

Index

SYMBOLS

\$Host **built-in variable**, 198
 // (**double forward slash**), 120
 :: (**double colon**), 120–121
 <Autosize> **node**, 248–249
 <Configuration> **node**, 240
 <ViewDefinitions> **node**, 240
 <ViewSelectedBy> **node**, 241–242
 <Wrap> **node**, 248
 \ (**backward slash**), 119, 144, 153–154
 : (**colon**), 120
 / (**forward slash**), 119, 144, 153–154
 \\ (**double backward slash**), 120

A

ACLs (access control lists),
 ISecurityDescriptorCmdletProvider, 162
activity parameter names, 264–266
adapted members, 37, 54
 Add() **method**, **pipelines**, 191–192
 add-content **cmdlet**, 133, 160
 add-member **cmdlet**, 54
 AddPSSnapIn() **method**, 177
 Add-pssnapin **cmdlet**
 loading custom snap-in, 26
 loading standard snap-in, 19–20
 saving snap-in configuration vs. using, 22
 AddScript() **method**, **pipelines**, 191–192
administration design principles, 2
ADSI, PowerShell and, 2
 Alias **provider**, 125–126
 AliasAttribute **metadata**, 273
 AllowEmptyCollection **metadata**, 277
 AllowEmptyString **metadata**, 276
 AllowNullAttribute **metadata**, 276
APIs, and provider errors, 122
 appendPath **parameter**, update-formatdata **cmdlet**, 239
architecture
 GUI integration with command line, 193–194
 host application, 9–10
 Windows PowerShell Engine, 10
argument validation attributes, 273–276
 ArgumentTransformationAttribute **metadata**, 279–280
assembly, creating RunspaceConfiguration, 177–178
attributes
 adding to dynamic parameters at runtime, 280
 custom parameter validation, 79–80
 metadata. *See* metadata
 -autosize **parameter**, format-wide **cmdlet**, 235–236
 <Autosize> **node**, **format configuration**, 248–249

B

backward slash, double (\\), **provider-direct paths**, 120
backward slash (\), **provider path separator**, 119, 144, 153–154
base members, defined, 37
base provider types
 CmdletProvider, 129
 ContainerCmdletProvider, 131–132
 ItemCmdletProvider, 129–130
 NavigationCmdletProvider, 132
 overview of, 128–129
 BaseObject **property**, PSObject, 33–34
 BeginProcessing() **method**, 67, 87–91
binding parameters, 66
BufferCells, 230–231
built-in providers
 Alias, 125–126
 Certificate, 128
 Environment, 126
 FileSystem, 126
 Function, 126–127
 overview of, 125
 Registry, 127–128
 Variable, 128
business logic, GUI integration with command line, 194

C

C# language, code examples in this book, 13
capabilities, provider
 design guidelines, 162
 ProviderInfo object with, 133
 providers working with, 122–123
captions, as prompts, 225
 Certificate **provider**, 128
class inheritance, formatting, 250–252
 clear-content **cmdlet**, 133, 159
 ClearItem() **method**, ItemCmdletProvider, 147
 clear-item **cmdlet**, 130
 clear-itemproperty **cmdlet**, 134, 156, 297–298
 cmd.exe, **PowerShell cmdlets vs.**, 63
 CmdletAttribute **metadata**, 271–272
 CmdletConfigurationEntry **class**, 180
 CmdletProvider **class**, 134–162
 ContainerCmdletProvider from, 147–152
 defined, 129
 description of, 283–287
 design tips, 163
 DriveCmdletProvider from, 129, 139–141
 handling provider errors, 121–122
 IContentCmdletProvider from, 159–162
 >IDynamicPropertyCmdletProvider from, 158–159
 IPropertyCmdletProvider from, 156–158
 ISecurityDescriptorCmdletProvider from, 162

CmdletProvider class (continued)

CmdletProvider class (continued)

- ItemCmdletProvider from, 141–147
 - methods and providers on, 136–138
 - NavigationCmdletProvider from, 153–156
 - overview of, 134–136
 - cmdlets**, 3–8
 - accessing host instance, 198
 - adding to RunspaceConfiguration, 180
 - as API layer for GUI applications, 193–195
 - COM and WMI support for, 8
 - ContainerCmdletProvider, 131–132
 - get-command, 5–6
 - get-help, 4–5
 - get-member, 6–7
 - IContentCmdletProvider, 133
 - IDynamicPropertyCmdletProvider, 134
 - interacting with host with built-in, 199–204
 - IPropertyCmdletProvider, 134
 - ISecurityDescriptorCmdletProvider, 134
 - ItemCmdletProvider, 129–130
 - NavigationCmdletProvider, 132
 - overview of, 3–4
 - parameter naming guidelines, 263–270
 - provider interaction, 303–305
 - providers vs., 118–119
 - support for existing native OS commands, 7–8
 - verb naming guidelines, 257–261
 - verb-noun syntax of, 2
 - in Windows PowerShell Engine, 10
 - cmdlets, developing**
 - best practices, 114–116
 - command discovery, 65–66
 - command invocation, 67
 - command-line parsing, 65
 - documenting cmdlet help, 106–114
 - generating pipeline output, 91–92
 - getting started, 63–65
 - parameter binding, 66
 - processing pipeline input, 84–91
 - reporting errors, 92–98
 - supporting ShouldProcess() method, 98–100
 - using parameters. *See* parameters
 - working with PowerShell path, 101–106
 - colon (:), drive-qualified paths**, 120
 - colon, double (::), provider paths**, 120–121
 - colors, customizing text**, 253–255
 - COM, PowerShell and**, 2, 8
 - command arguments**
 - binding to command parameters with, 66
 - command-line parsing using, 65
 - parameter binding of positional parameters using, 69
 - command discovery**, 65–66, 190
 - command invocation**, 67
 - command name**, 65, 69
 - command parameters**, 65–66, 191
 - CommandInvocationIntrinsics, CmdletProvider, 138
 - commands**
 - cmdlets. *See* cmdlets
 - constructing pipelines programmatically, 189–193
 - executing existing native OS, 7–8
 - executing PowerShell Engine API, 166–169
 - Commands collection, 175–176
 - communication verbs, cmdlet names**, 260
 - compatibility, of PowerShell**, 2
 - complete cell type**, 230–231
 - <Configuration> **XML node, format**, 240
 - confirm **parameter**, ShouldProcess(), 100–101
 - ConfirmImpact **property**, ShouldProcess(), 99
 - console file (.pscl), creating RunspaceConfiguration**, 177
 - constructors**
 - executing custom converters, 58
 - RunspaceInvoke class, 166–167
 - ContainerCmdletProvider **class**
 - Alias provider deriving from, 126–127
 - container-specific cmdlets deriving from, 304
 - defined, 288
 - description of, 290–294
 - Environment provider deriving from, 126
 - Function provider deriving from, 126–127
 - implementing providers from, 135
 - NavigationCmdletProvider deriving from, 132, 153
 - overview of, 131–132, 147–152
 - Variable provider deriving from, 128
 - container-specific cmdlets**, 304
 - content-related cmdlets**, 305
 - Continue value**
 - DebugPreference variable, 199
 - Write-Progress cmdlet, 203
 - Write-Verbose cmdlet, 200–201
 - Write-Warning cmdlet, 202
 - ConvertThroughString, **custom PSTypeConverter**, 59–60
 - Copy() **method, pipelines**, 175
 - copy-item **cmdlet**, 131, 149–151
 - copy-itemproperty **cmdlet**, 134, 158, 298–301
 - CreateNestedPipeline() **method**, 175, 212
 - CreatePipeline() **method**, 168–169, 189–193
 - CreateRunspace() **factory method**, 168–169, 198
 - credentials**
 - capabilities, 122
 - CredentialAttribute metadata, 277–278
 - getting from user, 226–227
 - csc.exe, **writing snap-ins**, 16
 - CurrentCulture **property**, PSHost, 210
 - CurrentUICulture **property**, PSHost, 210
 - Custom Control**, 246–248
 - custom view**, 235, 246–248
 - CustomPSSnapin **class**, 13, 23–27
- ## D
- data verbs, cmdlet naming**, 259
 - DataReady **property**, PipelineReader, 184–185
 - date parameter names, cmdlets**, 266
 - DebugPreference **variable**
 - in cmdlet and host interaction, 204
 - Write-Debug cmdlet, 199–200
 - WriteDebugLine method, 222–223
 - default parameter sets**, 72–73
 - description**, ProviderInfo, 133
 - deserialized objects, formatting**, 250
 - diagnostic verbs, cmdlet names**, 260
 - disallow attributes**, 276–278
 - distance algorithms**, 54–55
 - DLLs, registering snap-ins**, 19
 - double backward slash (\\), provider-direct paths**, 120
 - double colon (::), provider paths**, 120–121
 - double forward slash (//), provider-direct paths**, 120
 - DriveCmdletProvider **class**
 - CmdletProvider class vs., 134–135
 - description of, 287–288

drive-specific cmdlets deriving from, 303
 ItemCmdletProvider deriving from, 130, 141–142
 overview of, 129, 139–141

drive-qualified paths, 120

drives

core cmdlets for, 303
 ProviderInfo object, 133
 providers and, 121

dynamic parameters, 133, 280

dynamic property manipulation cmdlets, 305

E

EndProcessing() **method**, 67

EnterNestedPrompt() **method**, PSHost, 211–212

Environment **provider**, 126

error pipes

defined, 172
 retrieving non-terminating errors from, 173
 using RunspaceInvoke with, 167–168

ErrorDetails **class**, 95–96

ErrorRecord objects

handling provider errors, 121–122
 non-terminating errors returned as, 173
 overview of, 174
 reporting cmdlet errors, 93–95

errors

asynchronous pipeline, 182–185
 design tips, 163
 DriveCmdletProvider, 140
 initialization, 56
 provider developers handling, 121–122
 runtime, 55–56

errors, reporting cmdlet

creating ErrorDetails, 95–96
 creating ErrorRecord, 93–95
 non-terminating and terminating errors, 97–98
 overview of, 92–93
 PowerShell path, 101–106

ETS (Extended Type System), 29. *See also* PSObject exceptions

ErrorRecord, 93
 from ETS during runtime, 55–56

Exclude operation, capabilities, 122

exit **command**, 214

ExitNestedPrompt() **method**, PSHost, 212–214

expansion, path, 121

explicit cast operator, 58

Export-console **cmdlet**, 22, 23

extended members

defining methods and properties with, 31
 lookup algorithms and, 54
 overview of, 37

Extended Type System (ETS), 29. *See also* PSObject

F

F & O. *See* Formatting & Output

FileSystem **provider**, 126

filters

capabilities and, 122
 CmdletProvider, 138

Format and Output (F & O). *See* Formatting & Output

format parameter names, cmdlets, 266–267

format strings, 249

FormatConfigurationEntry **class**, 180

format-custom **cmdlet**, 235–237

format-list **cmdlet**, 234–237

format-table **cmdlet**, 224, 234, 236–237

Formatting & Output

class inheritance, 250–252

colors, 253–255

Custom Control, 246–248

format configuration file anatomy, 240–243

format configuration file example, 237–238

format strings, 249

formatting deserialized objects, 250

formatting without *.format.ps1xml, 236–237

ListControl, 244–246

loading format file(s), 238–240

miscellaneous configuration entries, 248–249

overview of, 224–225

selection sets, 253

TableControl, 243–244

view types, 233–236

WideControl, 246

formatting files, RunspaceConfiguration, 180

format-wide **cmdlet**, **wide view**, 235–236

forward slash, double (//), **provider-direct paths**, 120

forward slash (/), **provider path separator**, 119, 144, 153–154

Function **provider**, 126–127

functions, RunspaceConfiguration, 181

G

get-acl **cmdlet**, 134, 162

GetBufferContents() **method**, BufferCells, 230–231

get-childitem **cmdlet**, 131, 148, 290

get-command **cmdlet**, 5–6, 19–20

get-content **cmdlet**, 133, 159–161

get-help **cmdlet**

about providers, 117
 documenting cmdlet help, 106–114
 overview of, 4–5

GetItem() **method**, ItemCmdletProvider, 145–146

get-item **cmdlet**, 130

get-itemproperty **cmdlet**, 134, 156–158, 297–298

get-location **cmdlet**, 132, 147

get-member **cmdlet**, 6–8

get-pssnapin **cmdlet**, 10–11, 19

get-pssnapin -registered **command**, 22–23

GetResolvedProviderPathFromPSPath() **method**, **file path resolution**, 103–106

GetVariable() **method**, SessionStateProxy, 178

GroupBy, **format configuration files**, 242–243

GUI applications, cmdlets as API layer for, 193–195

GUID, creating for each host instance, 208

H

Hello World **provider**, 123–125, 135

help commands, 4–5, 106–114

host APIs, 9–10, 115

\$Host **built-in variable**, 198

Host **property**, PSCmdlet **class**, 198

hosts, 197–231

custom, 194–195

interaction with built-in cmdlets, 199–204

interaction with cmdlets, 204–207

hosts (continued)

hosts (continued)

interaction with PowerShell Engine, 197–199
 using `PSHost` class. See `PSHost` class
 using `PSHostRawUserInterface` class, 227–231
 using `PSHostUserInterface` class, 221–227

I

`IContentbreakCmdletProvider` **interface**, 305
`IContentCmdletProvider` **interface**, 132–133, 159–162, 295–296

`IContentReader` **interface**
 cmdlets supported by, 305
 description of, 296–297
 overview of, 159

`IContentWriter` **interface**
 cmdlets supported by, 305
 description of, 297
 overview of, 159–160

`IDynamicCmdletProvider` **interface**, 305

`IDynamicParameters` **interface**, 280

`IDynamicPropertyCmdletProvider` **interface**
 description of, 298–301
 overview of, 134
 working with, 158–159

`IEnumerable` **interface**, 172–173

`ImmediateBaseObject` **property**, `PSObject`, 33–34

implicit cast operator, custom converters, 58

Include operation, capabilities, 123

inheritance, class, 250–252

initialization errors, 56

`InitializeDefaultDrives()` **method**,
`DriveCmdletProvider`, 139

input

closing input pipe, 182
 passing to pipeline, 172–173
 pipeline parameter binding for, 87–91
 processing pipeline, 84–87
 using `RunspaceInvoke` with, 167–168

Inquire value

`DebugPreference` variable, 199
`Write-Progress` cmdlet, 203
`Write-Verbose` cmdlet, 200–201
`Write-Warning` cmdlet, 202

installation, PowerShell, 3

`installutil.exe`
 registering custom snap-ins, 26
 registering snap-ins, 17–19
 uninstalling custom snap-in, 26
 uninstalling standard snap-ins, 20–21
 writing snap-ins, 15–16

`InstanceId` **property**, `PSHost` **class**, 208–209

interfaces, optional provider, 132–134

intrinsic members, `PSObject`, 55

invocation, command, 67

`InvocationInfo`, `ErrorRecord`, 93, 95

`Invoke()` **method**

executing command in PowerShell Engine, 169
 passing input to pipeline, 172–173
 retrieving output from pipeline, 170
 using `RunspaceInvoke` with, 167

`InvokeAsync()` **method**, `Pipeline` **class**,
 181–182

`invoke-item` **cmdlet**, 130, 146

`ipconfig.exe` **command**, 7–8

`IPropertyCmdletProvider` **interface**

definition for, 297–298
`IDynamicPropertyCmdletProvider` deriving from, 134,
 158

Item-property cmdlets for, 134, 297, 304–305
 working with, 156–158

`ISecurityDescriptorCmdletProvider` **interface**

cmdlets supported by, 134, 305

defined, 134

overview of, 162

`IsGettable` **property**, `PSPropertyInfo`, 39

`IsItemContainer()` **method**, `NavigationCmdletProvider`,
 154

`IsSettable` **property**, `PSPropertyInfo`, 39

`IsValidPath()` **method**, `ItemCmdletProvider`, 143

`ItemCmdletProvider` **class**

cmdlets supported by, 303–304

`ContainerCmdletProvider` deriving from, 131–132, 147

description of, 288–290

implementing providers from, 135

overview of, 129–130

working with, 141–147

`ItemExists()` **method**

`IContentCmdletProvider`, 161

`IPropertyCmdletProvider`, 158

item-specific cmdlets, 303–304

J

`join-path` **cmdlet**, 132, 154

L

leading cell types, 230–231

lifecycle verbs, naming cmdlets, 261

list view, 234–235, 244–246

ListControl, 244–246

ListEntries, 245

`LocalRunspace` **class**, 168–169, 187–188

lookup algorithms, 54

M

`MakePath()` **method**, `NavigationCmdletProvider`, 154

`make-shell.exe`, 178

mandatory parameters, 67–68

member sets, `PSObject`

intrinsic, 55

overview of, 51

`PSMemberSet`, 52–53

`PSPropertySet`, 51–52

member types, `PSObject`, 38–48

methods, 46–50

overview of, 37–38

properties, overview, 38–39

`PSAliasProperty`, 45–46

`PSCodeProperty`, 43–45

`PSNoteProperty`, 40–41

`PSProperty`, 39–40

`PSScriptProperty`, 41–43

sets, 51–53

members, `PSObject`

base, adapted and extended, 37

distance algorithms, 54–55

intrinsic, 55

Pipeline Execution Thread

lookup algorithms, 54
 overview of, 34–35
 PSMemberInfoCollection and, 35–36
 ReadOnlyPSMemberInfoCollection and, 36–37
 well-known, 62
 MergeMyResults **method, pipelines**, 191
 MergeUnclaimedPreviousResults **property, pipelines**, 190–191
message strings, 96
messages, as prompts, 225
metadata, 271–281
 adding attributes to dynamic parameters at runtime, 280
 AliasAttribute, 273
 allow and disallow attributes, 276–278
 argument validation attributes, 273–276
 ArgumentTransformationAttribute, 279–280
 CmdletAttribute, 271–272
 overview of, 271
 ParameterAttribute, 272–273
 ValidateArgumentsAttribute, 278
 ValidateEnumeratedArgumentsAttribute, 279
 ValidateScriptAttribute, 281
 MoveItem() **method**, NavigationCmdletProvider, 155
 move-item **cmdlet**
 ContainerCmdletProvider, 151
 NavigationCmdletProvider, 132, 155
 move-itemproperty **cmdlet**, 134, 158, 298–301

N

name node, format configuration files, 241
 Name **property**
 PSHost class, 209–210
 writing snap-ins, 15–16
named parameter binding, 76
naming collisions, lookup algorithms, 54
naming conventions. See also parameter naming guidelines, cmdlets
 cmdlet best practices, 114–115
 custom snap-ins, 25
 DriveCmdletProvider, 139
 Get-Help cmdlet, 106
 writing snap-ins, 15–16
 NavigationCmdletProvider **class**
 Certificate provider deriving from, 128
 defined, 132
 description of, 294–295
 FileSystem provider deriving from, 126
 overview of, 153–156
 Registry provider derived from, 127
nested containers, NavigationCmdletProvider, 153
nested pipelines
 exiting, 212–213
 invoking, 211–212
 overview of, 174–175
.NET Framework, 2, 16
 NewDriveDynamicParameters() **method**,
 DriveCmdletProvider, 139
 new-item **cmdlet**, 131, 151, 290
 new-itemproperty **cmdlet**, 134, 158, 299–301
 new-psdrive **cmdlet**, 129
non-terminating errors
 defined, 93
 reporting, 97–98
 retrieving from error pipes, 173

NotifyBeginApplication() **method**, PSHost **class**, 214
 NotifyEndApplication() **method**, PSHost **class**, 214
nouns, cmdlet naming best practices, 114

O

object inheritance, 239
object-based language, PowerShell's, 2
 ObjectSecurity **class**,
 ISecurityDescriptorCmdletProvider, 162
 Open() **method, executing command**, 168–169
 OpenAsync() **method**, Runspace **class**, 181, 187–188
 out-default **cmdlet**, 254–255
 out-host **cmdlet**, 203–204
output
 collecting synchronously invoked pipeline, 173
 generating pipeline, 91–92
 reading asynchronous pipeline, 182–185
 retrieving pipeline, 170–172
 using RunspaceInvoke, 167–168

P

parameter binding
 overview of, 66
 and parameter sets, 75–78
 pipeline, 87–91
 for positional parameters, 69–70
 processing pipeline input with, 84–91
parameter naming guidelines, cmdlets, 263–270
 activity, 264–266
 date/time, 266
 format, 266–267
 property, 267–268
 quantity, 268
 resource, 268–269
 security, 269–270
 ubiquitous, 263–264
parameter sets, 71–78
 default, 72–73
 defining parameters belonging to multiple, 73–75
 overview of, 70–71
 parameter binding related to, 75–76
 pipeline parameter binding for, 88–91
 ParameterAttribute **metadata**, 272–273
parameters, 67–84
 best practices for cmdlet naming, 115
 mandatory, 67–68
 overview of, 67
 parameter sets, 71–78
 positional, 68–71
 transformation, 80–84
 validation, 78–80
parent-child relationships, ContainerCmdletProvider,
 147–148
 Parse **method, custom converters**, 58
parsing, command-line, 65
passwords, for credentials, 225
paths
 design guidelines, 162
 NavigationCmdletProvider and, 153
 provider, 119–121
 supported by ItemCmdletProvider, 129–130
 working with, 101–106
Pipeline Execution Thread, 220–221

PipelineReader class

PipelineReader **class**, 182–185

pipelines

- collecting output for, 173
- constructing programmatically, 189–193
- copying between runspaces, 175–176
- executing commands, 168–169
- generating output, 91–92
- Input, Output and Error properties, 172
- nested, 174–175
- parameter binding, 87–91
- passing input to, 172–173
- Pipeline class, 165
- processing input, 84–87
- retrieving output from, 170–172
- reusing, 175
- in Windows PowerShell Engine, 10, 165–166, 198

pipelines, running asynchronously, 181–187

- calling InvokeAsync, 181–182
- closing input pipe, 182
- monitoring StateChanged event, 185–186
- reading output and error, 182–185
- reading terminating errors, 186–187
- stopping, 187

PipelineStateInfo.Reason, 186–187

pop-location **cmdlet**, 131

positional parameters

- binding, 69–70, 76–78
- overview of, 68–71
- remaining-argument parameter, 70–71

PowerShell, introduction, 1–11

- cmdlets, 3–8
- design principles, 1–3
- extending. *See* snap-ins
- high-level architecture of, 9–11

PowerShell Engine API. *See also* hosts

- asynchronous runspace operations, 187–188
- cmdlets as API layer for GUI applications, 193–195
- configuring runspace, 176–181
- constructing pipelines programmatically, 189–193
- copying pipeline between runspaces, 175–176
- ErrorRecord type, 174
- executing command line, 166–168
- getting started, 166
- nested pipelines, 174–175
- output pipe in synchronous execution, 173
- passing input to pipeline, 172–173
- retrieving non-terminating errors from error pipe, 173
- retrieving pipeline output, 170–172
- reusing pipelines, 175
- running pipeline asynchronously, 181–187
- runspaces and pipelines, 165–166

PowerShell.exe, 9–10

precedence, TypeNames, 53–54

prependPath **parameter**, update-formatdata **cmdlet**, 239

PrivateData **property**, PSHost **class**, 210

ProcessRecord() **method**, command invocation, 67

profiles, saving snap-in configuration, 23

ProgressPreference **variable**, Write-Progress **cmdlet**, 203, 205, 224

ProgressRecord **object**, 223–224

Prompt()method, PSHostUserInterface **class**, 224–226

PromptForCredential() **method**, PSHostUserInterface **class**, 226–227

properties

- AliasAttribute metadata, 273

- CmdletProvider, 136–138

- defining with extended members, 31

- examining with get-member, 6–7

- objects wrapped by PSObject, 31

- ParameterAttribute metadata, 272–273

- ProviderInfo object, 133–134

- PSHost class. *See* PSHost class

- PSHostRawUserInterface class, 229

- PSObject, 38–46

property parameter names, cmdlets, 267–268

property-specific cmdlets, 304–305

ProviderCapabilities **enumerated type**

- design guidelines, 162

- overview of, 122–123

- ProviderInfo object, 133

ProviderConfigurationEntry **class**, 180

provider-direct paths, 120

ProviderInfo **object**, Hello World provider, 132–133

provider-internal paths, 120–121

provider-qualified paths, 120

providers, 117–164

- adding to RunspaceConfiguration, 180

- base provider types, 128–132

- built-in, 125–128

- capabilities, 122–123

- CmdletProvider. *See* CmdletProvider class

- cmdlets vs., 118–119

- core cmdlets for interaction with, 303–305

- design guidelines and tips, 162–163

- drives, 121

- error handling, 121–122

- Hello World provider, 123–125

- optional interfaces for, 132–134

- overview of, 117–118

- paths, 119–121

- reasons for implementing, 118

providers, base classes and interfaces, 283–301

- CmdletProvider, 283–284

- ContainerCmdletProvider, 290–294

- DriveCmdletProvider, 287–288

- IContentCmdletProvider, 295–296

- ItemCmdletProvider, 288–290

- NavigationCmdletProvider, 290

PSAdapted MemberSet, 55

PSAliasProperty, 45–46

PSBase MemberSet, 55

.psci (console file), creating RunspaceConfiguration from, 177

PSCodeProperty, 43–45

PSCustomObject, 31

PSDriveInfo **object**

- creating with DriveCmdletProvider, 129, 139–141

- design tips, 163

- NavigationCmdletProvider, 153

PSExtended MemberSet, 55

PSHost **class**, 207–221

- CurrentCulture property, 210

- CurrentUICulture property, 210

- defined, 198

- EnterNestedPrompt()method, 211–212

- ExitNestedPrompt()method, 212–214

- InstanceID property, 208–209

- Name property, 209–210

- NotifyBeginApplication()method, 214

- NotifyEndApplication()method, 214

RunspaceStateInfo property, runspace StateChanged event

overview of, 207–208
 PrivateData property, 210
 SetShouldExit() method, 214–221
 Version property, 210
 PSHostRawUserInterface **class**, 198–199, 227–231
 PSHostUserInterface **class**
 defined, 198–199
 overview of, 221–222
 Prompt() method, 224–226
 PromptForCredential() method, 226–227
 read methods, 227
 write methods, 224
 WriteDebugLine() method, 222–223
 WriteErrorLine() method, 223–224
 WriteProgress() method, 223–224
 WriteVerboseLine() method, 223
 WriteWarningLine() method, 223
 PSMemberInfo **objects**, 34–37
 PSMemberInfoCollection, 35–36
 PSMemberSet, 52–53
 PSNoteProperty, 40–41
 PObject
 constructing, 30–33
 distance algorithm, 54–55
 errors and exceptions, 55–56
 lookup algorithm, 54
 members, 34–37
 methods, 46–51
 overview of, 29–30, 37–38
 sets, 51–53
 supporting intrinsic members and MemberSets, 55
 ToString mechanism, 60
 type configuration, 60–62
 type conversion, 57–60
 TypeName, 53–54
 using ImmediateBaseObject and BaseObject, 33–34
 using objects returned from pipelines, 170
 using with IPropertyCmdletProvider, 158
 PObject, **properties**, 38–46
 defined, 34
 overview of, 38–39
 PSAliasProperty, 45–46
 PSCodeProperty, 43–45
 PSNoteProperty, 40–41
 PSParameterizedProperty, 50–51
 PSProperty, 39–40
 PSScriptProperty, 41–43
 PObject(**object**), 31
 PSProperty, 39–40
 PSPropertyInfo, 39
 PSPropertySet, 51–52
 PSScriptProperty, 41–43
 PSSnapin **class**
 defined, 13
 writing custom snap-ins, 23–27
 writing standard snap-ins. *See* snap-ins, standard
 PTypeConverter **class**, 59–60
 push-location **cmdlet**, 131–132

Q

quantity parameter names, **cmdlets**, 268

R

Read() **methods, output pipes**, 173
read methods, PSHostUserInterface **class**, 227
 Read-Host **built-in cmdlet**, 204
 ReadKey() **method**, PSHostRawUserInterface **class**, 229
 ReadLine() **host API**, 204, 227
 ReadLineAsSecureString() **host API**, 204, 227
 ReadOnlyPSMemberInfoCollection, 36–37
 ReadToEnd() **method, non-terminating errors**, 173
registry
 list of registered snap-ins in, 19
 registering custom snap-ins, 26
 registering snap-in without implementing snap-in class, 22–23
 registering snap-ins, 17–19
 unregistering snap-ins, 20–21
 Registry **provider**, 127–128
remaining-argument parameter, 70–71
 RemoveDrive() **method**, DriveCmdletProvider, 139–140
 remove-item **cmdlet**, 131, 290
 remove-itemproperty **cmdlet**, 134, 158, 299–301
 remove-psdrive **cmdlet**, 129
 RemovePSSnapIn() **method**, 177
 remove-ssnapin **cmdlet**
 Hello World provider, 133
 removing custom snap-in, 26
 removing snap-in, 20
 rename-item **cmdlet**, 131
 rename-item **cmdlet**, 290
 rename-itemproperty **cmdlet**, 134, 299–301
 Reset() **method, configuration collection**, 180
 resolve-path **cmdlet**, 132
resource parameter names, cmdlets, 268–269
resource strings, ErrorDetails **object**, 96
 RunInstaller **attribute, writing snap-ins**, 14–16
 Runspace **class. See also runspaces**
 RunspaceConfiguration **object**
 adding and removing snap-ins, 177
 creating from assembly, 177–178
 creating from console file, 177
 with custom configuration, 176
 executing command in PowerShell Engine, 166–167
 fine-tuning, 179–181
 hosting applications, 197–199
 loading format file(s), 240
 using SessionStateProxy to set/retrieve variables, 178
 RunspaceFactory **class, creating runspace**, 168–169
 RunspaceInvoke **class**, 166–168
runspaces
 adding and removing snap-ins, 177
 asynchronous operations, 187–188
 copying pipeline between, 175–176
 creating from assembly, 177–178
 creating from console file, 177
 creating with custom configuration, 176
 executing command in PowerShell Engine, 168–169
 fine-tuning, 179–181
 for hosting applications, 197–198
 PowerShell Engine API, 165–166
 Runspace class, 165
 using SessionStateProxy to set/retrieve variables, 178
 RunspaceStateInfo **property, runspace StateChanged event**, 188

runtime

runtime

- adding attributes to dynamic parameters at, 280
- command-line parsing at, 65
- errors, 55–56

S

ScriptConfigurationEntry, 181

scripts

- accessing host instance with, 198
- accessing PObject with, 62
- design principles, 2–3

security

- descriptor-related cmdlets for, 305
- parameter names for cmdlets, 269–270
- PromptForCredential() methods for, 226–227
- verbs, 261

selection sets, formatting, 253

SelectSingleNode() method, ItemCmdletProvider, 145

SessionState object

- CmdletProvider, 137
- design guidelines, 133
- ExitNestedPrompt() method, 213
- runspaces, 165

SessionStateProxy property, runspaces, 178

set-acl cmdlet, 134, 162

set-content cmdlet, 133, 159–161

SetItem() method, ItemCmdletProvider, 146

set-item cmdlet, 130

set-itemproperty cmdlet, 134, 156, 297–298

set-location cmdlet

- ContainerCmdletProvider, 131, 147
- NavigationCmdletProvider, 154

SetShouldExit() method, PSHost class, 214–221

SetVariable() method, SessionStateProxy, 178

shells, object-based vs. text-based, 2

ShouldContinue() method

- best practices for cmdlets, 115
- CmdletProvider used with, 137, 163
- working with, 101

ShouldProcess() method

- best practices for cmdlets, 115
- CmdletProvider used with, 136–137
- NavigationCmdletProvider used with, 155–156
- overview of, 98–100
- provider capabilities using, 122–123
- working with, 100–101

SilentlyContinue value

- DebugPreference variable with, 199
- Write-Progress cmdlet with, 203
- Write-Verbose cmdlet with, 200–201
- Write-Warning cmdlet with, 202

SnapinName key, snap-ins, 17–19

snap-ins

- configuring runspace by adding and removing, 177
- defined, 10
- loading format file(s) with, 240
- overview of, 14
- providers within, 117
- types of, 13
- viewing list of, 10–11
- writing custom, 23–27

snap-ins, standard, 14–23

- getting list of, 19
- loading to running shell, 19–20

overview of, 14

- registering, 17–19
- registering without implementing snap-in class, 21–22
- removing from running shell, 20
- saving configuration, 22
- starting with saved configuration, 22–23
- unregistering, 20–21
- using profile to save configuration, 23
- writing, 14–16

sort-object cmdlet, 172–173

Standard PS Language conversions, 57–58

Start() method, adding providers, 135

StartDynamicParameters() method, 136

StateChanged event

- handling runspace, 188
- monitoring pipeline, 185–186
- reading terminating errors, 186–187

Stop() method, pipeline, 187

Stop value

- DebugPreference variable, 199
- Write-Progress cmdlet, 203
- Write-Verbose cmdlet, 200–201
- Write-Warning cmdlet, 202

StopAsync() method, pipeline, 187

SupportsShouldProcess property, ShouldProcess() method, 99

System.Management.Automation assembly, 15, 166

T

table view, 234, 243–244

TableControl, 243–244

TableHeaders, TableControl, 243–244

TableRowEntries, TableControl, 244

Target object, ErrorRecord, 93

terminating errors

- defined, 93
- handling from commands, 171–172
- reading via PipelineStateInfo.Reason, 186–187
- reporting, 97–98

test-path cmdlet, 132

this pointer, PSMemberSet, 52–53

ThrowTerminatingError() method, 97, 186

ThrowTerminatingError(ErrorRecord)

CmdletProvider, 136

DriveCmdletProvider, 140

handling provider errors, 122

time parameter names, cmdlets, 266

ToString(), PObject, 60, 167

trailing cell type, 230–231

transformation, parameter, 80–84

Trap statement, runtime errors, 55–56

type configuration (TypeData), 60–62

type conversion, 57–60

type files, adding to RunspaceConfiguration, 181

TypeConfigurationEntry class, 181

TypeConverter, custom converters, 58

TypeNames

- basing extended members on, 37
- determining type of PObject with, 55
- overview of, 53–54

U

-u parameter, snap-ins, 20–21, 26

ubiquitous parameter names, cmdlets, 263–264

XmlProviderUtils.NormalizePath()

UICulture, **host**, 210
unbound argument list, 69–70
unbound positional parameter list, 69–70
 Update() **method, configuration collection**, 180
 update-formatdata **cmdlet**, 239
user feedback APIs, developing cmdlets, 114–115
usernames, prompting for credentials, 225

V

ValidateArgumentsAttribute **metadata**, 278
 ValidateCountAttribute **metadata**, 275
 ValidateEnumeratedArgumentsAttribute **metadata**, 279
 ValidateLengthAttribute **metadata**, 274–275
 ValidateNotNull **attribute metadata**, 277
 ValidateNotNullOrEmpty **attribute metadata**, 277
 ValidatePatternAttribute **metadata**, 274
 ValidateRangeAttribute **metadata**, 275–276
 ValidateScriptAttribute **metadata**, 281
 ValidateSetAttribute **metadata**, 274
validation parameters, 78–80
values, Write-Debug cmdlet, 199
Variable provider, 128
variables, built-in cmdlets interacting with host, 199–204
Vendor property, snap-ins, 16
verb naming guidelines, cmdlets
 common verbs, 257–258
 communication verbs, 260
 data verbs, 259
 diagnostic verbs, 260
 lifecycle verbs, 261
 security verbs, 261
verb-noun syntax, of cmdlets
 best practices for, 114
 defined, 2
 providers vs. cmdlets, 118–119
VerbosePreference variable
 in cmdlet and host interaction, 204–205
 Write-Verbose cmdlet, 200–201
 WriteVerboseLine method, 223
VerbsCommon class, cmdlet verbs, 257–258
VerbsCommunications class, naming cmdlets, 260
VerbsData class, naming cmdlets, 259
VerbsDiagnostic class, naming cmdlets, 260
VerbsLifecycle class, naming cmdlets, 261
VerbsSecurity class, naming cmdlets, 261
Version property, PSHost class, 210
 <ViewDefinitions> **XML node, format configuration**, 240
views
 custom, 235
 format configuration files, 241
 list, 234–235
 table, 234
 types of, 233–234
 wide, 235–236
 <ViewSelectedBy> **node, format configuration**, 241–242

W

WaitHandle **property, PipelineReader class**, 183–184
 WarningPreference **variable, cmdlet and host interaction**, 205
well-known members, PSObject, 62
 -whatif **parameter, ShouldProcess () method**, 100

wide view, 235–236, 246
WideControl, 246
WideEntries, 246
wildcard characters, 4, 122–123
 WildcarePattern **class**, 122–123
Windows PowerShell Engine, 10
WMI, PowerShell and, 2, 8
 <Wrap> **node, format configuration**, 248
Write() method
 IContentCmdletProvider interface, 161–162
 input pipe, 172, 173
 PSHostUserInterface class, 224
 Write-Host and Out-Host cmdlets, 203–204
Write-Debug cmdlet, 199–200
WriteDebug() method, 115, 204–207
WriteDebugLine() method, PSHostUserInterface class, 222–223
WriteError(ErrorRecord)
 best practices for, 115
 DriveCmdletProvider and, 140
 handling provider errors, 122
 reporting non-terminating errors with, 97
 writing to error stream with, 207
WriteErrorLine() method, PSHostUserInterface class, 223–224
Write-Host built-in cmdlet, 203–204
WriteItemObject () method
 CmdletProvider, 136
 ItemCmdletProvider, 146
WriteItemProperty() method,
 IDynamicPropertyCmdletProvider, 158
WriteLine() method
 PSHostUserInterface class, 224
 Write-Host and Out-Host cmdlets, 203–204
WriteObject() method
 generating pipeline output, 91–92
 writing to output stream with, 207
Write-Progress built-in cmdlet, 203
WriteProgress() method
 best practices for, 115
 cmdlet/host interaction and, 205, 207
 CmdletProvider and, 163
 PSHostUserInterface class with, 223–224
WriteVerbose() method
 best practices for, 115
 cmdlet/host interaction and, 204–205, 207
 CmdletProvider and, 136
 ItemCmdletProvider and, 144
 write-verbose **cmdlet**, 200–201
WriteVerboseLine() method, PSHostUserInterface class, 223
WriteWarning() method
 best practices for, 115
 cmdlet/host interaction and, 205, 207
 CmdletProvider and, 136
 DriveCmdletProvider and, 140
 Write-Warning **cmdlet**, 202, 223
WriteWarningLine() method, PSHostUserInterface class, 223

X

XmlDriveInfo **class**, 140–141
 XmlProviderUtils.NormalizePath(), 144