

# Using PowerShell Script Blocks

---



**Jeff Hicks**

AUTHOR/TEACHER

@jeffhicks | <https://jdhitsolutions.com>



# Script Block

A collection of statements or expressions that can be used as a single unit. A script block can use parameters and write to the pipeline.

.





A script block is core PowerShell element

- Where-Object
- ForEach-Object
- Invoke-Command
- Ad-hoc commands

It is a way of defining a block of commands that you want to execute

You'll see references to "scriptblocks" and "script blocks"

Complex script blocks often used in PowerShell scripting



```
PS C:\> $sb = { Get-Service | where {$_.status -eq 'running'} }
```

## Creating a Script Block

Place your code inside { }



```
PS C:\> $sb = { Get-Service | where {$_.status -eq 'running'} }
```

## Creating a Script Block

Place your code inside { }

You can have as much code as you want



```
PS C:\> & $sb
```

## Running a Script Block

Use the Invoke operator



```
PS C:\> & $sb
```

Status	Name	DisplayName
-----	-----	-----
Running	Appinfo	Application Information
Running	Apple Mobile Devi?	Apple Mobile Device Service
Running	AudioEndpointBuil?	Windows Audio Endpoint Builder
Running	Audiosrv	Windows Audio
...		

## Running a Script Block

Use the Invoke operator



```
PS C:\> & $sb
```

Status	Name	DisplayName
-----	-----	-----
Running	Appinfo	Application Information
Running	Apple Mobile Devi?	Apple Mobile Device Service
Running	AudioEndpointBuil?	Windows Audio Endpoint Builder
Running	Audiosrv	Windows Audio
...		

## Running a Script Block

Use the Invoke operator

The output is the same as if you had manually run the code in the script block

The script block makes it simple to re-use





```
PS C:\> Invoke-Command $sb
```

## Running a Script Block

Use the Invoke-Command

Very useful when it comes to running commands over PowerShell remoting



```
Get-Process |  
Where-Object {$_.ws -ge 100MB} |  
Select-Object ID,Name,WS
```

Id	Name	WS
--	----	--
11696	Code	138956800
22480	Dropbox	209121280
2940	ekrn	324157440
2932	esrv_svc	155602944
9364	explorer	200912896
1696	firefox	409370624
6640	firefox	314294272
18724	googledrivesync	178024448
26160	OneDrive	112345088
...		

- ◀ Script blocks are used everywhere in PowerShell
- ◀ Where-Object uses a filtering script block



```
$new = @{  
    Path      = "C:\Work"  
    ItemType = "Directory"  
    Force     = $True  
}  
  
1..10 | foreach-object {  
    New-Item -Name "Data-$_" @new  
}
```

- ◀ Foreach item run the code in the script block
- ◀ Don't confuse a hashtable with a script block



```
PS C:\> $sb = {Param($log,$count) Get-WinEvent -log $log -max $count}
```

## Using Parameters

Add a Param() block

Parameters are all positional



```
PS C:\> $sb = { Param($log,$count) Get-WinEvent -log $log -max $count }  
PS C:\> &$sb system 2 | Format-List ProviderName,ID,LevelDisplayName,Message
```

## Using Parameters

You should specify all parameter values

You can use the & operator

Parameters separated by spaces



```
PS C:\> &$sb system 2 | Format-List ProviderName,ID,LevelDisplayName,Message
```

```
ProviderName      : Microsoft-Windows-Hyper-V-VmSwitch  
Id                : 233  
LevelDisplayName  : Information  
Message           : The operation 'Delete' succeeded on nic 612425AC-7915...
```

```
ProviderName      : Microsoft-Windows-Hyper-V-VmSwitch  
Id                : 234  
LevelDisplayName  : Information  
Message           : NIC 612425AC-7915-46D5-B24E-615F2D46AA2F successfully...
```

## Using Parameters

You should specify all parameter values

You can use the & operator

Parameters separated by spaces



```
PS C:\> $sb = {Param($log,$count) Get-WinEvent -log $log -max $count}  
PS C:\> Invoke-Command -ScriptBlock $sb -ArgumentList System,2
```

## Using Parameters

Or use Invoke-Command

Parameters separated by commas



# Using a Job Script Block

```
Start-Job {  
    param($log,$count)  
    Get-WinEvent -FilterHashtable @{  
        Logname = $log  
        SuppressHashFilter = @{Level=4}  
    } -MaxEvents $count |  
    Group-Object ProviderName -NoElement |  
    Sort-Object Count -Descending  
} -ArgumentList System,1000 -Name LogInfo
```





# Getting Job Results

```
Receive-Job loginfo -Keep | Format-Table -AutoSize
```

Count	Name
-------	------

-----	-----
-------	-------

273	Microsoft-Windows-Hyper-V-VmSwitch
-----	------------------------------------

188	Microsoft-Windows-DistributedCOM
-----	----------------------------------

103	Service Control Manager
-----	-------------------------

71	Microsoft-Windows-Kernel-General
----	----------------------------------

59	Microsoft-Windows-Kernel-Processor-Power
----	--

40	Microsoft-Windows-FilterManager
----	---------------------------------

32	Netwtw10
----	----------

21	Microsoft-Windows-Time-Service
----	--------------------------------

18	Microsoft-Windows-DHCPv6-Client
----	---------------------------------

...



# Functions in the Future

```
Function Get-LogInfo {  
    param($log, $count)  
    Get-WinEvent -FilterHashtable @{  
        Logname           = $log  
        SuppressHashFilter = @{Level = 4}  
    } -MaxEvents $count |  
    Group-Object ProviderName -NoElement |  
    Sort-Object Count -Descending  
}
```

This can be developed further into a rich PowerShell command



## Key Points



Script Blocks are used often in PowerShell

They are treated as units of code

They can use parameters

They can write to the pipeline

You can create your own for ad-hoc work

Don't focus on script blocks alone – recognize them when you see them

