

brief contents

- 1 ■ Welcome to PowerShell 1
- 2 ■ Working with types 46
- 3 ■ Operators and expressions 81
- 4 ■ Advanced operators and variables 114
- 5 ■ Flow control in scripts 154
- 6 ■ PowerShell functions 185
- 7 ■ Advanced functions and scripts 220
- 8 ■ Using and authoring modules 270
- 9 ■ Module manifests and metadata 314
- 10 ■ Metaprogramming with scriptblocks and dynamic code 351
- 11 ■ PowerShell remoting 405
- 12 ■ PowerShell workflows 458
- 13 ■ PowerShell Jobs 499
- 14 ■ Errors and exceptions 528
- 15 ■ Debugging 560
- 16 ■ Working with providers, files, and CIM 604
- 17 ■ Working with .NET and events 661
- 18 ■ Desired State Configuration 711
- 19 ■ Classes in PowerShell 761
- 20 ■ The PowerShell and runspace APIs 796

contents

<i>preface</i>	<i>xxi</i>
<i>acknowledgments</i>	<i>xxiii</i>
<i>about this book</i>	<i>xxv</i>
<i>about the cover illustration</i>	<i>xxxii</i>

1 Welcome to PowerShell 1

- 1.1 What is PowerShell? 4
 - Shells, command lines, and scripting languages* 4
- 1.2 PowerShell example code 6
 - Navigation and basic operations* 7 ■ *Basic expressions and variables* 9
 - Processing data* 10 ■ *Flow-control statements* 13 ■ *Scripts and functions* 14
 - Remote administration* 15
- 1.3 Core concepts 17
 - Command concepts and terminology* 18 ■ *Commands and cmdlets* 18
 - Command categories* 20 ■ *Aliases and elastic syntax* 23
- 1.4 Parsing the PowerShell language 26
 - How PowerShell parses* 27 ■ *Quoting* 27 ■ *Expression-mode and command-mode parsing* 29 ■ *Statement termination* 31 ■ *Comment syntax in PowerShell* 33
- 1.5 How the pipeline works 35
 - Pipelines and streaming behavior* 35 ■ *Parameters and parameter binding* 37

- 1.6 Formatting and output 39
 - Formatting cmdlets* 40 ▪ *Outputter cmdlets* 42
- 1.7 Summary 45

2 *Working with types* 46

- 2.1 Type management in the wild, wild West 47
 - Types and classes* 47 ▪ *PowerShell: A type-promiscuous language* 49
 - Type system and type adaptation* 51 ▪ *Finding the available types* 53
- 2.2 Basic types and literals 55
 - String literals* 55 ▪ *Numbers and numeric literals* 59
- 2.3 Collections: dictionaries and hashtables 61
 - Creating and inspecting hashtables* 61 ▪ *Ordered hashtables* 64
 - Modifying and manipulating hashtables* 65 ▪ *Hashtables as reference types* 66
- 2.4 Collections: arrays and sequences 67
 - Collecting pipeline output as an array* 68 ▪ *Array indexing* 68
 - Polymorphism in arrays* 68 ▪ *Arrays as reference types* 69
 - Singleton arrays and empty arrays* 70
- 2.5 Type literals 71
 - Type name aliases* 72 ▪ *Generic type literals* 72 ▪ *Accessing static members with type literals* 73
- 2.6 Type conversions 74
 - How type conversion works* 74 ▪ *PowerShell's type-conversion algorithm* 75
 - Special type conversions in parameter binding* 78
- 2.7 Summary 80

3 *Operators and expressions* 81

- 3.1 Arithmetic operators 83
 - Addition operator* 84 ▪ *Multiplication operator* 86 ▪ *Subtraction, division, and the modulus operators* 87
- 3.2 Assignment operators 88
 - Multiple assignments* 89 ▪ *Multiple assignments with type qualifiers* 89
 - Assignment operations as value expressions* 90
- 3.3 Comparison operators 91
 - Scalar comparisons* 92 ▪ *Comparisons and case sensitivity* 93
 - Using comparison operators with collections* 95
- 3.4 Pattern matching and text manipulation 97
 - Wildcard patterns and the -like operator* 98 ▪ *Regular expressions* 99
 - The -match operator* 100 ▪ *The -replace operator* 102 ▪ *The -join operator* 104 ▪ *The -split operator* 107

- 3.5 Logical and bitwise operators 109
- 3.6 Where() and ForEach() methods 110
 - Where() method* 111 ▪ *ForEach() method* 112
- 3.7 Summary 113

4 **Advanced operators and variables 114**

- 4.1 Operators for working with types 115
- 4.2 Unary operators 117
- 4.3 Grouping and subexpressions 119
 - Subexpressions \$(...)* 120 ▪ *Array subexpressions @(...)* 121
- 4.4 Array operators 123
 - Comma operator* 123 ▪ *Range operator* 126 ▪ *Array indexing and slicing* 127 ▪ *Using the range operator with arrays* 129
 - Working with multidimensional arrays* 130
- 4.5 Property and method operators 132
 - Dot operator* 132 ▪ *Static methods and the double-colon operator* 136
 - Indirect method invocation* 138
- 4.6 Format operator 139
- 4.7 Redirection and redirection operators 140
- 4.8 Working with variables 143
 - Creating variables* 143 ▪ *Variable name syntax* 145 ▪ *Working with variable cmdlets* 147 ▪ *Splatting a variable* 150
- 4.9 Summary 152

5 **Flow control in scripts 154**

- 5.1 Conditional statement 156
- 5.2 Looping statements 158
 - while loop* 158 ▪ *do-while loop* 159 ▪ *for loop* 160
 - foreach loop* 160
- 5.3 Labels, break, and continue 164
- 5.4 switch statement 166
 - Basic use of the switch statement* 166 ▪ *Using wildcard patterns with the switch statement* 167 ▪ *Using regular expressions with the switch statement* 168 ▪ *Processing files with the switch statement* 171
 - Using the \$switch loop enumerator in the switch statement* 172
- 5.5 Flow control using cmdlets 173
 - ForEach-Object cmdlet* 173 ▪ *Where-Object cmdlet* 178
- 5.6 Statements as values 181

5.7 A word about performance 182

5.8 Summary 184

6 PowerShell functions 185

6.1 Fundamentals of PowerShell functions 186

Passing arguments using \$args 187 ▪ *Example functions: ql and qs* 189

6.2 Declaring formal parameters for a function 190

Mixing named and positional parameters 191 ▪ *Adding type constraints to parameters* 192 ▪ *Handling variable numbers of arguments* 194

Initializing function parameters with default values 195 ▪ *Using switch parameters to define command switches* 196 ▪ *Switch parameters vs.*

Boolean parameters 199

6.3 Returning values from functions 204

Debugging problems in function output 206 ▪ *The return statement* 208

6.4 Using simple functions in a pipeline 209

Functions with begin, process, and end blocks 211

6.5 Managing function definitions in a session 212

6.6 Variable scoping in functions 214

Declaring variables 214 ▪ *Using variable scope modifiers* 217

6.7 Summary 218

7 Advanced functions and scripts 220

7.1 PowerShell scripts 221

Script execution policy 221 ▪ *Passing arguments to scripts* 223

Exiting scripts and the exit statement 225 ▪ *Scopes and scripts* 226

Managing your scripts 228 ▪ *Running PowerShell scripts from other applications* 229

7.2 Writing advanced functions and scripts 230

Specifying script and function attributes 231 ▪ *The CmdletBinding attribute* 232

▪ *The OutputType attribute* 238 ▪ *Specifying parameter attributes* 240

▪ *Creating parameter aliases with the Alias attribute* 248
▪ *Parameter validation attributes* 249

7.3 Dynamic parameters and dynamicParam 256

Steps for adding a dynamic parameter 256 ▪ *When should dynamic parameters be used?* 258

7.4 Cmdlet default parameter values 258

Creating default values 259 ▪ *Modifying default values* 260

Using scriptblocks to determine default value 262

- 7.5 Documenting functions and scripts 262
 - Automatically generated help fields* 263
 - *Creating manual help content* 264
 - Comment-based help* 264
 - *Tags used in documentation comments* 266
- 7.6 Summary 268

8 Using and authoring modules 270

- 8.1 The role of a module system 271
 - Module roles in PowerShell* 271
 - *Module mashups: composing an application* 272
- 8.2 Module basics 273
 - Module terminology* 274
 - *Modules are single-instance objects* 274
- 8.3 Working with modules 275
 - Finding modules on the system* 275
 - *Loading a module* 278
 - Removing a loaded module* 283
- 8.4 Writing script modules 286
 - A quick review of scripts* 286
 - *Turning a script into a module* 289
 - Controlling member visibility with Export-ModuleMember* 291
 - Installing a module* 295
 - *How scopes work in script modules* 302
 - Nested modules* 303
- 8.5 Binary modules 307
 - Creating a binary module* 308
 - *Nesting binary modules in script modules* 310
- 8.6 Summary 313

9 Module manifests and metadata 314

- 9.1 Module folder structure 315
- 9.2 Module manifest structure 316
- 9.3 Production manifest elements 321
 - Module identity* 322
 - *Runtime dependencies* 322
- 9.4 Construction manifest elements 323
 - The loader manifest elements* 325
 - *Module component load order* 328
- 9.5 Content manifest elements 329
- 9.6 Advanced module operations 330
 - The PSModuleInfo object* 330
 - *Using the PSModuleInfo methods* 334
 - The defining module vs. the calling module* 337
 - *Setting module properties from inside a script module* 340
 - *Controlling when modules can be unloaded* 341
 - *Running an action when a module is removed* 342

- 9.7 Publishing a module to a PowerShell Gallery 343
 - A module to publish* 344
 - PSData Packaging elements* 345
 - Publishing a module* 348
 - Publishing module updates* 349
- 9.8 Summary 350

10 **Metaprogramming with scriptblocks and dynamic code** 351

- 10.1 Scriptblock basics 352
 - Invoking commands* 353
 - Getting CommandInfo objects* 353
 - The scriptblock literal* 355
 - Defining functions at runtime* 357
- 10.2 Building and manipulating objects 358
 - Looking at members* 359
 - Defining synthetic members* 360
 - Using Add-Member to extend objects* 361
 - Adding note properties with New-Object* 367
- 10.3 Using the Select-Object cmdlet 370
- 10.4 Dynamic modules 372
 - Dynamic script modules* 372
 - Closures in PowerShell* 374
 - Creating custom objects from modules* 378
- 10.5 Steppable pipelines 379
 - How steppable pipelines work* 379
 - Creating a proxy command with steppable pipelines* 381
- 10.6 A closer look at the type-system plumbing 384
 - Adding a property* 386
 - Shadowing an existing property* 388
- 10.7 Extending the PowerShell language 389
 - Little languages* 389
 - Type extension* 390
- 10.8 Building script code at runtime 394
 - The Invoke-Expression cmdlet* 394
 - The ExecutionContext variable* 395
 - The ExpandString() method* 396
 - The InvokeScript() method* 397
 - Mechanisms for creating scriptblocks* 397
 - Creating functions using the function: drive* 398
- 10.9 Compiling code with Add-Type 399
 - Defining a new .NET class: C#* 400
 - Defining a new enum at runtime* 400
 - Dynamic binary modules* 402
- 10.10 Summary 403

11 **PowerShell remoting** 405

- 11.1 PowerShell remoting overview 406
 - Commands with built-in remoting* 406
 - The PowerShell remoting subsystem* 408
 - Enabling remoting* 409
 - Additional setup steps for workgroup environments* 411
 - Authenticating the connecting user* 412
 - Enabling remoting in the enterprise* 413

- 11.2 Applying PowerShell remoting 413
 - Basic remoting examples* 414 ▪ *Adding concurrency to the examples* 415
 - Solving a real problem: multi-machine monitoring* 416
- 11.3 PowerShell remoting sessions and persistent connections 419
 - Additional session attributes* 421 ▪ *Using the New-PSSession cmdlet* 422 ▪ *Interactive sessions* 423 ▪ *Managing PowerShell sessions* 425 ▪ *Copying files across a PowerShell remoting session* 428
- 11.4 Implicit remoting 429
 - Using implicit remoting* 430 ▪ *How implicit remoting works* 432
- 11.5 Considerations when running commands remotely 435
 - Remote session startup directory* 435 ▪ *Profiles and remoting* 435
 - Issues running executables remotely* 437 ▪ *Using files and scripts* 437
 - Using local variables in remote sessions* 438 ▪ *Reading and writing to the console* 439 ▪ *Remote output vs. local output* 440 ▪ *Processor architecture issues* 441
- 11.6 Building custom remoting services 442
 - Working with custom configurations* 443 ▪ *Creating a custom configuration* 444 ▪ *Access controls and endpoints* 446
 - Constraining a PowerShell session* 448
- 11.7 PowerShell Direct 455
- 11.8 Summary 456

12 PowerShell workflows 458

- 12.1 Workflow overview 459
 - Why use workflows* 459 ▪ *Workflow architecture* 460
 - Your first workflow* 462 ▪ *Running a workflow* 466
 - Cmdlets vs. activities* 467 ▪ *Workflow restrictions* 471
- 12.2 Workflow keywords 472
 - Parallel* 473 ▪ *Sequence* 474 ▪ *InlineScript* 475
 - Foreach -parallel* 477
- 12.3 Using workflows effectively 479
 - Workflow parameters* 480 ▪ *Variables in workflows* 483
 - Nested workflows* 485
- 12.4 Workflow cmdlets 489
 - Workflow execution options* 490 ▪ *Workflow sessions* 493
 - Invoking as workflow* 497
- 12.5 Summary 497

13 PowerShell Jobs 499

13.1 Background jobs in PowerShell 500

The job commands 501 ▪ *Working with the job cmdlets* 502
Working with multiple jobs 507 ▪ *Starting jobs on remote computers* 508
Running jobs in existing sessions 511 ▪ *Job types* 512

13.2 Workflows as jobs 514

Checkpoints 514 ▪ *Suspending workflows* 517 ▪ *Workflows and reboots* 518

13.3 Scheduled jobs 522

Creating scheduled jobs 522 ▪ *Modifying a scheduled job* 524
Managing scheduled jobs 525

13.4 Summary 526

14 Errors and exceptions 528

14.1 Error handling 529

ErrorRecords and the error stream 530 ▪ *The \$Error variable and -ErrorVariable parameter* 535 ▪ *Determining whether a command had an error* 540 ▪ *Controlling the actions taken on an error* 542

14.2 Dealing with errors that terminate execution 546

The try/catch/finally statement 547 ▪ *The throw statement* 550

14.3 PowerShell and the event log 551

The EventLog cmdlets 551 ▪ *Examining the PowerShell event log* 555
Get-WinEvent 557

14.4 Summary 559

15 Debugging 560

15.1 Script instrumentation 561

The Write cmdlets* 561 ▪ *Writing events to the event Log* 568
Catching errors with strict mode 569 ▪ *Static analysis of scripts* 574

15.2 Capturing session output 578

Starting the transcript 579 ▪ *What gets captured in the transcript* 580

15.3 PowerShell script debugging features 582

The Set-PSDebug cmdlet 583 ▪ *Nested prompts and the Suspend operation* 586

15.4 Command-line debugging 589

Working with breakpoint objects 590 ▪ *Setting breakpoints on commands* 593
Setting breakpoints on variable assignment 594 ▪ *Debugger limitations and issues* 595

- 15.5 Beyond scripts 596
 - Debugging PowerShell jobs* 596 ▪ *Debugging remote scripts* 599
 - Debugging PowerShell runspaces* 600

15.6 Summary 602

16 **Working with providers, files, and CIM 604**

16.1 PowerShell providers 605

- PowerShell core cmdlets* 606 ▪ *Working with PSDrives* 607
- Working with paths* 608 ▪ *The Registry provider* 612

16.2 Files, text, and XML 614

- File processing* 614 ▪ *Unstructured text* 618 ▪ *XML structured text processing* 625 ▪ *Converting text output to objects* 640

16.3 Accessing COM objects 644

16.4 Using CIM 652

- The CIM cmdlets* 653 ▪ *CIM sessions* 658

16.5 Summary 660

17 **Working with .NET and events 661**

17.1 .NET and PowerShell 662

- Using .NET from PowerShell* 662 ▪ *PowerShell and GUIs* 668

17.2 Real-time events 677

- Foundations of event handling* 677 ▪ *Synchronous events* 679
- Asynchronous events* 682 ▪ *Working with asynchronous .NET events* 684
- Asynchronous event handling with scriptblocks* 688 ▪ *Automatic variables in the event handler* 688 ▪ *Dynamic modules and event handler state* 690
- Queued events and the Wait-Event cmdlet* 691 ▪ *Working with CIM events* 693 ▪ *Class-based CIM event registration* 694 ▪ *Engine events* 701 ▪ *Generating events in functions and scripts* 702
- Remoting and event forwarding* 703 ▪ *How eventing works* 707

17.3 Summary 710

18 **Desired State Configuration 711**

18.1 DSC model and architecture 712

- The need for configuration management* 712 ▪ *Desired State Configuration model* 713 ▪ *DSC architecture* 718

18.2 Push mode to a single node 719

- Create configuration* 719 ▪ *MOF file contents* 720
- Applying the configuration* 722 ▪ *Testing the configuration*

application 724 ▪ *Viewing the current configuration* 725
Removing a configuration 726

18.3 Pushing to multiple nodes 727

Parameterizing the computer name 727 ▪ *Using configuration data* 729
Configuration data and roles 731 ▪ *Issues with push mode* 733

18.4 DSC in pull mode 734

Pull server architecture 734 ▪ *Creating a pull server* 735
Publishing a MOF file 741

18.5 Configuring the Local Configuration Manager 744

LCM settings 745 ▪ *Configuring LCM to use a pull server* 748

18.6 Partial configurations 751

Partial configurations: yes or no 751 ▪ *Pushing partial configurations* 752 ▪ *Pulling partial configurations* 757

18.7 Summary 760

19 **Classes in PowerShell 761**

19.1 Writing classes in PowerShell 762

Using properties in a PowerShell class 762 ▪ *Class member attributes* 766
PowerShell enumerations 768

19.2 Methods in PowerShell classes 770

Method basics 771 ▪ *Static methods* 771 ▪ *Instance methods* 773
Method overloads 776 ▪ *Hidden methods* 778 ▪ *Constructors in PowerShell classes* 779

19.3 Extending existing classes 782

Creating a derived class 782 ▪ *Overriding members on the base class* 784
Extending .NET classes 786

19.4 Classes, modules, using, and namespaces 787

19.5 Writing class-based DSC resources 790

19.6 Summary 795

20 **The PowerShell and runspace APIs 796**

20.1 PowerShell API basics 797

Multi-command pipelines 798 ▪ *Building pipelines incrementally* 799
Handling execution errors 801 ▪ *Adding scripts and statements* 803

20.2 Runspaces and the PowerShell API	807
<i>Existing runspace and isolated execution</i>	807
<i>Creating runspaces</i>	810
<i>Using runspaces for concurrency</i>	811
20.3 Runspace pools	813
20.4 Out-of-process runspaces	817
20.5 Remote runspaces	818
<i>Sessions and runspaces</i>	818
<i>Creating remote runspaces</i>	818
20.6 Managing runspaces	820
20.7 Summary	821
<i>appendix PowerShell 6.0 for Windows, Linux, and macOS</i>	823
<i>index</i>	842

preface

The second edition of this book was based on PowerShell v2. Since then we've seen a number of PowerShell releases—the current one is v5.1 with v6 in beta as we write. PowerShell use has grown astronomically to the extent that the PowerShell community is large enough to support independent conferences in North America, Europe, and Asia. User groups are available in all parts of the world.

PowerShell v2 was a big release bringing modules, remoting, and jobs. Subsequent releases have been as big in terms of their impact—PowerShell v3 brought PowerShell workflows and the CIM cmdlets, PowerShell v4 brought Desired State Configuration, and PowerShell v5 brought the deployment changes in PowerShell. These are only the headline items—there are many other small but important changes that expand and improve PowerShell 5. All of these changes are in a new edition of the book.

One big difference to the previous editions is that this book is now a single volume. Before we bring you the experience and knowledge of creating and deploying PowerShell solutions with extensive practical experience using PowerShell to solve real-world problems. Even with two authors creating the third edition has been a somewhat task. We had to drastically prune the material in the second edition to make room for the new material we had to cover. At one point we even decided to drop some of the content. We settled on a single volume and even though we've had to put some topics as being out of scope we've covered all of the new PowerShell.

So why write the book? The answer is the same now as it was then—we wanted the PowerShell community to have a way to see "inside the box" and have a