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 'Enable-PSBreakpoint'

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

NAME

Enable-PSBreakpoint

SYNOPSIS

Enables the breakpoints in the current console.

SYNTAX

Enable-PSBreakpoint [-Breakpoint] <System.Management.Automation.Breakpoint[]> [-PassThru] [-Confirm] [-WhatIf] [<CommonParameters>]

Enable-PSBreakpoint [-Id] <System.Int32[]> [-PassThru] [-Confirm] [-WhatIf] [<CommonParameters>]

DESCRIPTION

The `Enable-PSBreakpoint` cmdlet re-enables disabled breakpoints. You can use it to enable all breakpoints, or specific breakpoints by providing breakpoint objects or IDs.

A breakpoint is a point in a script where execution stops temporarily so that you can examine the state of the script.

Newly created breakpoints are automatically

enabled, but can be disabled using `Disable-PSBreakpoint`.

Technically, this cmdlet changes the value of the Enabled property of a breakpoint object to True .

`Enable-PSBreakpoint` 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

-Breakpoint <System.Management.Automation.Breakpoint[]>

Specifies the breakpoints to enable. Provide a variable containing breakpoints or a command that gets breakpoint objects, such as `Get-PSBreakpoint`. You can also

pipe breakpoint objects to `Enable-PSBreakpoint`.

```
Required?                    true
Position?                    0
Default value                None
Accept pipeline input?       True (ByValue)
Accept wildcard characters?  false
```

-Id <System.Int32[]>

Specifies the Id numbers of the breakpoints to enable. The default value is all breakpoints. Provide the Id by number or in a variable. You can't pipe Id numbers

to `Enable-PSBreakpoint`. To find the Id of a breakpoint, use the `Get-PSBreakpoint` cmdlet.

```
Required?                    true
Position?                    0
Default value                None
Accept pipeline input?       True (ByPropertyName)
Accept wildcard characters?  false
```

```
-PassThru <System.Management.Automation.SwitchParameter>
    Returns an object representing the breakpoint being enabled. By default, this cmdlet doesn't generate any output.


    Required?                    false
    Position?                    named
    Default value                False
    Accept pipeline input?       False
    Accept wildcard characters?  false


-Confirm <System.Management.Automation.SwitchParameter>
    Prompts you for confirmation before running the cmdlet.


    Required?                    false
    Position?                    named
    Default value                False
    Accept pipeline input?       False
    Accept wildcard characters?  false


-WhatIf <System.Management.Automation.SwitchParameter>
    Shows what would happen if the cmdlet runs. The cmdlet isn't run.


    Required?                    false
    Position?                    named
    Default value                False
    Accept pipeline input?       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

System.Management.Automation.Breakpoint

You can pipe a breakpoint object to this cmdlet.

OUTPUTS

None

By default, this cmdlet returns no output.

System.Management.Automation.Breakpoint

When you use the PassThru parameter, this cmdlet returns a breakpoint object representing the enabled breakpoint.

NOTES

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

- `ebp`

- The `Enable-PSBreakpoint` cmdlet doesn't generate an error if you try to enable a breakpoint that is already enabled. As such, you can enable all breakpoints
without error, even when only a few are disabled.

- Breakpoints are enabled when you create them by using the `Set-PSBreakpoint` cmdlet. You don't need to enable newly created breakpoints.

-------------- Example 1: Enable all breakpoints --------------

```
Get-PSBreakpoint | Enable-PSBreakpoint
```

Using aliases, this example can be abbreviated as `gbp | ebp`.

------------- Example 2: Enable breakpoints by ID -------------

```
Enable-PSBreakpoint -Id 0, 1, 5
```

----------- Example 3: Enable a disabled breakpoint -----------

```
$B = Set-PSBreakpoint -Script "sample.ps1" -Variable Name -PassThru
$B | Enable-PSBreakpoint -PassThru

AccessMode : Write
Variable   : Name
Action     :
Enabled    : False
HitCount   : 0
Id         : 0
Script     : C:\ps-test\sample.ps1
ScriptName : C:\ps-test\sample.ps1

AccessMode : Write
Variable   : Name
Action     :
Enabled    : True
HitCount   : 0
Id         : 0
Script     : C:\ps-test\sample.ps1
ScriptName : C:\ps-test\sample.ps1
```

`Set-PSBreakpoint` creates a breakpoint on the Name variable in the `Sample.ps1` script saving the breakpoint object in the `$B` variable. The PassThru parameter
displays the value of the Enabled property of the breakpoint is False .

`Enable-PSBreakpoint` re-enables the breakpoint. Again, using the PassThru parameter we see that the value of the Enabled property is True .

-------- Example 4: Enable breakpoints using a variable --------

```
$B = Get-PSBreakpoint -Id 3, 5
Enable-PSBreakpoint -Breakpoint $B
```

`Get-PSBreakpoint` gets the breakpoints and saves them in the `$B` variable. Using the Breakpoint parameter, `Enable-PSBreakpoint` enables the breakpoints.

This example is equivalent to running `Enable-PSBreakpoint -Id 3, 5`.

RELATED LINKS

Online Version: https://learn.microsoft.com/powershell/module/microsoft.powershell.utility/enable-psbreakpoint?view=powershell-5.1&WT.mc_id=ps-gethelp

Disable-PSBreakpoint

Get-PSBreakpoint

Get-PSCallStack

Remove-PSBreakpoint

Set-PSBreakpoint