MyWebUniversity
python
PowerShell Get-Help cmdlet
PS C:\> Get-Help
FPDF Library
PDF generator
FPDF
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-StrictMode'

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

NAME

Set-StrictMode

SYNOPSIS

Establishes and enforces coding rules in expressions, scripts, and script blocks.

SYNTAX

Set-StrictMode -Off [<CommonParameters>]

Set-StrictMode -Version <System.Version> [<CommonParameters>]

DESCRIPTION

The `Set-StrictMode` cmdlet configures strict mode for the current scope and all child scopes, and turns it on and off. When strict mode is on, PowerShell generates a

terminating error when the content of an expression, script, or script block violates basic best-practice coding rules.

Use the Version parameter to determine the coding rules to enforce.

`Set-PSDebug -Strict` cmdlet turns on strict mode for the global scope. `Set-StrictMode` affects only the current scope and its child scopes. Then, you can use it in
a script or function to override the setting inherited from the global scope.

When `Set-StrictMode` is off, PowerShell has the following behaviors:

- Uninitialized variables are assumed to have a value of `0` (zero) or `$Null`, depending on type

- References to non-existent properties return `$Null`

- Results of improper function syntax vary with the error conditions

- Attempting to retrieve a value using an invalid index in an array returns `$Null`

PARAMETERS

-Off <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet turns strict mode off for the current scope and all child scopes.

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

-Version <System.Version>

Specifies the conditions that cause an error in strict mode. This parameter accepts any valid PowerShell version number. Any number higher than `3` is treated as
`Latest`. The value supplied must be the string `Latest` or a string that can be converted to a System.Version type. The version must match a valid release
version of PowerShell.

The effective values for this parameter are:

- `1.0` - Prohibits references to uninitialized variables, except for uninitialized variables in strings. - `2.0` - Prohibits references to uninitialized
variables. This includes uninitialized variables in strings. - Prohibits references to non-existent properties of an object. - Prohibits function calls
that use the syntax for calling methods. - `3.0` - Prohibits references to uninitialized variables. This includes uninitialized variables in strings. -
Prohibits references to non-existent properties of an object. - Prohibits function calls that use the syntax for calling methods. - Prohibit out of bounds or
unresolvable array indexes. - `Latest` - Selects the latest version available. The latest version is the most strict. Use this value to make sure that
scripts use the strictest available version, even when new versions are added to PowerShell.

> [!CAUTION] > Using `Latest` for Version in scripts isn't deterministic. The meaning of `Latest` can change > in new releases of PowerShell. A script written for
an older version of PowerShell that uses > `Set-StrictMode -Version Latest` is subject to more restrictive rules when run in a newer version > of PowerShell.

```
Required?                    true
Position?                    named
Default value                None
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

None

You can't pipe objects to this cmdlet.

OUTPUTS

    None

        This cmdlet returns no output.

NOTES

        While the Version parameter accepts values greater than `3.0`, there are no additional rules defined for anything higher than `3.0`.

        `Set-StrictMode` is effective only in the scope it's set in and in its child scopes. For more information about scopes in PowerShell, see about_Scopes

        (about/about_Scopes.md).

```
    -------- Example 1: Turn on strict mode as version 1.0 --------

    # Strict mode is off by default.
    $a -gt 5

    False

    Set-StrictMode -Version 1.0
    $a -gt 5

    The variable $a cannot be retrieved because it has not been set yet.

    At line:1 char:3
    + $a <<<<  -gt 5
    + CategoryInfo          : InvalidOperation: (a:Token) [], RuntimeException
    + FullyQualifiedErrorId : VariableIsUndefined
```

With strict mode set to version `1.0`, attempts to reference variables that aren't initialized fail.

-------- Example 2: Turn on strict mode as version 2.0 --------

```
# Strict mode is off by default.
function add ($a, $b) {
    '$a = ' + $a
    '$b = ' + $b
    '$a+$b = ' + ($a + $b)
}
add 3 4

$a = 3
$b = 4
$a+$b = 7

add(3,4)

$a = 3 4
$b =
$a+$b = 3 4

Set-StrictMode -Version 2.0
add(3,4)

The function or command was called like a method. Parameters should be separated by spaces,
as described in 'Get-Help about_Parameter.'
At line:1 char:4
+ add <<<< (3,4)
+ CategoryInfo          : InvalidOperation: (:) [], RuntimeException
+ FullyQualifiedErrorId : StrictModeFunctionCallWithParens

Set-StrictMode -Off
```

```
    $string = "This is a string."
    $null -eq $string.Month

    True


    Set-StrictMode -Version 2.0
    $string = "This is a string."
    $null -eq $string.Month

    Property 'Month' cannot be found on this object; make sure it exists.
    At line:1 char:9
    + $string. <<<< month
    + CategoryInfo          : InvalidOperation: (.:OperatorToken) [], RuntimeException
    + FullyQualifiedErrorId : PropertyNotFoundStrict
```

This command turns strict mode on and sets it to version `2.0`. As a result, PowerShell returns an error if you use method syntax, which uses parentheses and commas,
for a function call or reference uninitialized variables or non-existent properties.

The sample output shows the effect of version `2.0` strict mode.

Without version `2.0` strict mode, the `(3,4)` value is interpreted as a single array object to which nothing is added. With version `2.0` strict mode, it's correctly
interpreted as faulty syntax for submitting two values.

Without version `2.0`, the reference to the non-existent Month property of a string returns only `$Null`. With version `2.0`, it's interpreted correctly as a
reference error.

-------- Example 3: Turn on strict mode as version 3.0 --------

```
    # Strict mode is off by default.
    $a = @(1)
    $null -eq $a[2]
```

```
$null -eq $a['abc']


True

True


Set-StrictMode -Version 3.0

$a = @(1)

$null -eq $a[2]

$null -eq $a['abc']


Index was outside the bounds of the array.

At line:1 char:1

+ $null -eq $a[2]

+ ~~~~~~~~~~~~~~~

    + CategoryInfo          : OperationStopped: (:) [], IndexOutOfRangeException

    + FullyQualifiedErrorId : System.IndexOutOfRangeException


Cannot convert value "abc" to type "System.Int32". Error: "Input string was not in a correct format."

At line:1 char:1

+ $null -eq $a['abc']

+ ~~~~~~~~~~~~~~~~~~~

    + CategoryInfo          : InvalidArgument: (:) [], RuntimeException

    + FullyQualifiedErrorId : InvalidCastFromStringToInteger
```

With strict mode set to version `3` or higher, invalid or out of bounds indexes result in errors.

RELATED LINKS

Online Version: https://learn.microsoft.com/powershell/module/microsoft.powershell.core/set-strictmode?view=powershell-5.1&WT.mc_id=ps-gethelp

Set-PSDebug