

# How to Get File Version in PowerShell

We can use PowerShell to get the file version of any file on your computer. Getting the version information can be useful to check if a software update or deployment was successful, or to inventory the software version on different machines.

In this article, I will show you how to quickly get the file version info, check if the version is higher or lower than a specified version number, and create a custom function for your in PowerShell Profile to make it even more easier.

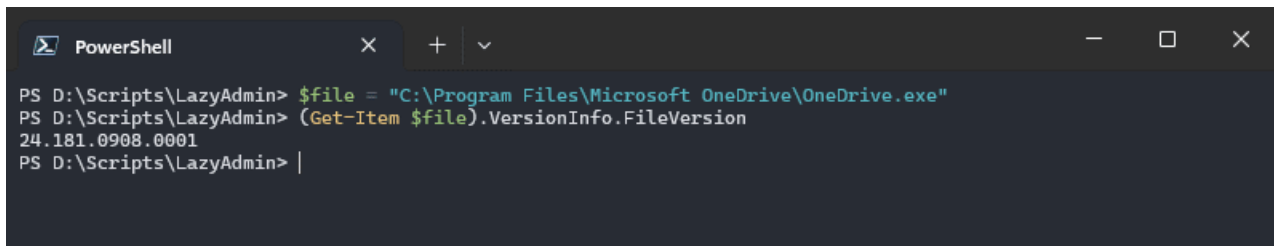
## PowerShell Get File Version

Let's take a look at a couple of methods to get the file version in PowerShell. Most methods are pretty much the same, so just use one which is most convenient for you. We are going to use the OneDrive executable as an example for this.

### Using Get-Item

The most recommended way to get the version information is to use the `Get-Item` cmdlet. The `FileVersion` property is part of the `VersionInfo` property. By placing the `Get-Item` cmdlet between parentheses, we can easily get the complete version number:

```
# Set the file path (will be using this in all examples)
$file = "C:\Program Files\Microsoft OneDrive\OneDrive.exe"
# Get version information with Get-Item
(Get-Item $file).VersionInfo.FileVersion
```



```
PowerShell
PS D:\Scripts\LazyAdmin> $file = "C:\Program Files\Microsoft OneDrive\OneDrive.exe"
PS D:\Scripts\LazyAdmin> (Get-Item $file).VersionInfo.FileVersion
24.181.0908.0001
PS D:\Scripts\LazyAdmin> |
```

Instead of the complete file version number, we can also get the raw version number. This will split the number up into the major, minor, build, and revision numbers.

This is especially useful when you only need to check the major or minor build number of an application.

```
(Get-Item $file).VersionInfo.FileVersionRaw
```

```
# Result
```

```
Major Minor Build Revision
```

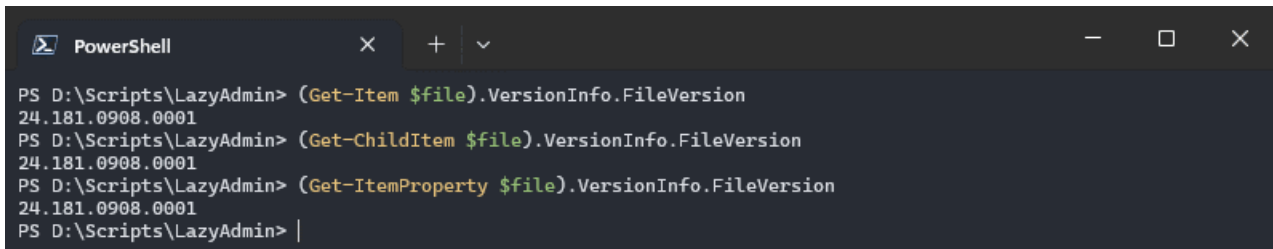
```
-----
```

## Using Get-ChildItem or Get-ItemProperty

---

Instead of the Get-Item cmdlet, we can also use the `Get-ChildItem` or the `Get-ItemProperty` cmdlet. As mentioned, all 3 methods are pretty much the same:

```
# Using Get-ChildItem
(Get-ChildItem $file).VersionInfo.FileVersion
# Using Get-ItemProperty
(Get-ItemProperty $file).VersionInfo.FileVersion
```



```
PowerShell
PS D:\Scripts\LazyAdmin> (Get-Item $file).VersionInfo.FileVersion
24.181.0908.0001
PS D:\Scripts\LazyAdmin> (Get-ChildItem $file).VersionInfo.FileVersion
24.181.0908.0001
PS D:\Scripts\LazyAdmin> (Get-ItemProperty $file).VersionInfo.FileVersion
24.181.0908.0001
PS D:\Scripts\LazyAdmin> |
```

I have run some tests to check if one method is faster, and as expected, the difference between the three is minimal after running 50 tests:

- Average time for Get-Item: 1.078332 ms
- Average time for Get-ChildItem: 1.108278 ms
- Average time for Get-ItemProperty: 1.02891 ms

## Using System.Diagnostics.FileVersionInfo .NET Class

---

Another method to get the file version info is by using the `System.Diagnostics.FileVersionInfo` .NET Class. The advantage of using the .NET class directly is that it's faster than the other three methods.

```
([System.Diagnostics.FileVersionInfo]::GetVersionInfo($file)).FileVersion
```

The difference is not really big, but when you need to get the version information for a lot of files or check it on many machines, then this might save you some time. With the .NET class, it took on average only 0.79294 ms to get the version information, making it around 20% faster than the other methods.

## Custom Function Get-FileVersion

---

If you need to check the file version on a regular basis, then it might be worth creating a custom function and storing it in your [PowerShell profile](#). This way, you can easily retrieve the version information with an easy-to-remember command, instead of using the `Get-Item` cmdlet with all the correct properties.

You can copy and paste the function below to your PowerShell profile:

```
function Get-FileVersion {
```

```

param (
[string]$FilePath,
[switch]$raw = $false
)
if (-not (Test-Path $FilePath)) {
Write-Error "File not found: $FilePath"
return
}
try {
$versionInfo = [System.Diagnostics.FileVersionInfo]::GetVersionInfo($FilePath)
if (-not($raw)) {
return $versionInfo.FileVersion
}else{
return $versionInfo.FileVersionRaw
}
} catch {
Write-Error "Unable to get version info for: $FilePath"
}
}

```

This way you can just use the following cmdlet in your PowerShell sessions:

```
Get-FileVersion "C:\Program Files\Microsoft OneDrive\OneDrive.exe"
```

By default, it will return the FileVersion number for the given file. If you add the parameter -raw, then you will get the version number split up into major, minor, build, and revision.

## Comparing File Versions

---

When you need to compare file versions, it's important to cast the version number into a [version] type. Otherwise, will PowerShell do a string comparison, which is not that accurate in combination with version numbers.

The [Version] type is created to compare versions that contain multiple numerical components, like the major, minor, build, and revision number. This way PowerShell knows how to handle comparisons between version numbers that are otherwise not possible with regular string comparisons.

```

# Get the file version and cast it to a [Version] object
$version = [Version]((Get-Item $file).VersionInfo.FileVersion)
# Define the target version for comparison, also cast to [Version]
$targetVersion = [Version]"24.180.708.1"
# Compare the versions
if ($version -gt $targetVersion) {
Write-Host "The file version is newer than 24.180.708.1"
} else {
Write-Host "The file version is older or equal to 24.180.708.1"
}

```

## Wrapping Up

---

There are a couple of ways to get the File Version information with PowerShell. However, the normal cmdlets are pretty much the same, when it comes to usability and performance.

If you need to retrieve the information from a lot of files, then it makes sense to the faster .NET class. Also, make sure that you try out the custom function in your PowerShell profile.

Hope you liked this article, if you have any questions, just drop a comment below.

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

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

Не удается связаться с сервисом reCAPTCHA. Проверьте подключение к Интернету и перезагрузите страницу.

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