Full credit is given to the above companies including the OS that this PDF file was generated!

PowerShell Get-Help on command 'Set-Variable'

PS C:\Users\wahid> Get-Help Set-Variable

NAME

Set-Variable

SYNOPSIS

Sets the value of a variable. Creates the variable if one with the requested name does not exist.

SYNTAX

Set-Variable [-Name] <System.String[]> [[-Value] <System.Object>] [-Description <System.String>] [-Exclude <System.String[]>] [-Force] [-Include <System.String[]>] [-Option {None | ReadOnly | Constant | Private | AllScope | Unspecified}] [-PassThru] [-Scope <System.String>] [-Visibility {Public | Private}] [-Confirm] [-WhatIf] [<CommonParameters>]

DESCRIPTION

The `Set-Variable` cmdlet assigns a value to a specified variable or changes the current value. If the variable does not exist, the cmdlet creates it.

PARAMETERS

-Description <System.String>

Specifies the description of the variable.

-Exclude <System.String[]>

Specifies an array of items that this cmdlet excludes from the operation. The value of this parameter qualifies the Path parameter. Enter a path element or pattern, such as `*.txt`. Wildcards are permitted.

-Force <System.Management.Automation.SwitchParameter>

Allows you to create a variable with the same name as an existing read-only variable, or to change the value of a read-only variable.

By default, you can overwrite a variable, unless the variable has an option value of `ReadOnly` or `Constant`. For more information, see the Option parameter.

-Include <System.String[]>

Specifies an array of items that this cmdlet includes in the operation. The value of this parameter qualifies the Name parameter. Enter a name or name pattern, such as `c*`. Wildcards are permitted.

-Name <System.String[]>

Specifies the variable name.

-Option <System.Management.Automation.ScopedItemOptions>

Specifies the value of the Options property of the variable.

Valid values are:

- `None`: Sets no options. (`None` is the default.)

- `ReadOnly`: Can be deleted. Cannot be changed, except by using the Force

parameter.

- `Constant`: Cannot be deleted or changed. `Constant` is valid only when you are creating a

variable. You cannot change the options of an existing variable to `Constant`. - `Private`: The variable is available only in the current scope.

- `AllScope`: The variable is copied to any new scopes that are created.

These values are defined as a flag-based enumeration. You can combine multiple values together to set multiple flags using this parameter. The values can be passed to the Option parameter as an array of values or as a comma-separated string of those values. The cmdlet will combine the values using a binary-OR operation. Passing values as an array is the simplest option and also allows you to use tab-completion on the values.

-PassThru <System.Management.Automation.SwitchParameter>

Returns an object representing the new variable. By default, this cmdlet does not generate any output.

-Scope <System.String>

Specifies the scope of the variable.The acceptable values for this parameter are:

- `Global`

- `Local`

- `Script`

- `Private`

- A number relative to the current scope (0 through the number of scopes,
where 0 is the current

scope and 1 is its parent).

`Local` is the default.

For more information, see about_Scopes
(../Microsoft.PowerShell.Core/About/about_scopes.md).

-Value <System.Object>
Specifies the value of the variable.

-Visibility <System.Management.Automation.SessionStateEntryVisibility>
Determines whether the variable is visible outside of the session in which
it was created. This parameter is designed for use in scripts and commands
that will be delivered to other users.

Valid values are:

- `Public`: The variable is visible. (`Public` is the default.)

- `Private`: The variable is not visible.

When a variable is private, it does not appear in lists of variables, such
as those returned by `Get-Variable`, or in displays of the Variable:
drive. Commands to read or change the value of a private variable return
an error. However, the user can run commands that use a private variable
if the commands were written in the session in which the variable was

defined.

-Confirm <System.Management.Automation.SwitchParameter>

Prompts you for confirmation before running the cmdlet.

-WhatIf <System.Management.Automation.SwitchParameter>

Shows what would happen if the cmdlet runs. The cmdlet is not run.

<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).

--------- Example 1: Set a variable and get its value ---------

Set-Variable -Name "desc" -Value "A description"

Get-Variable -Name "desc"

Name Value

---- -----

desc A description

--------- Example 2: Set a global, read-only variable ---------

Set-Variable -Name "processes" -Value (Get-Process) -Option constant -Scope

global -Description "All processes" -PassThru |

Format-List -Property *

The command uses the `Set-Variable` cmdlet to create the variable. It uses the

PassThru parameter to create an object representing the new variable, and it

uses the pipeline operator (`|`) to pass the object to the `Format-List`
cmdlet. It uses the Property parameter of `Format-List` with a value of all
(`*`) to display all properties of the newly created variable.

The value, `(Get-Process)`, is enclosed in parentheses to ensure that it is
executed before being stored in the variable. Otherwise, the variable contains
the words `Get-Process`.
------ Example 3: Understand public vs. private variables ------

```
New-Variable -Name "counter" -Visibility Public -Value 26
$Counter

26

Get-Variable c*

Name                 Value
----                 -----
Culture              en-US
ConsoleFileName
ConfirmPreference    High
CommandLineParameters {}
Counter              26

Set-Variable -Name "counter" -Visibility Private
Get-Variable c*

Name                 Value
----                 -----
Culture              en-US
ConsoleFileName
ConfirmPreference    High
CommandLineParameters {}
```

$counter

"Cannot access the variable '$counter' because it is a private variable"

.\use-counter.ps1
#Commands completed successfully.

This command shows how to change the visibility of a variable to Private. This variable can be read and changed by scripts with the required permissions, but it is not visible to the user.

REMARKS

To see the examples, type: "get-help Set-Variable -examples".
For more information, type: "get-help Set-Variable -detailed".
For technical information, type: "get-help Set-Variable -full".
For online help, type: "get-help Set-Variable -online"