

# Full credit is given to all the above companies including the Operating System that this PDF file was generated!

# Windows PowerShell Get-Help on Cmdlet 'Set-PSBreakpoint'

PS:\>Get-HELP Set-PSBreakpoint -Full

NAME

Set-PSBreakpoint

#### SYNOPSIS

Sets a breakpoint on a line, command, or variable.

# SYNTAX

Set-PSBreakpoint [[-Script] <System.String[]>] [-Line] <System.Int32[]> [[-Column] <System.Int32>] [-Action <System.Management.Automation.ScriptBlock>]

[<CommonParameters>]

Set-PSBreakpoint [[-Script] <System.String[]>] [-Action <System.Management.Automation.ScriptBlock>] -Command <System.String[]> [<CommonParameters>]

Set-PSBreakpoint [[-Script] <System.String[]>] [-Action <System.Management.Automation.ScriptBlock>] [-Mode {Read | Write | ReadWrite}] -Variable <System.String[]>

[<CommonParameters>]

#### DESCRIPTION

The `Set-PSBreakpoint` cmdlet sets a breakpoint in a script or in any command run in the current session. You can use

`Set-PSBreakpoint` to set a breakpoint before

executing a script or running a command, or during debugging, when stopped at another breakpoint.

`Set-PSBreakpoint` cannot set a breakpoint on a remote computer. To debug a script on a remote computer, copy the script to the local computer and then debug it

locally.

Each `Set-PSBreakpoint` command creates one of the following three types of breakpoints:

- Line breakpoint - Sets breakpoints at particular line and column coordinates.

- Command breakpoint - Sets breakpoints on commands and functions.

- Variable breakpoint - Sets breakpoints on variables.

You can set a breakpoint on multiple lines, commands, or variables in a single `Set-PSBreakpoint` command, but each `Set-PSBreakpoint` command sets only one type of

breakpoint.

At a breakpoint, PowerShell temporarily stops executing and gives control to the debugger. The command prompt changes to `DBG>`, and a set of debugger commands become

available for use. However, you can use the Action parameter to specify an alternate response, such as conditions for the breakpoint or instructions to perform

additional tasks such as logging or diagnostics.

The `Set-PSBreakpoint` cmdlet is one of several cmdlets designed for debugging PowerShell scripts. For more information about the PowerShell debugger, see

about\_Debuggers (../Microsoft.PowerShell.Core/About/about\_Debuggers.md).

#### PARAMETERS

-Action <System.Management.Automation.ScriptBlock>

Specifies commands that run at each breakpoint instead of breaking. Enter a script block that contains the commands.

You can use this parameter to set conditional

breakpoints or to perform other tasks, such as testing or logging.

If this parameter is omitted, or no action is specified, execution stops at the breakpoint, and the debugger starts.

When the Action parameter is used, the Action script block runs at each breakpoint. Execution does not stop unless the script block includes the Break keyword. If

you use the Continue keyword in the script block, execution resumes until the next breakpoint.

For more information, see about\_Script\_Blocks (../Microsoft.PowerShell.Core/About/about\_Script\_Blocks.md), about\_Break

(../Microsoft.PowerShell.Core/About/about\_Break.md), and about\_Continue (../Microsoft.PowerShell.Core/About/about\_Continue.md).

Required?falsePosition?namedDefault valueNoneAccept pipeline input?FalseAccept wildcard characters?false

#### -Column <System.Int32>

Specifies the column number of the column in the script file on which execution stops. Enter only one column number. The default is column 1.

The Column value is used with the value of the Line parameter to specify the breakpoint. If the Line parameter specifies multiple lines, the Column parameter sets

a breakpoint at the specified column on each of the specified lines. PowerShell stops executing before the statement or expression that includes the character at

the specified line and column position.

Columns are counted from the top left margin beginning with column number 1 (not 0). If you specify a column that does not exist in the script, an error is not

declared, but the breakpoint is never executed.

Required? false

Position? 2

Default value 1

Accept pipeline input? False

Accept wildcard characters? false

#### -Command <System.String[]>

Sets a command breakpoint. Enter cmdlet names, such as `Get-Process`, or function names. Wildcards are permitted.

Execution stops just before each instance of each command is executed. If the command is a function, execution stops each time the function is called and at each

BEGIN, PROCESS, and END section.

Required?	true	
Position?	named	
Default value	None	
Accept pipeline in	nput? False	
Accept wildcard characters? true		

#### -Line <System.Int32[]>

Sets a line breakpoint in a script. Enter one or more line numbers, separated by commas. PowerShell stops immediately before executing the statement that begins

on each of the specified lines.

Lines are counted from the top left margin of the script file beginning with line number 1 (not 0). If you specify a blank line, execution stops before the next

non-blank line. If the line is out of range, the breakpoint is never hit.

true

Position?

Default value None

Accept pipeline input? False

Accept wildcard characters? false

1

-Mode <System.Management.Automation.VariableAccessMode>

Specifies the mode of access that triggers variable breakpoints. The default is Write .

This parameter is valid only when the Variable parameter is used in the command. The mode applies to all breakpoints set in the command. The acceptable values for

this parameter are:

- Write - Stops execution immediately before a new value is written to the variable. - Read - Stops execution when the variable is read, that is, when its value

is accessed, either to be assigned, displayed, or used. In read mode, execution does not stop when the value of the variable changes. - ReadWrite - Stops

execution when the variable is read or written.

Required?	false
Position?	named
Default value	None
Accept pipeline ir	nput? False
Accept wildcard	characters? false

#### -Script <System.String[]>

Specifies an array of script files that this cmdlet sets a breakpoint in. Enter the paths and file names of one or more script files. If the files are in the

current directory, you can omit the path. Wildcards are permitted.

By default, variable breakpoints and command breakpoints are set on any command that runs in the current session.

This parameter is required only when setting a

line breakpoint.

Required?falsePosition?0Default valueNoneAccept pipeline input?FalseAccept wildcard characters?false

#### -Variable <System.String[]>

Specifies an array of variables that this cmdlet sets breakpoints on. Enter a comma-separated list of variables without dollar signs (`\$`).

Use the Mode parameter to determine the mode of access that triggers the breakpoints. The default mode, Write, stops execution just before a new value is written

to the variable.

Required?	true	
Position?	named	
Default value	None	
Accept pipeline in	nput? False	
Accept wildcard characters? false		

## <CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug,

ErrorAction, ErrorVariable, WarningAction, WarningVariable,

OutBuffer, PipelineVariable, and OutVariable. For more information, see

about\_CommonParameters (https:/go.microsoft.com/fwlink/?LinkID=113216).

# INPUTS

#### None

You can't pipe objects to this cmdlet.

#### OUTPUTS

System.Management.Automation.CommandBreakpoint

#### System.Management.Automation.LineBreakpoint

#### System.Management.Automation.VariableBreakpoint

`Set-PSBreakpoint` returns an object that represents each breakpoint that it sets.

#### NOTES

Windows PowerShell includes the following aliases for `Set-PSBreakpoint`:

- `sbp`

- `Set-PSBreakpoint` cannot set a breakpoint on a remote computer. To debug a script on a remote computer, copy the script to the local computer and then debug

it locally. - When you set a breakpoint on more than one line, command, or variable, `Set-PSBreakpoint` generates a breakpoint object for each entry. - When

setting a breakpoint on a function or variable at the command prompt, you can set the breakpoint before or after you create the function or variable.

----- Example 1: Set a breakpoint on a line -----

Set-PSBreakpoint -Script "sample.ps1" -Line 5

Column : 0

Line : 5

Action :

Enabled : True

HitCount : 0

ld : 0

Script : C:\ps-test\sample.ps1

ScriptName : C:\ps-test\sample.ps1

When you set a new breakpoint by line number, the `Set-PSBreakpoint` cmdlet generates a line breakpoint object ( System.Management.Automation.LineBreakpoint ) that

includes the breakpoint ID and hit count.

----- Example 2: Set a breakpoint on a function ------

Set-PSBreakpoint -Command "Increment" -Script "sample.ps1"

Command : Increment

Action :

Enabled : True

HitCount : 0

ld : 1

Script : C:\ps-test\sample.ps1

ScriptName : C:\ps-test\sample.ps1

The result is a command breakpoint object. Before the script runs, the value of the HitCount property is 0. ------ Example 3: Set a breakpoint on a variable ------

Set-PSBreakpoint -Script "sample.ps1" -Variable "Server" -Mode ReadWrite

Example 4: Set a breakpoint on every command that begins with specified text

Set-PSBreakpoint -Script Sample.ps1 -Command "write\*"

Example 5: Set a breakpoint depending on the value of a variable

Set-PSBreakpoint -Script "test.ps1" -Command "DiskTest" -Action { if (\$Disk -gt 2) { break } }

The value of the Action is a script block that tests the value of the `\$Disk` variable in the function.

The action uses the `break` keyword to stop execution if the condition is met. The alternative (and the default) is Continue

----- Example 6: Set a breakpoint on a function ------

```
PS> Set-PSBreakpoint -Command "checklog"
```

ld : 0

```
Command : checklog
```

Enabled : True

HitCount: 0

```
Action :
```

function CheckLog {

```
>> get-eventlog -log Application |
```

```
>> where {($_.source -like "TestApp") - and ($_.Message -like "*failed*")}
```

>>}

```
>>
```

```
PS> Checklog
```

```
DEBUG: Hit breakpoint(s)
```

DEBUG: Function breakpoint on 'prompt:Checklog'

----- Example 7: Set breakpoints on multiple lines ------

PS C:\> Set-PSBreakpoint -Script "sample.ps1" -Line 1, 14, 19 -Column 2 -Action {&(log.ps1)}

Column : 2

Line : 1

Action :

Enabled : True

HitCount : 0

ld :6

Script : C:\ps-test\sample.ps1

ScriptName : C:\ps-test\sample.ps1

Column : 2

Line : 14

Action :

Enabled : True

HitCount : 0

ld : 7

Script : C:\ps-test\sample.ps1

ScriptName : C:\ps-test\sample.ps1

Column : 2

Line : 19

Action :

Enabled : True

HitCount : 0

ld : 8

Script : C:\ps-test\sample.ps1

ScriptName : C:\ps-test\sample.ps1

## **RELATED LINKS**

Online

Version:

 $https://learn.microsoft.com/powershell/module/microsoft.powershell.utility/set-psbreakpoint?view=powershell-5.1\&WT.mc\_idwt.psbreakpoint?view=powershell-5.1\&WT.mc\_idwt.psbreakpoint?view=powershell-5.1\&WT.mc\_idwt.psbreakpoint?view=powershell-5.1\&WT.mc\_idwt.psbreakpoint?view=powershell-5.1\&WT.mc\_idwt.psbreakpoint?view=powershell-5.1\&WT.mc\_idwt.psbreakpoint?view=powershell-5.1\&WT.mc\_idwt.psbreakpoint?view=powershell-5.1\&WT.mc\_idwt.psbreakpoint?view=powershell-5.1\&WT.mc\_idwt.psbreakpoint?view=powershell-5.1\&WT.mc\_idwt.psbreakpoint?view=powershell-5.1\&WT.mc\_idwt.psbreakpoint?view=powershell-5.1\&WT.mc\_idwt.psbreakpoint?view=powershell-5.1\&WT.mc\_idwt.psbreakpoint?view=powershell-5.1\&WT.mc\_idwt.psbreakpoint.psbreakpoint?view=powershell-5.1\&WT.mc\_idwt.psbreakpoint.psbre$ 

=ps-gethelp

Disable-PSBreakpoint

Enable-PSBreakpoint

Get-PSBreakpoint

Get-PSCallStack

Remove-PSBreakpoint