_Service objects), 316
StartName properties (Win32_Service objects), 316
StartService methods (Win32_Service objects), 316
StartTime property (IADsPrintJob object), 352
StartType property (IADsService object), 359
startup scripts, assigning through Group Policy, 418-421
StartupParameters property (IADsService object), 359
State properties (Win32_Service objects), 316
static member functions, calling (PowerShell), 673
static methods, 306
statistics, listing, 586
Status properties
Win32_Service objects, 316
WshScriptExec object, 189
IADsPrintJobOperations object, 353
IADsPrintQueueOperations object, 356
IADsServiceOperations object, 360
StdErr properties (WshScriptExec object), 189
stdin files, 159-161
StdIn properties (WshScriptExec object), 189
stdout files, 159-162
StdOut properties (WshScriptExec object), 189

Stop method (IADsServiceOperations object), 361

stopping runaway command-line programs, 440

StopService methods (Win32_Service objects), 316

storing batch files, 492

Stream object, 250

string constants (VBScripts), 52

string operators, PowerShell, 643-646

string-manipulation functions (VBScript), 71, 73-74
- extracting parts of strings, 72-73
- InStr functions, 71-72
- InStrRev functions, 72

strings
- joining
  - & character, 54-55, 75
  - + operators, 54-55
- literal string matching (findstr command-line programs), 549
- PowerShell, 628

structure of WSF files, 378

subdirectories, copying (xcopy command-line programs), 554-555

SubFolders properties (Scripting.Folder object), 141

subject, specifying in messages (CDO), 263

Subject property (CDO.Message objects), 238

subroutines, 69, 89, 513-514

substituting arguments (batch files), 496

summarizing Event Log results, 425-427

supportLDAPVersion property (RootDSE object), 369

SWbemLocator objects (WbemScripting objects), 288, 291-292

SWbemMethod objects (WbemScripting objects), 289, 308-309

SWbemmethodSet objects (WbemScripting objects), 289, 308

SWbemObject objects (WbemScripting objects), 289, 306-307

SWbemObjectSet objects (WbemScripting objects), 289, 305-306

SWbemProperty objects (WbemScripting objects), 289

SWbemPropertySet objects (WbemScripting objects), 289, 308-309

SWbemSecurity objects (WbemScripting objects), 288

SWbemServices objects (WbemScripting objects), 288
- Delete method, 302
- ExecMethod method, 302
- ExecQuery method, 302-305
- InstancesOf method, 303
- security property, 302
- WMI, specifying security options, 300

switch command, PowerShell, 657-660

switches. *See* named arguments

syntax for VBScript functions, 70

system information, collecting, 312-313

system-wide environment variables, 459-461

T

tabs, inserting into text files, 157

tags
- WSF files, 379-384
- XML tags, 168, 379-383

TargetPath properties (WshShortcut object), 194

TargetPath properties (WshUrlShortcut object), 194

Task Scheduler, scheduling scripts, 428-431

taskkill command (management power tools), 568-569

tasklist command (console programs), 441

tasklist command (management power tools), 565-567

tasks, managing, 315-317

TCP/IP utilities (Windows XP), 743

TelephoneNumber property (IADsO object), 370

TelephoneNumber property (IADsOU object), 370

Terminate method (WshScriptExec object), 189

terminating
applications in NTVDM, 540
command-line programs, 440
loops, 65-66
taskkill, 568-569

testing
DNS server, 589
file/folder attribute values, 142-143
multiple attributes, 143
scheduled scripts, 430-431

tests, performing with extended if command, 503

text
displaying, 84
files
reading from, 152-153
writing to, 154-156
parsing, 487-488
reading from console programs, 191-193
text files
creating, 157-159
inserting tabs, 157
writing Unix-compatible text files, 159
wildcards, matching (findstr command-line programs), 550-551

text string messages, sending, 257-258

text tags, XML (Extensible Markup Language), 168-169

TextBody property (CDO.Message objects), 238

TextBodyPart property (CDO.Message objects), 238

TextStream object (Windows Scripting Host), 150
methods
Close method, 152
Read method, 152, 163, 166
ReadAll method, 152
ReadLine method, 152-153
Skip method, 152
SkipLine method, 152
Write method, 152
WriteBlankLines method, 152, 159
WriteLine method, 152, 154, 156, 159
properties
AtEndOfLine properties, 151
AtEndOfStream properties, 151, 162
Column properties, 151
Line properties, 151

third-party script editing tools, 27

throw command, PowerShell, 664

time command (CMD), 474

Time constants (VBScript), 52

Time() function, 75

TimeElapsed property (IADsPrintJobOperations object), 353

TimeSubmitted property (IADsPrintJob object), 352

timezoneid fields (CDO.configuration objects), cdoTimeZoneId constants, 254-256

title command (CMD), 474

To property (CDO.Message objects), 238

TotalPages property (IADsPrintJob object), 352

TotalSize properties (Scripting.Drive object), 137

tracert command (networking utilities), 591-592

tracert command-line tool, 591-592

tracing scripts, 43

trap command, PowerShell, 662

tree command-line tool, 553-554

trees, containers versus leaves, 330-331

troubleshooting MS-DOS programs, 540-541

Trust Policy control (Windows XP), 42

type command (CMD), 474

Type properties (Scripting.File object), 145

Type properties (Scripting.Folder object), 141

U

UAC (User Account Control), 436
handling during installer script creation, 400-401

Ucase functions (VBScripts), 68

unattended backups, performing with xcopy, 556-557

unattended scripts
creating, 421-423
logged information, controlling, 423

UNC pathnames, 518

uninstall option, providing with installer script, 402

Unix-compatible text files, writing, 159

unnamed arguments, processing for WSF files, 389-390

Unnamed collection (command-line arguments) properties, 389

unsigned scripts, disabling, 408

UntilTime property
IADsPrintJob object, 352
IADsPrintQueue object, 355

updated files, copying (xcopy command-line programs), 555-556

UpdateProfile arguments (WSHNetwork objects), 209-211

user directories, 321
managing, 319
security, 328-330

user information (networks), displaying, 212-214

user input, prompting for, 162

user profile logon scripts, 416-418

User property
IADsPrintJob object, 352
IADsResource object, 346
IADsSession object, 361

user-friendly scripts, creating, 376-377

UserDomain properties (WSHNetwork objects), 208

UserName arguments (WSHNetwork objects), 209-210

UserName properties (WSHNetwork objects), 208

UserPath property
IADsPrintJob object, 352
IADsResource object, 346
IADsSession object, 361

V

validating batch file arguments, 519

values
AddRelatedBodyPart method constant values, 241
CDO.Configuration object values, 252-256
CreateMHTMLBody method constant values, 242
DSNOptions property constant values, 240-241
extracting from PowerShell arrays, 636-637

variables
dynamic environment variables, 728
environment variables
batch files, 506-508
expressions, 727
for command variable (CMD), 485-486, 726

PowerShell, 623-624
predefined, 630-632
releasing, 630
scope, 665, 667, 669
predefined environment variables, 727-728
scope, 91-92
Time variables, 716
VBScript, 50-51
arrays, 89-91
automatic conversion, 57
operators, 53-56, 717-718
syntax, 714-716

VBA, VBScript omitted features, 723

VBE extensions (script files), 20

VBScript, 14, 49-50
arrays, 89-91
automatic conversion, 57
collections, 67
constant definitions (CDO) Website, 241
constants, 51-53, 722
error handling, 86-87
expressions, syntax, 716
flow control, 57
Do While statement, 63-65
Exit Do statement, 65-66
For…Each statement, 68
For…Next statement, 66-67
If…Then statement, 58-61
Select Case statement, 61-63
functions, 720-722
built-in functions, 720
calling, 69
date and time, 75-78, 722
InputBox, 82-84
MsgBox, 79-82
named constants, 53
objects, 98-99
omitted VBA features, 723-724
string-manipulation, 71-74
syntax, 70
interpreters, 14
objects, 98-100
collections, 102-103
nested objects, 101-102
releasing, 102
omitted VBA features, 723
operators, 53, 717-718
arithmetic, 55
comparison, 55
logical, 56
precedence, 54
procedures
functions, creating, 87-88
subroutines, 89
program statements, 718-719
variables, 50-51
scope, 91-92
syntax, 714-716
Wscript.Echo command, 84-85

VBScript newsgroup Websites, 92

ver command (CMD), 475

verify on/off command (CMD), 475

verifying exit status of batch files, 501

Version property (IADsService object), 359

viewing
Call Stack window, 46-47
classes, 118
Registry, 115
script files, pipe mechanisms, 33
script variables, Script Debugger (Windows), 45
scripts, Call Stack window (Windows XP), 46
Windows Registry, 115

Virtual DOS Machine, 522-523
AUTOEXEC.NT file, 535-536
CONFIG.NT file, 532-535
configuring, 525-526
drive letters, mapping, 537
environment variables, 536
print redirection, 538
Print Screen function, 538
programs, terminating, 540
Properties dialog box
Compatibility tab, 532
Font tab, 528
Memory tab, 528
Miscellaneous Settings tab, 530-531
Program tab, 526-528
Screen tab, 530
serial communications, 539
special-purpose devices, running, 539-540

virtual environment variables, 459-461

vol command (CMD), 475

VolumeName properties (Scripting.Drive object), 137

W

WBEM (Web-Based Enterprise Management), 282

WbemScripting objects (WMI), 287-290, 302-309
- SWbemLocator objects, 288-289, 308
- SWbemLocator objects (Windows Management Interface), connecting to, 291-292
- SWbemMethod objects, 289
- SWbemObject objects, 289
  - *Delete method, 307*
  - *Instances method, 307*
  - *Methods properties, 306*
  - *Path properties, 307*
  - *Property properties, 307*
  - *Put method, 307*
- SWbemObjectSet objects, 289, 305-306
- SWbemProperty objects, 289
- SWbemPropertySet objects, 289, 308-309
- SWbemSecurity objects, 288
- SWbemServices objects, 288
  - *Delete method, 302*
  - *ExecMethod method, 302*
  - *ExecQuery method, 302-305*
  - *InstancesOf method, 303*
  - *security property, 302*
  - *specifying security options, 300*
- Win32_Service objects, 317
  - *ChangeStartMode methods, 316*
  - *ContinueService methods, 316*
  - *DesktopInteract properties, 315*
  - *DisplayName properties, 315*
  - *InterrogateService methods, 316*
  - *Name properties, 315*
  - *PathName properties, 315*
  - *PauseService methods, 316*
  - *Started properties, 315*
  - *StartMode properties, 316*
  - *StartName properties, 316*
  - *StartService methods, 316*
  - *State properties, 316*
  - *Status properties, 316*
  - *StopService methods, 316*

WbemScripting.SWbemLocator object, 291-292

web pages, sending, 259-260

Web-Based Enterprise Management standard, 282

Web-based enterprise management. *See* WBEM

websites
- ActiveState, 15
- ADSI, 374
- LDAP, 367
- Microsoft, 70
- Microsoft Developer's Network, 50
- Microsoft Developers Website, 232
- Microsoft TechNet, 50
- MIME (Multipart Internet Mail Extensions), 234
- Perl, 15
- Ruby, 15
- Script Debugger (Windows), 43
- Script Encoder, 415
- VBScript constant definitions (CDO), 241
- VBScript newsgroup Websites, 92
- WMI (Windows Management Interface), 317
- X.500, 367

while command, PowerShell, 654

wildcards, matching text (findstr command-line programs), 550-551

Win32_ComputerSystem object
- methods, 308-309
- properties, 308-309

Win32_Service objects (WbemScripting objects), 317
- ChangeStartmode methods, 316
- ContinueService methods, 316
- DesktopInteract properties, 315
- DisplayName properties, 315
- InterrogateService methods, 316
- Name properties, 315
- PathName properties, 315
- PauseService methods, 316

Started properties, 315
StartMode properties, 316
StartName properties, 316
StartService methods, 316
State properties, 316
Status properties, 316
StopService methods, 316

windows
Call Stack window (Windows XP), viewing scripts, 46
command prompt window, batch files, 513-520
console window, 442

Windows operating systems
programs, running, 187-188
remote management with WMI, 283-287

Windows 2000 Professional, remote management with WMI, 285

Windows 7
logon scripts, assigning, 418
remote management with WMI, 286-287
scripts, scheduling, 428-429

Windows Explorer, CMD shell, 435

Windows NT, remote management with WMI, 285

Windows objects, 94
ActivePerl, 106
collections, 108-109
Perl Object Interface, 107-108
running Perl scripts in Windows Script Host, 106-107
ActivePython, 109-110
collections, 110
automation, GetObject function, 99-100
classes, 95
collection objects, 96-97
ActivePerl, 108-109
ActivePython, 110
Count properties, 103
For Each loops, 103
Item methods, 103
JScript, 104-106
COM (Common Object Model) objects, 94
container objects, 96
CSLID, 116
defining, 10-11, 93
instances, 95
JScript
collections, 104-106
Enumerator objects, 105
methods
AddPrinterConnection, 208, 225-226, 229-230
AddWindowsPrinterConnection, 209, 223-224
defining, 93, 100
EnumNetworkDrives()$209, 214-217
EnumPrinterConnections()$209, 222-223
MapNetworkDrive, 210, 218-221
RemoveNetworkDrive, 210, 219-220
RemovePrinterConnection, 211, 226-228
SetDefaultPrinter, 211, 228
naming, 97
nested objects, 101-102
object browsers, 118
OLE/COM Object Viewer, 119
viewing classes, 118
OLE/COM Object Viewer, 119
properties
defining, 93, 100-101
releasing, 102
VBScript, 98-99
Windows Registry, viewing, 115
WScript, 111-114
Wscript.Network objects, managing printer connections, 221
WSHNetwork objects
ComputerName properties, 208
displaying network user information, 212-214
managing drive mappings, 214
UserDomain properties, 208
UserName properties, 208

Windows Registry, 201-202
information, saving, 203-205
viewing, 115

Windows Script Component files, 23

Windows Script Debugger, 43
altering script variable values, 45

displaying script function values, 45
function keys, 44
viewing script variables, 45
Windows Script files, 21-22
Windows Script Host, 10
ActivePerl, objects, 106-109
ActivePython, objects, 109-110
Cscript, 29
command options, 33-36
running script files, 30
running script files from batch files, 40
saving script files, output redirection, 33
security, 41
viewing script files, pipe mechanisms, 33
defining, 11-12
JScript, objects, 104-106
network connections, managing, 207-208
objects
CSLID, 116
object browsers, 118-119
object browsers, viewing classes, 118
OLE/COM Object Viewer, 119
viewing Windows Registry, 115
Perl scripts, running, 106-107
printer connections, managing, 207-208
script files
creating shortcuts, 39
running, 29
running from batch files, 39
security, 40-42
VBScript, 49, 713. *See* VBScript
Wscript. *See* Wscript
Windows Scripts
email
sending, 232-235
Windows service packs, monitoring, 313-315
Windows Vista
logon scripts, assigning, 418
remote management with WMI, 286
scripts, scheduling, 428-429
Windows XP
administrative tools, 736-738
batch file commands, 738-739
built-in commands, 738-739
Call Stack window, viewing scripts, 46
command-line programs
findstr, 547-551
GUI shortcuts, 545
more, 552-553
tree, 553
xcopy, 554-556
Control panel applets, 546-547
DOS commands, 739-740
file management tools
attrib command, 557-559
cacls command, 559-563
forcedos compatibility program, 524-525
GUI programs, 744-745
logoff program, 741
management power tools, 563
driverquery command, 564
sc command, 569-570
taskkill command, 568-569
tasklist command, 565-567
more program, 741
networking tools, 571, 741-742
ipconfig command, 571-573
net command, 574
net continue command, 574
net file command, 574
net help command, 575
net helpmsg command, 575
net localgroup command, 575
net pause command, 575
net print command, 575-576
net send command, 576
net session command, 576-577
net share command, 577
net start command, 578
net statistics command, 578
net stop command, 578
net use command, 579-581
net user command, 581-583
net view command, 583-584
netstat command, 584-586
nslookup command, 586-589
ping command, 589-590
tracert command, 591-592
NTVDM (Windows NT Virtual DOS Machine), 522-523
configuring, 525-536
configuring serial communications, 539

MS-DOS hardware support, 539
networking, 536-537
printing, 537
printing redirection, 538
Print Screen function, 538
scripts, scheduling, 430
software development aids, 742
sort program, 741
TCP/IP utilities, 743
WMI (Windows Management Interface), 279, 312
DCOM security, 294-300
Hotfixes, monitoring, 313-314
local computers, connecting to, 294
monikers, connecting with, 292-293
monikers, specifying security options, 300
namespaces, 281-283
printers, managing, 313
scripts, 312
services, managing, 315-317
tasks, managing, 315-317
WBEM (Web-Based Enterprise Management), 282, 287-292, 302-309
Windows Service Packs, monitoring, 313-314

Windows XP Home Edition, remote management with WMI, 285

Windows XP Professional, remote management with WMI, 285

WindowStyle properties (WshShortcut object), 194

WindowStyle properties (WshUrlShortcut object), 194

WinNT provider, 332-333
IADs object, 333-336
IADsCollection object, 336-338
IADsComputer object, 340-342
IADsComputerOperations object, 340-342
IADsContainer object, 336-338
IADsDomain object, 342-344
IADsFileService object, 345-347
IADsFileServiceOperations object, 345-347
IADsFileShare object, 347-348
IADsGroup object, 349-350
IADsMembers object, 350
IADsNamespaces object, 351
IADsPrintJob object, 351-354
IADsPrintJobOperations object, 351-354
IADsPrintQueue object, 354-357
IADsPrintQueueOperations object, 354-357
IADsService object, 357-361
IADsServiceOperations object, 357-361
IADsSession object, 361-362
IADsUser object, 362-363

With statements (VBScript), 102

Wizards, Schedule Task Wizard, 430

WMI (Windows Management Interface), 279, 312
authentication, 295-296
DCOM security, 294
authentication, 295-296
encryption, 295-296
impersonation, 295-297
permission control, 298-299
specifying options, 299-300
Windows remote management, 283-287
encryption, 295-296
example scripts
printers, managing, 313
system information, collecting, 312-313
tasks, managing, 315-317
Windows service packs, monitoring, 314-315
functions, 281
hotfixes, monitoring, 313-314
local computers, connecting to, 294
namespaces, 281-283
objects, 288-289, 291
monikers, 292-293
SWbemMethod, 308-309
SWbemObject, 306-307
WbemScripting.SWbemLocator, 291-292
Win32_ComputerSystem, 308-309
Scriptomatic, downloading, 310
security
impersonation, 297-298
monikers, 300-301

privileges, 298-299
SWbemObjectSet collection object, 305-306
SWbemServices object
methods, 302-303
properties, 302
security options, specifying, 300
WBEM (Web-Based Enterprise Management), 282
WbemScripting objects, 287-292, 302-309
Win32_Service object
methods, 316
properties, 315-316
Windows Service Packs, monitoring, 313-314
WQL queries, 303-305

workgroup networks, remote management with WMI, 284-287

WorkingDirectory properties (WshShortcut object), 194

WorkingDirectory properties (WshUrlShortcut object), 194

workstations, requiring signed scripts, 414

WQL queries
WMI (Windows Management Interface), 303-305

Write methods (TextStream object), 152

WriteBlankLines methods (TextStream object), 152, 159

WriteLine methods (TextStream object), 152-159

writing
files, 149
scripts
ADSI, 370-371
for other users, 376-377
remote-management, 403-405
Scriptomatic, 310-311
unattended scripts, 421, 423
uninstall scripts, 402
text files, Unix-compatible, 159
text to files, 154-156

WSC (Windows Script Component) files, 23

WScript
arguments, 31-32
command options, 33-36
methods, 112-113
objects, 111-114
script files, running from batch files, 40
script security, 41
script shortcuts, creating, 39

Wscript.Echo command (VBScript), 84-85

WScript.Network object
default printer, setting, 228
DOS printer sessions, redirecting, 225-226
drive mappings, 218-221
network connections, managing, 207
network user information, retrieving, 212-214
printers
connecting to, 223-228
information, displaying, 222-223

Wscript.Shell object
Environment property, 196-197
methods
AppActivate method, 183-184
CreateShortcut method, 184
Exec method, 184
ExpandEnvironmentStrings method, 184
LogEvent method, 184
Popup method, 184-185
RegDelete method, 186
RegRead method, 186, 202
RegWrite method, 186, 203
Run method, 186
SendKeys method, 186
properties
CurrentDirectory properties, 182
Environment properties, 182-183, 196-197
SpecialFolders properties, 183

WSF (Windows Script Files), 21-22
command-line arguments
extracting named arguments, 387
named arguments, processing, 386-388
processing, 386
processing named arguments, 386
processing unnamed arguments, 389

unnamed arguments, processing, 389-390
creating, 377-378, 390, 394
example script, 390-394
formats, 379-383
multiple scripts, enclosing, 390
named arguments (command-line arguments)
extracting, 387
processing, 386
online help, providing, 384-385
structure, 378
tags, 379-384
unnamed arguments (command-line arguments), processing, 389
XML tags, 379-383

WSHNetwork objects
methods
AddPrinterConnection, 208, 225-226, 229-230
AddWindowsPrinterConnection, 209, 223-224
EnumNetworkDrives, 209, 214-217
EnumPrinterConnections, 209, 222-223
MapNetworkDrive, 210, 218-221
RemoveNetworkDrive, 210, 219-220
RemovePrinterConnection, 211, 226-228
SetDefaultPrinter, 211, 228
properties, 208

WshScriptExec object (Windows Scripting Host)
methods, Terminate method, 189
properties
ExitCode properties, 188
ProcessID properties, 189
Status properties, 189
StdErr properties, 189
StdIn properties, 189
StdOut properties, 189

WshShortcut object (Windows Scripting Host)
methods, Save methods, 194
properties
Arguments properties, 193
Description properties, 194
FullName properties, 194
Hotkey properties, 194
IconLocation properties, 194
TargetPath properties, 194
WindowStyle properties, 194
WorkingDirectory properties, 194

WshUrlShortcut object (Windows Scripting Host)
methods, Save methods, 194
properties
Arguments properties, 193
Description properties, 194
FullName properties, 194
Hotkey properties, 194
IconLocation properties, 194
TargetPath properties, 194
WindowStyle properties, 194
WorkingDirectory properties, 194

X-Y-Z

X.500, 364-367

xcopy (command-line programs), 554, 557
backups, 555
subdirectories, copying, 554
unattended backups, 556
updated files, copying, 555-556

XML (Extensible Markup Language), 167
DTD, 168
elements, 168
files
creating, 179-181
reading, 177-178
nodes, 169
siblings, 169
tags, 168-169
WSF files, 377
named arguments, processing, 386-388
online help, providing, 384-385
structure, 378
tags, 379-384
unnamed arguments, processing, 389-390
text/markup tags, 168-169

xml properties
DOMDocument object, 171
IXMLDOMNode object, 174