



WINDOWS PowerShell IN ACTION

Bruce Payette

Foreword by Jeffrey Snover

 MANNING

contents

foreword xv
preface xvii
acknowledgments xix
about this book xx

Part 1 LEARNING POWERSHELL 1

- 1 Welcome to PowerShell 3***
 - 1.1 What is PowerShell? 5
 - Shells, command-lines, and scripting languages 5 ♦ Why a new shell?
 - Why now? 7 ♦ The last mile problem 7
 - 1.2 Soul of a new language 8
 - Learning from history 8 ♦ Leveraging .NET 9
 - 1.3 Brushing up on objects 10
 - Reviewing object-oriented programming 11
 - Objects in PowerShell 12
 - 1.4 Dude! Where's my code? 13
 - Installing and starting PowerShell 13 ♦ Command editing 15
 - Command completion 16 ♦ Evaluating basic expressions 17
 - Processing data 18
 - 1.5 Summary 23
- 2 The basics 25***
 - 2.1 Command concepts and terminology 27
 - Commands and cmdlets 27 ♦ Command categories 30
 - Aliases and elastic syntax 34
 - 2.2 Parsing and PowerShell 37
 - How PowerShell parses 37 ♦ Quoting 38 ♦ Expression mode and command mode parsing 41 ♦ Statement termination 43

2.3	Pipelines and commands	45
	Pipelines and streaming behavior	46
	Parameters and parameter binding	47
2.4	Formatting and output	48
	The formatting cmdlets	49 ♦ The outputter cmdlets
		51
2.5	Summary	54
3	<i>Working with types</i>	55
3.1	Type management in the wild, wild west	55
	PowerShell: a type-promiscuous language	56
	The type system and type adaptation	58
3.2	Basic types and literals	60
	Strings	60 ♦ Numbers and numeric literals
		64 ♦ Collections: dictionaries and hashtables
		66 ♦ Collections: arrays and sequences
		71 ♦ Type literals
		75
3.3	Type conversions	79
	How type conversion works	79 ♦ PowerShell's type-conversion algorithm
		82 ♦ Special type conversions in parameter binding
		85
3.4	Summary	86
4	<i>Operators and expressions</i>	87
4.1	Arithmetic operators	89
	The addition operator	89 ♦ The multiplication operator
		92
	Subtraction, division, and the modulus operator	94
4.2	The assignment operators	96
	Multiple assignments	97 ♦ Multiple assignments with type qualifiers
		98
	Assignment operations as value expressions	100
4.3	Comparison operators	101
	Scalar comparisons	102 ♦ Using comparison operators with collections
		105
4.4	The pattern matching operators	107
	Wildcard patterns	107 ♦ Regular expressions
		108
4.5	Logical and bitwise operators	113
4.6	Summary	113
5	<i>Advanced operators and variables</i>	115
5.1	Operators for working with types	115
5.2	The unary operators	117
5.3	Grouping, subexpressions, and array subexpressions	119
5.4	Array operators	123
	The comma operator “,”	123 ♦ The range operator
		126
	Array indexing	127

5.5	Property and method operators	132
	The “.” operator	133 ♦ Static methods and the “::” operator 136
5.6	The PowerShell format operator -F	137
5.7	Redirection and the redirection operators	138
5.8	Variables	141
5.9	Summary	145
6	<i>Flow control in scripts</i>	147
6.1	Using the if/elseif/else statement	148
6.2	The while loop	151
6.3	The do/while loop	152
6.4	The for loop	153
6.5	The foreach loop	155
6.6	Labels, break, and continue	159
6.7	The PowerShell switch statement	161
	Basic use of the PowerShell switch statement	161 ♦ Using wildcard patterns with the switch statement 163 ♦ Using regular expressions with the switch statement 164 ♦ Processing files with the switch statement 167
	Using the \$switch loop enumerator in the switch statement	168
6.8	Flow control using cmdlets	169
	The Foreach-Object cmdlet	170 ♦ The Where-Object cmdlet 173
6.9	The value of statements	175
6.10	Summary	176
7	<i>Functions and scripts</i>	177
7.1	Function basics	178
7.2	Formal parameters and the param statement	181
	Specifying parameter types	183 ♦ Handling variable numbers of arguments 185 ♦ Initializing function parameters 186 ♦ Using switch parameters to define flags 188 ♦ Variables and scoping rules 190
	Using variable scope modifiers	193
7.3	Returning values from functions	193
	Debugging function output	196 ♦ The return statement 198
7.4	Using functions in a pipeline	199
	Filters and functions	201 ♦ Functions as cmdlets 202
7.5	Managing functions	204

7.6	Scripts at long last	205
	Passing arguments to scripts	207 ♦ The param statement
	and scripts	208 ♦ Scopes and scripts
	208 ♦ Exiting scripts and the exit statement	209
	Dotting scripts and functions	210
7.7	Summary	212
8	<i>Scriptblocks and objects</i>	214
8.1	Scriptblock basics	215
	Invoking commands	216 ♦ Getting CommandInfo objects
	217 ♦ The ScriptBlock literal	219 ♦ Defining functions at runtime
	220	
8.2	Building and manipulating objects	222
	Looking at members	222 ♦ Synthetic members
	223 ♦ Using Add-Member to extend objects	224 ♦ Using the select-object cmdlet
	230	
8.3	A closer look at the type-system plumbing	233
	Adding a property	235 ♦ Shadowing an existing property
	236	
8.4	Extending the PowerShell language	237
	Little languages	237 ♦ Adding a CustomClass keyword to PowerShell
	238	
8.5	Type extension	243
8.6	Building code at runtime	245
	The Invoke-Expression cmdlet	245 ♦ The ExecutionContext variable
	246	
	Creating functions using the function: drive	248
8.7	Summary	249
9	<i>Errors, exceptions, and script debugging</i>	251
9.1	Error handling	252
	ErrorRecords and the error stream	253 ♦ The \$Error variable and
	-ErrorVariable parameter	256 ♦ The \$? and \$LASTEXITCODE
	variables	259 ♦ \$ErrorActionPreference and the -ErrorAction
	parameter	261
9.2	Dealing with errors that terminate execution	265
	The trap statement	265 ♦ The throw statement
	268	
9.3	Script debugging	270
	Debugging with the host APIs	270 ♦ The Set-PSDebug cmdlet
	271	
	Tracing statement execution	271 ♦ Stepping through statement
	execution	275 ♦ Catching undefined variables with strict mode
	276	
9.4	Nested prompts and breakpoints	277
	Suspending a script while in step-mode	277 ♦ Creating a breakpoint
	command	279 ♦ The script call stack, or “How did I get here?”
	281	
9.5	Low-level tracing	283
	The Trace-Command cmdlet	283 ♦ Tracing type conversions
	285	
	Tracing parameter binding	287

- 9.6 The PowerShell event log 291
 - Examining the event log 291
 - Exchange 2007 and the PowerShell event log 293
- 9.7 Summary 293

Part 2 USING POWERSHELL 295

10 Processing text, files, and XML 297

- 10.1 Processing unstructured text 298
 - Using System.String to work with text 298
 - Using regular expressions to manipulate text 304
- 10.2 File processing 305
 - Working with PSDrives 307 ♦ Working with paths that contain wildcards 309 ♦ Reading and writing files 313
 - Searching files with the Select-String cmdlet 319
- 10.3 XML processing 322
 - Using XML as objects 322 ♦ Loading and saving XML files. 326
 - Processing XML documents in a pipeline 333 ♦ Using XPath 334
 - The Import-Clixml and Export-Clixml cmdlets 339
- 10.4 Summary 342

11 Getting fancy—.NET and WinForms 344

- 11.1 Using .NET from PowerShell 345
 - .NET basics 345 ♦ Working with assemblies 346 ♦ Finding types 348
 - Creating instances of types 350 ♦ PowerShell is not C#—A cautionary tale 353 ♦ Working with generic types 358
- 11.2 PowerShell and the Internet 361
 - Example: Retrieving a web page 361 ♦ Example: Processing an RSS feed 362 ♦ Example: Writing a web server in PowerShell 364
- 11.3 PowerShell and graphical user interfaces 371
 - WinForms basics 371 ♦ Example: "My first form" 372 ♦ Example: Simple dialog 374 ♦ Example: A WinForms library 376 ♦ Example: A simple calculator 379 ♦ Example: Displaying data 385
 - Example: Using the GDI+ to do graphics 387
- 11.4 Summary 391

12 Windows objects: COM and WMI 392

- 12.1 Working with COM in PowerShell 393
 - Automating Windows with COM 396 ♦ Networking, applications, and toys 405 ♦ Using the ScriptControl object 415
 - Issues with COM 417

12.2	Working with WMI in PowerShell	421
	Exploring WMI—what is it, and why do you care?	421 ♦ The Get-WmiObject cmdlet 422 ♦ The WMI object adapter 423 ♦ WMI shootout—VBScript versus PowerShell 425 ♦ The WMI type shortcuts 429
	Working with WMI methods	432 ♦ Working with WMI events 433
	Putting modified WMI objects back	434
12.3	So which object model should I choose?	437
12.4	Summary	437
13	<i>Security, security, security</i>	440
13.1	Introduction to security	441
	What security is	441 ♦ What security is not 441
	Perception and security	442
13.2	Security modeling	443
	Introduction to threat modeling	444 ♦ Classifying threats using the STRIDE model 444 ♦ Security basics: Threats, assets, and mitigations 445
13.3	Securing the PowerShell environment	449
	Secure by default	449 ♦ Managing the command path 450
	Choosing a script execution policy	451
13.4	Signing scripts	453
	How public key encryption and one-way hashing work	453 ♦ Signing authorities and certificates 454 ♦ Creating a self-signed certificate 455
	Using a certificate to sign a script	458 ♦ Enabling strong private key protection for your certificate 462
13.5	Writing secure scripts	465
	Using the SecureString class	465 ♦ Working with credentials 468
	Avoiding Invoke-Expression	471
13.6	Summary	474
<i>appendix A</i>	<i>Comparing PowerShell to other languages</i>	<i>476</i>
<i>appendix B</i>	<i>Admin examples</i>	<i>499</i>
<i>appendix C</i>	<i>The PowerShell grammar</i>	<i>520</i>
<i>index</i>		<i>531</i>