

How to use Test-Path cmdlet in PowerShell

When working with files or folders in PowerShell, it's important to test if the path exists, before you try to read or save contents. This is where the Test-Path cmdlet comes in. It checks if all elements of a path exist and returns `$true` or `$false`.

Now the Test-Path cmdlet in PowerShell isn't limited to files and folders only. It can also be used to test the registry path, test if a file is newer than a particular date, and verify if a path is a valid path or not.

In this article, we are going to take a look at how to use the **Test-Path cmdlet** and look at the different options that you can do with it.

PowerShell Test-Path

The PowerShell Test-Path cmdlet comes with different parameters that we can use. I have listed the most commonly used parameters below:

Parameter	Description
-Path	The path to test
-Include	Specifies a path element to include, for example, *.txt
-Exclude	Specifies a path element to exclude, for example, *.txt
-PathType	Allows you to check if a path is a Container, Leaf or either
-IsValid	Check if a path is a valid path
-OlderThan	Check if the file is older than the specified datetime object
-NewerThan	Check if the file is newer than the specified datetime object

Test-Path Parameters

So to simply test if a path exists, we only need to specify the Path and the cmdlet will return either `$true` or `$false` depending if all the elements of the path exist or not:

```
Test-Path -Path "c:\Temp\TestFiles"
```

```
# Result
```

```
True
```

We can use this in our scripts with a simple if statement like this:

```
$path = "c:\Temp\TestFiles"
if (Test-Path -Path $path) {
```

```
Get-ChildItem -Path $path  
}
```

PowerShell check if File Exists

We can also use Test-Path in PowerShell to check if a file exists or not. For this, we will need to specify the full path, including the file name in the path parameter:

```
# Check if the file LineNumbers.txt exists  
$file = "C:\Temp\TestFiles\LineNumbers.txt"  
Test-Path -Path $file  
# Result  
True
```

Now, in this case, we are obviously checking for a file, because we have specified the full path including the file with the extension. But if you don't know if the path leads to a file or folder, then you can specify the PathType parameter.

Let's say we want to know if the `$profile` point to a file. For this, we set the `PathType` to `Leaf`:

```
# Check if $profile is a file:  
Test-Path -Path $profile -PathType leaf  
# Result  
True
```

PowerShell check if Folder Exists

Just like with a file, we can also check in PowerShell if a folder exists. For this we only need to specify the path to the folder:

```
Test-Path -Path "c:\Temp\TestFiles"  
# Result  
True
```

If we want to be sure that the path point to a folder, then we can add the `-PathType` parameter and set it to `container`. This way the Test-Path cmdlet will not only check if the path exists but also if it's a folder or not:

```
# Check if $profile is a folder:  
Test-Path -Path $profile -PathType container  
# Result  
False
```

If you want to know if the folder contains files, then you don't need to do a count with Get-ChildItem, etc. We can simply use `Test-Path` and add a wildcard `*` to it to check if there are files in the given path:

```
# Check if there are files in the folder  
Test-Path -Path "c:\Temp\TestFiles\*"
```

```
# Result
True
```

Using Include and Exclude

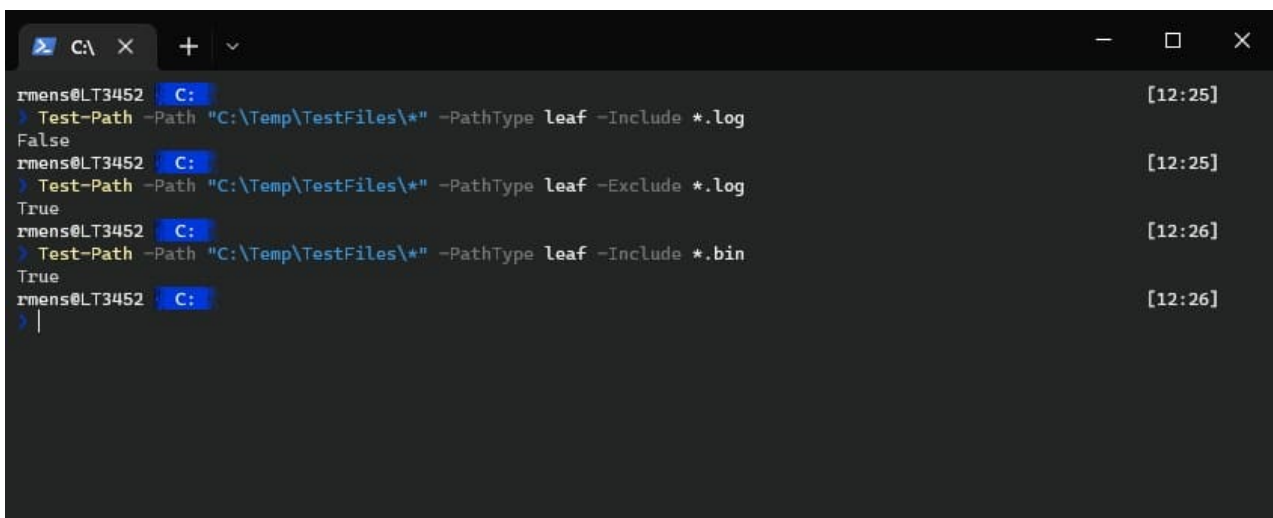
If you want to test if a folder has particular files or not, then you can use the `-include` and `-exclude` parameters. This way we can check if a folder contains a specified file type or any file but the specified file type.

For example, we want to check if there are log files in the following path:

```
# Check if the path contains .log files
Test-Path -Path "C:\Temp\TestFiles\*" -PathType leaf -Include *.log
# Result
False
```

If we want to test if there are any files except the `.log` file then we can use the `exclude` for example:

```
# Check if the path contains .log files
Test-Path -Path "C:\Temp\TestFiles\*" -PathType leaf -Exclude *.log
# Result
True
```

A screenshot of a PowerShell terminal window with a dark background. The window title bar shows 'C:\' and standard window controls. The terminal content shows a series of commands and their outputs. The first command is 'Test-Path -Path "C:\Temp\TestFiles*" -PathType leaf -Include *.log', which returns 'False'. The second command is 'Test-Path -Path "C:\Temp\TestFiles*" -PathType leaf -Exclude *.log', which returns 'True'. The third command is 'Test-Path -Path "C:\Temp\TestFiles*" -PathType leaf -Include *.bin', which returns 'True'. Each command is preceded by a prompt 'rmens@LT3452 C:'. On the right side of the terminal, there are timestamps: '[12:25]' for the first two commands and '[12:26]' for the third. The cursor is at the end of the third command's output.

```
rmens@LT3452 C: [12:25]
> Test-Path -Path "C:\Temp\TestFiles\*" -PathType leaf -Include *.log
False
rmens@LT3452 C: [12:25]
> Test-Path -Path "C:\Temp\TestFiles\*" -PathType leaf -Exclude *.log
True
rmens@LT3452 C: [12:26]
> Test-Path -Path "C:\Temp\TestFiles\*" -PathType leaf -Include *.bin
True
rmens@LT3452 C: [12:26]
> |
```

Test if a path is valid

When you are creating a path dynamically in your script, then it can be a good idea to test if the generated path is a valid path or not. For this, we can use the parameter `-IsValid`. This won't check if the path exists or not, only if it's a valid path.

For example, we can check if all the locations in the `PATH` system variable are valid paths or not:

```
$env:path -split ";" | Test-Path -IsValid
Or maybe a bit easier example is:
```

```
$storage = "c:\temp\testfiles\  
$newFolder = "\LazyFolder"  
$path = $newFolder + $storage # yes, the wrong way around  
Test-Path -path $path -IsValid  
# Result  
False
```

Check if a File or Folder is Newer or Older than a Date

Besides testing if a path exists or not, we can also check if a file or folder is newer or older than a specified date. This is particularly useful if you want to know if a file has been updated or not.

The parameters **-NewerThan** and **-OlderThan** requires a date or datetime object. This can be a datetime string or datetime object. So to test if the folder "testfiles" is newer than the first of Feb, we can do the following:

```
Test-Path -Path "C:\Temp\TestFiles\" -NewerThan "01 feb 2023"  
# Result  
True
```

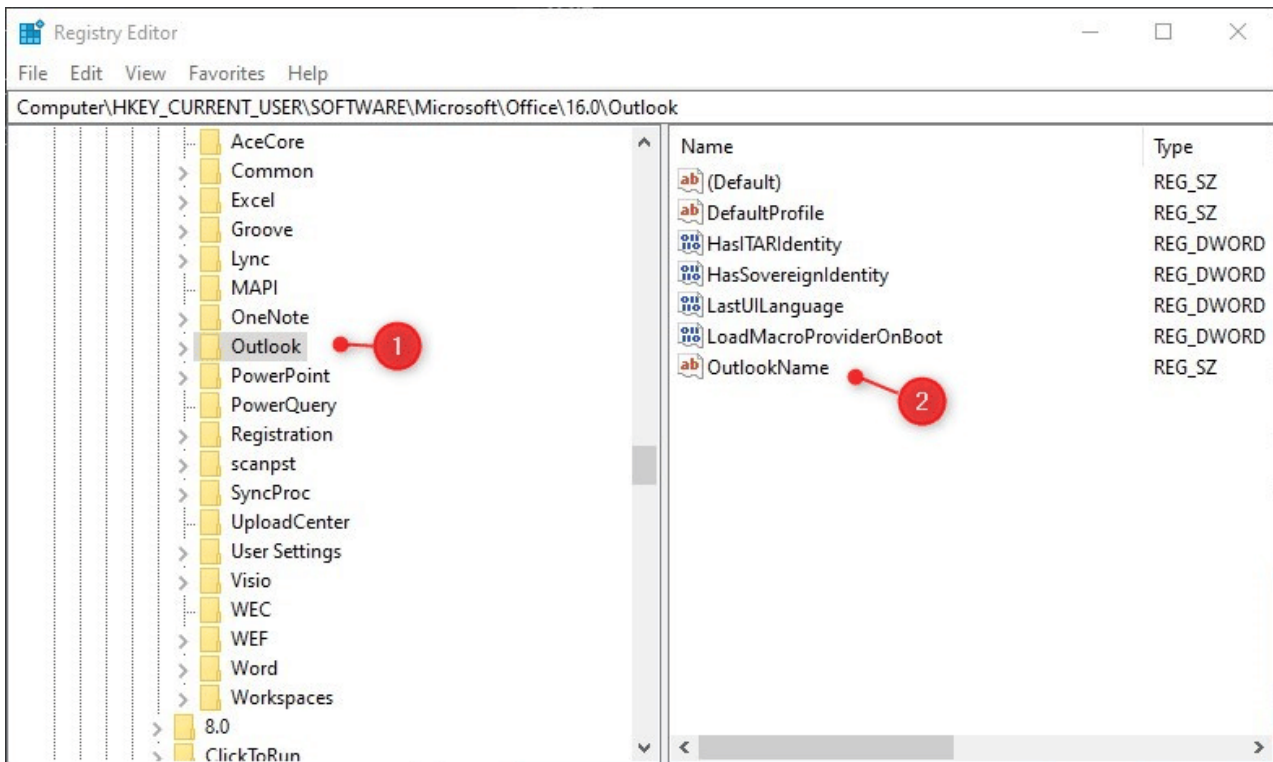
But we can also check if the folder is older than 3 days for example. For this we first create a date object in PowerShell, which we can then use in the Test-Path cmdlet:

```
# Get the date from 3 days ago  
$date = (Get-Date).AddDays(-3)  
# Test if the folder is newer  
Test-Path -Path "C:\Temp\TestFiles\" -NewerThan $date  
You can do the same with OlderThan of course.
```

Test Registry Paths with PowerShell

The Test-Path cmdlet in PowerShell can also be used to check if registry paths exist or not. Now there is one important thing to note here, it can't test the registry entries.

If you look at the screenshot below, we can test if the key Outlook (1) exists or not. But we can't test if the entry "OutlookName" (2) exists.



So to test the registry key, we only need to specify the path of the key like this:

```
Test-Path -Path "HKCU:\Software\Microsoft\Office\16.0\Outlook"
```

As you can see in the results below, the path to key Outlook exists, but it can't find the path to the OutlookName entry:

```
C:\> Test-Path -Path "HKCU:\Software\Microsoft\Office\16.0\Outlook"
True
C:\> Test-Path -Path "HKCU:\Software\Microsoft\Office\16.0\Outlook\OutlookName"
False
```

Wrapping Up

Testing or verifying if resources exist or not is important in any programming or scripting language. It helps you prevent errors or unsuspected behavior in your scripts and allows you to do proper error handling.

The Test-Path cmdlet in PowerShell can do more than most think, but in all cases, it only outputs true or false.

I hope you found this article useful, if you have any questions, then just drop a comment below.

Did you **Liked** this **Article**?

Get the latest articles like this **in your mailbox**
or share this article

I hate spam to, so you can unsubscribe at any time.