Windows PowerShell Get-Help on Cmdlet 'Test-Path'

PS:\>Get-HELP Test-Path -Full

NAME

Test-Path

SYNOPSIS

Determines whether all elements of a path exist.

SYNTAX

Test-Path [-Credential <System.Management.Automation.PSCredential>] [-Exclude <System.String[]>] [-Filter <System.String>] [-Include <System.String[]>] [-IsValid]

-LiteralPath <System.String[]> [-NewerThan <System.Nullable`1[[System.DateTime]]>] [-OlderThan <System.Nullable`1[[System.DateTime]]>] [-PathType {Any | Container |

Leaf}] [-UseTransaction] [<CommonParameters>]

Test-Path [-Path] <System.String[]> [-Credential <System.Management.Automation.PSCredential>] [-Exclude <System.String[]>] [-Filter <System.String>] [-Include

<System.String[]>] [-IsValid] [-NewerThan <System.Nullable`1[[System.DateTime]]>] [-OlderThan <System.Nullable`1[[System.DateTime]]>] [-PathType {Any | Container |

Leaf}] [-UseTransaction] [<CommonParameters>]

DESCRIPTION

The `Test-Path` cmdlet determines whether all elements of the path exist. It returns `$true` if all elements exist and `$false` if any are missing. It can also tell

whether the path syntax is valid and whether the path leads to a container or a terminal or leaf element. If the Path is a whitespace or empty string, then the cmdlet

returns `$false`. If the Path is `$null`, an array of `$null` or an empty array, the cmdlet returns a non-terminating error.

PARAMETERS

-Credential <System.Management.Automation.PSCredential>

> [!NOTE] > This parameter isn't supported by any providers installed with PowerShell. To impersonate another > user, or elevate your credentials when running

this cmdlet, use > Invoke-Command (../Microsoft.PowerShell.Core/Invoke-Command.md).

Required? false

Position? named

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-Exclude <System.String[]>

Specifies items that this cmdlet omits. The value of this parameter qualifies the Path parameter. Enter a path element or pattern, such as `*.txt`. Wildcard

characters are permitted.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? true

-Filter <System.String>

Specifies a filter in the format or language of the provider. The value of this parameter qualifies the Path parameter. The syntax of the filter, including the

use of wildcard characters, depends on the provider. Filters are more efficient than other parameters, because the provider applies them when it retrieves the

objects instead of having PowerShell filter the objects after they're retrieved.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? true

-Include <System.String[]>

Specifies paths that this cmdlet tests. The value of this parameter qualifies the Path parameter. Enter a path element or pattern, such as `*.txt`. Wildcard

characters are permitted.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? true

-IsValid <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet tests the syntax of the path, regardless of whether the elements of the path exist. This cmdlet returns `$true` if the path syntax is

valid and `$false` if it's not. If the path being tested includes a drive specification, the cmdlet returns false when the drive does not exist. PowerShell

returns false because it doesn't know which drive provider to test.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-LiteralPath <System.String[]>

Specifies a path to be tested. Unlike Path , the value of the LiteralPath parameter is used exactly as it's typed. No characters are interpreted as wildcard

characters. If the path includes characters that could be interpreted by PowerShell as escape sequences, you must enclose the path in single quote so that they

won't be interpreted.

Required? true

Position? named

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-NewerThan <System.Nullable`1[[System.DateTime]]>

This is a dynamic parameter made available by the FileSystem provider.

Specify a time as a DateTime object.

Before PowerShell 7.5, the cmdlet ignores:

- This parameter when you specify PathType as any value other than `Any`. - The OlderThan parameter when used with this parameter. - This parameter when Path

points to a directory.

Starting with PowerShell 7.5, you can use this parameter with any value for the PathType parameter, to test a date range with the OlderThan parameter, and to test

the age of directories.

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

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-OlderThan <System.Nullable`1[[System.DateTime]]>

This is a dynamic parameter made available by the FileSystem provider.

Specify a time as a DateTime object.

Before PowerShell 7.5, the cmdlet ignores:

- This parameter when you specify PathType as any value other than `Any`. - This parameter when used with the NewerThan parameter. - This parameter when Path

points to a directory.

Starting with PowerShell 7.5, you can use this parameter with any value for the PathType parameter, to test a date range with the NewerThan parameter, and to test

the age of directories.

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

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-Path <System.String[]>

Specifies a path to be tested. Wildcard characters are permitted. If the path includes spaces, enclose it in quotation

marks.

Required? true

Position? 0

Default value None

Accept pipeline input? True (ByPropertyName, ByValue)

Accept wildcard characters? true

-PathType <Microsoft.PowerShell.Commands.TestPathType>

Specifies the type of the final element in the path. This cmdlet returns `$true` if the element is of the specified type and `$false` if it's not. The acceptable

values for this parameter are:

- `Container` - An element that contains other elements, such as a directory or registry key.

- `Leaf` - An element that doesn't contain other elements, such as a file.

- `Any` - Either a container or a leaf.

Tells whether the final element in the path is of a particular type.

> [!CAUTION] > > Up to PowerShell version 6.1.2, when the IsValid and PathType switches are specified > together, the `Test-Path` cmdlet ignores the PathType

switch and only validates the syntactic > path without validating the path type. > > According to issue #8607

(https://github.com/PowerShell/PowerShell/issues/8607), fixing this > behavior may be a breaking change in a future version, where the IsValid and PathType >

switches belong to separate parameter sets, and thus, can't be used together avoiding this > confusion.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-UseTransaction <System.Management.Automation.SwitchParameter>

Includes the command in the active transaction. This parameter is valid only when a transaction is in progress. For more information, see about_Transactions (../Microsoft.PowerShell.Core/About/about_Transactions.md)

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.String

You can pipe a string that contains a path, but not a literal path, to this cmdlet.

OUTPUTS

System.Boolean

The cmdlet returns a Boolean value.

NOTES

The cmdlets that contain the Path noun (the Path cmdlets) work with path and return the names in a concise format

that all PowerShell providers can interpret.

They're designed for use in programs and scripts where you want to display all or part of a path in a particular format. Use them as you would use Dirname ,

Normpath , Realpath , Join , or other path manipulators.

The `Test-Path` is designed to work with the data exposed by any provider. To list the providers available in your session, type `Get-PSProvider`. For more

information, see about_Providers (../Microsoft.PowerShell.Core/About/about_Providers.md).

---------- Example 1: Test a path ----------

Test-Path -Path "C:\Documents and Settings\DavidC"

True

This command checks whether all elements in the path exist, including the `C:` directory, the `Documents and Settings` directory, and the `DavidC` directory. If any

are missing, the cmdlet returns `$false`. Otherwise, it returns `$true`.

---------- Example 2: Test the path of a profile ----------

Test-Path -Path $profile

False

Test-Path -Path $profile -IsValid

True

These commands test the path of the PowerShell profile.

The first command determines whether all elements in the path exist. The second command determines whether the syntax of the path is correct. In this case, the path

is `$false`, but the syntax is correct `$true`. These commands use `$profile`, the automatic variable that points to the

location for the profile, even if the profile

doesn't exist.

For more information about automatic variables, see about_Automatic_Variables (../Microsoft.PowerShell.Core/About/about_Automatic_Variables.md).

Example 3: Check whether there are any files besides a specified type

Test-Path -Path "C:\CAD\Commercial Buildings*" -Exclude *.dwg

False

This command checks whether there are any files in the Commercial Buildings directory other than .dwg files.

The command uses the Path parameter to specify the path. Because the path includes a space, the path is enclosed in quotation marks. The asterisk at the end of the

path indicates the contents of the Commercial Building directory. With long paths, such as this one, type the first few letters of the path, and then use the TAB key

to complete the path.

The command specifies the Exclude parameter to specify files to omit from the evaluation.

In this case, because the directory contains only .dwg files, the result is `$false`.

---------- Example 4: Check for a file ----------

Test-Path -Path $profile -PathType leaf

True

This command checks whether the path stored in the `$profile` variable leads to a file. In this case, because the PowerShell profile is a `.ps1` file, the cmdlet

returns `$true`.

---------- Example 5: Check paths in the Registry ----------

Test-Path -Path "HKLM:\Software\Microsoft\PowerShell\1\ShellIds\Microsoft.PowerShell"

True

Test-Path -Path "HKLM:\Software\Microsoft\PowerShell\1\ShellIds\Microsoft.PowerShell\ExecutionPolicy"

False

--- Example 6: Test if a file is newer than a specified date ---

Test-Path $pshome\PowerShell.exe -NewerThan "July 13, 2009"

True

-------- Example 7: Test a path with null as the value --------

Test-Path $null

Test-Path $null, $null

Test-Path @()

Test-Path : Cannot bind argument to parameter 'Path' because it is null.

At line:1 char:11

+ Test-Path $null

+ ~~~~~

+ CategoryInfo : InvalidData: (:) [Test-Path], ParameterBindingValidationException

+ FullyQualifiedErrorId :

ParameterArgumentValidationErrorNullNotAllowed,Microsoft.PowerShell.Commands.TestPathCommand

----- Example 8: Test a path with whitespace as the value -----

```
Test-Path ' '
Test-Path ''

True
Test-Path : Cannot bind argument to parameter 'Path' because it is an empty string.
At line:1 char:11
+ Test-Path ''
+           ~~
    + CategoryInfo          : InvalidData: (:) [Test-Path], ParameterBindingValidationException
    + FullyQualifiedErrorId : ParameterArgumentValidationErrorEmptyStringNotAllowed,Microsoft.PowerShell.Commands.TestPathCommand
```

---- Example 9: Test a path that may have an invalid drive ----

```
Test-Path -IsValid Z:\abc.txt
Test-Path -IsValid FileSystem::Z:\abc.txt

False
True
```

RELATED LINKS

Online Version: https://learn.microsoft.com/powershell/module/microsoft.powershell.management/test-path?view=powershell-5.1&WT.mc_id=ps-gethelp

Convert-Path

Join-Path

Resolve-Path

Split-Path