

PowerShell®

Notes for Professionals

Chapter 3: Operators

An operator is a character that represents an action. It tells the compiler/interpreter to perform mathematical, relational or logical operation and produce final result. PowerShell interprets its categories accordingly like arithmetic operators perform operations primarily on numbers, bit strings and other data types. Along with the basic operators, PowerShell has a number of special and coding effort (eg. -like, -match, -replace, etc).

Section 3.1: Comparison Operators

PowerShell comparison operators are comprised of a leading dash (-) followed by a name (greater, lt, etc.).

Names can be preceded by special characters to modify the behavior of the operator:

- 1 # Case-Insensitive Explicit (-ci)
- 2 # Case-Sensitive Explicit (-cs)

Case-insensitive is the default if not specified. ("a" -eq "A") same as ("a" -eq "A").

Simple comparison operators:

```
2 -eq 2 # Equal to (-eq)
2 -ne 4 # Not equal to (-ne)
5 -gt 3 # Greater-than (-gt)
5 -ge 5 # Greater-than or equal to (-ge)
5 -lt 10 # Less-than (-lt)
5 -le 5 # Less-than or equal to (-le)
```

String comparison operators:

```
"mystring" -like "string" # Match using the wildcard
"mystring" -notlike "string" # Does not match using the wildcard
"mystring" -match "string" # Matches a string using regular expressions
"mystring" -notmatch "string" # Does not match a string using regular expressions
```

Collection comparison operators:

```
"abc" -contains "def" # Returns true when the value is in the array (left)
"abc" -notcontains "def" # Returns true when the value is not in the array (left)
"def" -in "abc" # Returns true when the value is in the array (right)
"def" -notin "abc" # Returns true when the value is not in the array (right)
```

Section 3.2: Arithmetic Operators

```
1 + 2 # Addition
1 - 2 # Subtraction
-1 # Get negative value
1 * 2 # Multiplication
1 / 2 # Division
1 % 2 # Modulus
```

Chapter 12: PowerShell Functions

A function is basically a named block of code. When you call the function name, the script block within that function runs. It is a list of PowerShell statements that has a name that you assign. When you run a function, you type the function name. It is a method of saving time when tackling repetitive tasks. PowerShell formats in three parts: the keyword 'function', followed by a Name, finally, the payload containing the script block, which is enclosed by curly/parenthesis style brackets.

Section 12.1: Basic Parameters

A function can be defined with parameters using the param block:

```
function Write-Greeting {
    param (
        [Parameter(Mandatory=$true)]
        [string]$name,
        [Parameter(Mandatory=$true)]
        [int]$age
    )
    "Hello $name, you are $age years old."
}
```

Or using the simple function syntax:

```
function Write-Greeting ($name, $age) {
    "Hello $name, you are $age years old."
}
```

Note: Casing parameters is not required in either type of parameter definition.

Simple function syntax (SFS) has very limited capabilities in comparison to the param block. Though you can define parameters to be exposed within the function, you cannot specify Parameter Attributes, utilize Parameter Validation, include [OutputBinding()] , with SFS (and this is a non-exhaustive list).

Functions can be invoked with ordered or named parameters.

The order of the parameters on the invocation is matched to the order of the declaration in the function header (default), or can be specified using the Position Parameter Attribute (as shown in the advanced function example above).

```
$greeting = Write-Greeting "Us" 80
```

Alternatively, this function can be invoked with named parameters

```
$greeting = Write-Greeting -name "Bob" -age 82
```

Section 12.2: Advanced Function

This is a copy of the advanced function snippet from the PowerShell ISE. Basically this is a template for many things you can use with advanced functions in PowerShell. Key points to note:

- get-help integration - the beginning of the function contains a comment block that is set up to be get-help cmdlet. The function block may be located at the end, if desired.
- cmdletbinding - function will behave like a cmdlet.

PowerShell® Notes for Professionals

Chapter 23: Sending Email

Parameter	Details
Attachments<String[]>	Path and file names of files to be attached to the message. Paths and filenames can be piped to Send-MailMessage.
Bcc<String[]>	Email addresses that receive a copy of an email message but does not appear as a recipient in the message. Enter names (optional) and the email address (required), such as Name someone@example.com or someone@example.com.
Body<String, BodyAsHtml>	Content of the email message.
Cc<String[]>	It indicates that the content is in HTML format.
Credential	Email addresses that receive a copy of an email message. Enter names (optional) and the email address (required), such as Name someone@example.com or someone@example.com.
DeliveryNotificationOption	Specifies a user account that has permission to send message from specified email address. The default is the current user. Enter name such as User or Domain\User, or enter a PSID object.
Encoding	Specifies the delivery notification options for the email message. Multiple values can be specified. Acceptable values: None, OnSuccess, OnFailure, Delay, Never.
From	Encoding for the body and subject. Acceptable values: ASCII, UTF8, UTF7, UTF32, Unicode, BigEndianUnicode, Default, OEM.
Port	Email address from which the mail is sent. Enter names (optional) and the email address (required), such as Name someone@example.com or someone@example.com.
Priority	Alternate port on the SMTP server. The default value is 25. Available from Windows PowerShell 5.0.
SmtpServer	Priority of the email message. Acceptable values: Normal, High, Low.
Subject	Name of the SMTP server that sends the email message. Default value is the value of the \$PSMtaServer variable.
To	Subject of the email message.
UseSsl	(required), such as Name someone@example.com or someone@example.com. Uses the Secure Sockets Layer (SSL) protocol to establish a connection to the remote computer to send mail.

A useful technique for Exchange Server administrators is to be able to send email messages via SMTP from PowerShell. Depending on the version of PowerShell installed on your computer or server, there are multiple ways to send emails via PowerShell. There is a native cmdlet option that is simple and easy to use. It uses the cmdlet:

Section 23.1: Send-MailMessage with predefined parameters

```
Send-MailMessage -To "fred@fred.com" -Subject "Email Subject" -Attachments @"C:\files\example1.txt", "C:\files\example2.txt" -Body "Example Body" -Cc "fred@fred.com" -Credential fred -DeliveryNotification "OnSuccess" -Encoding UTF8 -Port 33
```

PowerShell® Notes for Professionals

100+ pages
of professional hints and tricks

Contents

About	1
Chapter 1: Getting started with PowerShell	2
Section 1.1: Allow scripts stored on your machine to run un-signed	2
Section 1.2: Aliases & Similar Functions	2
Section 1.3: The Pipeline - Using Output from a PowerShell cmdlet	3
Section 1.4: Calling .Net Library Methods	4
Section 1.5: Installation or Setup	5
Section 1.6: Commenting	5
Section 1.7: Creating Objects	6
Chapter 2: Variables in PowerShell	7
Section 2.1: Simple variable	7
Section 2.2: Arrays	7
Section 2.3: List Assignment of Multiple Variables	7
Section 2.4: Scope	8
Section 2.5: Removing a variable	8
Chapter 3: Operators	9
Section 3.1: Comparison Operators	9
Section 3.2: Arithmetic Operators	9
Section 3.3: Assignment Operators	10
Section 3.4: Redirection Operators	10
Section 3.5: Mixing operand types, the type of the left operand dictates the behavior	11
Section 3.6: Logical Operators	11
Section 3.7: String Manipulation Operators	11
Chapter 4: Special Operators	13
Section 4.1: Array Expression Operator	13
Section 4.2: Call Operation	13
Section 4.3: Dot sourcing operator	13
Chapter 5: Basic Set Operations	14
Section 5.1: Filtering: Where-Object / where / ?	14
Section 5.2: Ordering: Sort-Object / sort	14
Section 5.3: Grouping: Group-Object / group	15
Section 5.4: Projecting: Select-Object / select	16
Chapter 6: Conditional logic	17
Section 6.1: if, else and else if	17
Section 6.2: Negation	17
Section 6.3: If conditional shorthand	18
Chapter 7: Loops	19
Section 7.1: Foreach	19
Section 7.2: For	19
Section 7.3: ForEach() Method	19
Section 7.4: ForEach-Object	20
Section 7.5: Continue	21
Section 7.6: Break	21
Section 7.7: While	22
Section 7.8: Do	22
Chapter 8: Switch statement	24

Section 8.1: Simple Switch	24
Section 8.2: Switch Statement with CaseSensitive Parameter	24
Section 8.3: Switch Statement with Wildcard Parameter	24
Section 8.4: Switch Statement with File Parameter	25
Section 8.5: Simple Switch with Default Condition	25
Section 8.6: Switch Statement with Regex Parameter	26
Section 8.7: Simple Switch With Break	26
Section 8.8: Switch Statement with Exact Parameter	27
Section 8.9: Switch Statement with Expressions	27
Chapter 9: Strings	28
Section 9.1: Multiline string	28
Section 9.2: Here-string	28
Section 9.3: Concatenating strings	28
Section 9.4: Special characters	29
Section 9.5: Creating a basic string	29
Section 9.6: Format string	30
Chapter 10: HashTables	31
Section 10.1: Access a hash table value by key	31
Section 10.2: Creating a Hash Table	31
Section 10.3: Add a key value pair to an existing hash table	31
Section 10.4: Remove a key value pair from an existing hash table	32
Section 10.5: Enumerating through keys and Key-Value Pairs	32
Section 10.6: Looping over a hash table	32
Chapter 11: Working with Objects	34
Section 11.1: Examining an object	34
Section 11.2: Updating Objects	35
Section 11.3: Creating a new object	35
Section 11.4: Creating Instances of Generic Classes	37
Chapter 12: PowerShell Functions	39
Section 12.1: Basic Parameters	39
Section 12.2: Advanced Function	39
Section 12.3: Mandatory Parameters	41
Section 12.4: Parameter Validation	41
Section 12.5: Simple Function with No Parameters	43
Chapter 13: PowerShell Classes	44
Section 13.1: Listing available constructors for a class	44
Section 13.2: Methods and properties	45
Section 13.3: Constructor overloading	45
Section 13.4: Get All Members of an Instance	46
Section 13.5: Basic Class Template	46
Section 13.6: Inheritance from Parent Class to Child Class	47
Chapter 14: PowerShell Modules	48
Section 14.1: Create a Module Manifest	48
Section 14.2: Simple Module Example	48
Section 14.3: Exporting a Variable from a Module	49
Section 14.4: Structuring PowerShell Modules	49
Section 14.5: Location of Modules	49
Section 14.6: Module Member Visibility	49
Chapter 15: PowerShell profiles	50

Section 15.1: Create an basic profile	50
Chapter 16: Calculated Properties	51
Section 16.1: Display file size in KB - Calculated Properties	51
Chapter 17: Using existing static classes	52
Section 17.1: Adding types	52
Section 17.2: Using the .Net Math Class	52
Section 17.3: Creating new GUID instantly	52
Chapter 18: Built-in variables	54
Section 18.1: \$PSScriptRoot	54
Section 18.2: \$Args	54
Section 18.3: \$PSItem	54
Section 18.4: \$?	54
Section 18.5: \$error	54
Chapter 19: Automatic Variables	56
Section 19.1: \$OFS	56
Section 19.2: \$?	56
Section 19.3: \$null	56
Section 19.4: \$error	57
Section 19.5: \$pid	57
Section 19.6: Boolean values	57
Section 19.7: \$ / \$PSItem	58
Section 19.8: \$PSVersionTable	58
Chapter 20: Environment Variables	59
Section 20.1: Windows environment variables are visible as a PS drive called Env:	59
Section 20.2: Instant call of Environment Variables with \$env:	59
Chapter 21: Splatting	60
Section 21.1: Piping and Splatting	60
Section 21.2: Passing a Switch parameter using Splatting	60
Section 21.3: Splatting From Top Level Function to a Series of Inner Function	61
Section 21.4: Splatting parameters	61
Chapter 22: PowerShell "Streams"; Debug, Verbose, Warning, Error, Output and Information	63
Section 22.1: Write-Output	63
Section 22.2: Write Preferences	63
Chapter 23: Sending Email	65
Section 23.1: Send-MailMessage with predefined parameters	65
Section 23.2: Simple Send-MailMessage	66
Section 23.3: SMTPClient - Mail with .txt file in body message	66
Chapter 24: PowerShell Remoting	67
Section 24.1: Connecting to a Remote Server via PowerShell	67
Section 24.2: Run commands on a Remote Computer	67
Section 24.3: Enabling PowerShell Remoting	69
Section 24.4: A best practise for automatically cleaning-up PSSessions	70
Chapter 25: Working with the PowerShell pipeline	71
Section 25.1: Writing Functions with Advanced Lifecycle	71
Section 25.2: Basic Pipeline Support in Functions	71
Section 25.3: Working concept of pipeline	72
Chapter 26: PowerShell Background Jobs	73

Section 26.1: Basic job creation	73
Section 26.2: Basic job management	73
Chapter 27: Return behavior in PowerShell	75
Section 27.1: Early exit	75
Section 27.2: Gotcha! Return in the pipeline	75
Section 27.3: Return with a value	75
Section 27.4: How to work with functions returns	75
Section 27.5: Gotcha! Ignoring unwanted output	77
Chapter 28: CSV parsing	78
Section 28.1: Basic usage of Import-Csv	78
Section 28.2: Import from CSV and cast properties to correct type	78
Chapter 29: Working with XML Files	80
Section 29.1: Accessing an XML File	80
Section 29.2: Creating an XML Document using XmlWriter()	81
Section 29.3: Adding snippets of XML to current XmlDocument	82
Chapter 30: Communicating with RESTful APIs	88
Section 30.1: Post Message to hipChat	88
Section 30.2: Using REST with PowerShell Objects to GET and POST many items	88
Section 30.3: Use Slack.com Incoming Webhooks	88
Section 30.4: Using REST with PowerShell Objects to Get and Put individual data	89
Section 30.5: Using REST with PowerShell to Delete items	89
Chapter 31: PowerShell SQL queries	90
Section 31.1: SQLExample	90
Section 31.2: SQLQuery	90
Chapter 32: Regular Expressions	91
Section 32.1: Single match	91
Section 32.2: Replace	93
Section 32.3: Replace text with dynamic value using a MatchEvaluator	93
Section 32.4: Escape special characters	94
Section 32.5: Multiple matches	94
Chapter 33: Aliases	97
Section 33.1: Get-Alias	97
Section 33.2: Set-Alias	97
Chapter 34: Using the progress bar	98
Section 34.1: Simple use of progress bar	98
Section 34.2: Usage of inner progress bar	98
Chapter 35: PowerShell.exe Command-Line	100
Section 35.1: Executing a command	100
Section 35.2: Executing a script file	101
Chapter 36: Cmdlet Naming	102
Section 36.1: Verbs	102
Section 36.2: Nouns	102
Chapter 37: Running Executables	103
Section 37.1: GUI Applications	103
Section 37.2: Console Streams	103
Section 37.3: Exit Codes	103
Chapter 38: Enforcing script prerequisites	104
Section 38.1: Enforce minimum version of PowerShell host	104

[Click here to download full PDF material](#)