ITPro Today™

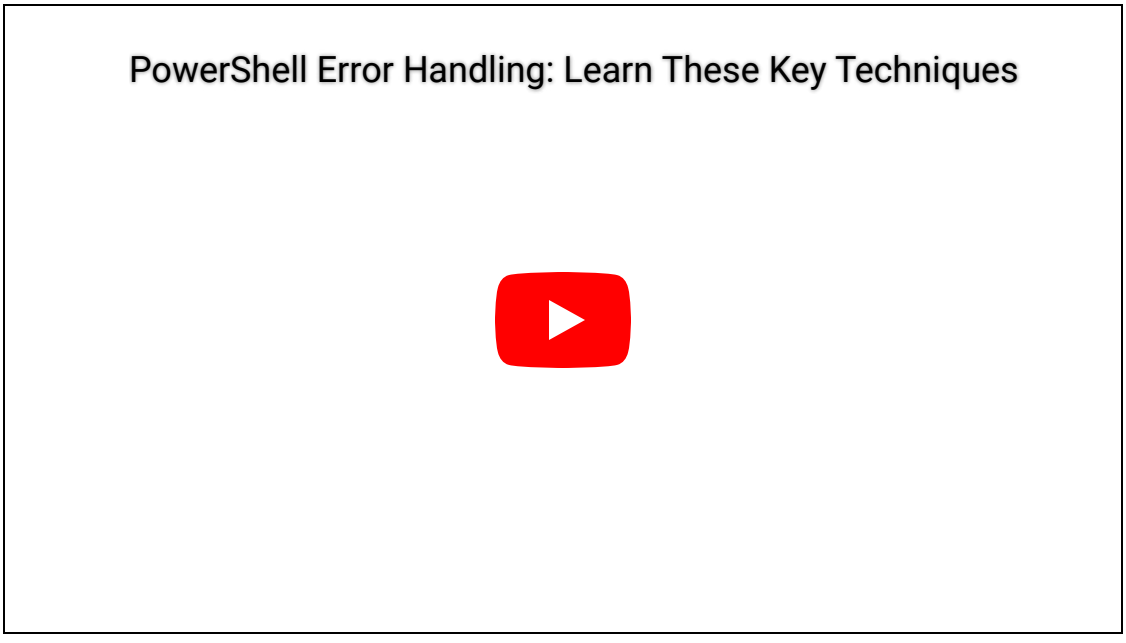
END-USER PLATFORMS > POWERSHELL

PowerShell Error Handling: Learn These Key Techniques

Join PowerShell expert Brien Posey as he explains the PowerShell error management. This video tutorial details essential error-handling techniques.

Brien Posey | Apr 03, 2024

In this tutorial, PowerShell expert Brien Posey guides viewers through essential techniques for managing errors within PowerShell scripts. Through practical demonstrations, Posey explores methods such as suppressing non-fatal errors, setting up Try and Catch blocks, and using warning messages to streamline script execution.



The transcript has been lightly edited for length and clarity.

Transcript:

Related: 4 Cmdlets To Display Text on the PowerShell Console Screen

Brien Posey: Hello, greetings and welcome. I'm Brien Posey. In this video, I want to show you some techniques that are commonly used for handling errors in PowerShell.

If you look on the right side of the screen, you can see a really simple script that I've created:

```
c:
cd\
cd scripts
New-Item -Name “Temp” -ItemType “Directory”
Write-Host ‘Do some other stuff.’
```

The first line says *c:* – this simply changes to the C: drive if you're not already there. Then we have *cd* – this drops the command prompt down to the root directory. Then we have cd scripts – this opens the Scripts folder. Then we have a line that says *New-Item -Name “Temp” -ItemType “Directory.”* What this line does is create a new folder called Temp. Then the very last line we have is *Write-Host ‘Do some other stuff.’* This line is just a generic representation of any other instructions you would normally execute. The point is that we have a line of code that executes after we create the Directory.

So, let's go ahead and execute the script. I'll go ahead and press Enter, and we can see that the new folder has been created.

Now I'm going to run the script again. This time I get an error message because the folder already exists.

Let's look at what's going on in the first three steps in the script. The c:, cd\, and cd scripts commands are silent. They don't produce any sort of visible output, so we don't have anything showing up in our PowerShell window.

Then when we execute the New-Item cmdlet, that's where we get the error. But notice right here, PowerShell went ahead and displayed, 'Do some other stuff.' So, what happened is that even though we got an error, this wasn't a fatal error. PowerShell, by default, when it encounters a nonfatal error, will keep running. In this particular case, the folder already existed, so it threw an error message but then went on running, doing exactly what it was supposed to do.

If this were a real-world script that was trying to create a folder and then do some other stuff, it would be fine. It would absolutely do what it needs to do. The problem with this is that it's ugly. If you gave this to someone unsuspecting, and they ran the script and got an error message, they would see all the red text on the screen, and they probably wouldn't know what to do with it.

-ErrorAction SilentlyContinue

Let's look at some ways that we might suppress this error. One of the things that we can do is to add an -ErrorAction to the New-Item cmdlets. The way that we do that is by going to the end of the New-Item line and simply typing:

```
New-Item -Name "Temp" -ItemType "Directory" -ErrorAction SilentlyContinue
```

What this does is tell PowerShell that if this line causes an error, then just ignore it. Don't even display the error. Just go about doing whatever you're doing so long as it's a non-fatal error.

Let me go ahead and save this file. I'll rerun the script, and you can see this time the error has been suppressed. All we see is, 'Do some other stuff.' So, we still encountered an error, but the difference is that PowerShell isn't showing the error.

That's one way that we can handle errors in PowerShell.

Try & Catch

The problem is that we don't necessarily know for sure that we're going to encounter a nonfatal error in our scripts. Sometimes errors can be fatal. If you suspect that a fatal error might occur and cause a script to stop running, then you don't want to suppress that. A better idea is to use Try and Catch.

Let's look at how that works. Okay, so I paused the video for a moment, and I added a few lines of code to my script:

```
c:
cd\
cd scripts

Try {
New-Item -Name “Temp” -ItemType “Directory”
}
Catch {
Write-Host “The folder already exists”}

Write-Host ‘Do some other stuff.’
```

The first three lines are exactly what we had before. You'll notice that we still have the New-Item cmdlet just as we did before, but now the New-Item cmdlet is enclosed in brackets. The reason why we can enclose this cmdlet in brackets is because we're using the Try command. What the Try command does is execute whatever command is in brackets to see if it produces an error or not. If it does produce an error, then we use the Catch command to handle the error. So, in this case, the Catch command is going to execute the line, Write-Host “The folder already exists”.

So, rather than displaying a PowerShell-style error message, what we're going to get instead is a line of text saying the folder already exists.

Let's go ahead and run the script to see what it does.

When I run the script, you see that we get a big red error message just like we did before we did anything. Let's look at what's going on here. The error message that we get is:

```
New-Item : An item with the specified name C:\scripts\Temp already exists.
```

This is the same error message that we got earlier. It would appear the Try and Catch aren't doing anything. Remember, the Catch statement is designed so that if an error does occur, we're supposed to see a line of text saying the folder already exists. Clearly, that's not happening.

So, why is that? Well, the reason has to do with something that I talked about earlier. I mentioned that PowerShell's default behavior is that if it encounters a nonfatal error, then it will go ahead and progress the rest of the way through the script. That's exactly what's happening here. We're encountering an error, that error is nonfatal, and so the rest of the script is executing. You can see, 'Do some other stuff.'

But remember the -ErrorAction that I talked about earlier. When I had the New-Item cdmlet, I suppressed the error, which was a nonfatal error, by typing -ErrorAction SilentlyContinue.

If I go ahead, save this, and then run the script again, the error goes away. But I don't get that message saying that the folder already exists. So, why is that? Well, it has to do with the difference between fatal and nonfatal errors. This is a nonfatal error, which means that Try and Catch isn't going to work. Try and Catch are only designed for situations in which a fatal error occurs.

Now, in a situation like this, where we're trying to create a folder but that folder already exists, we're never going to encounter a fatal error. So, what can we do if we want to use Try and Catch? Well, the way that we would fix this problem is by changing the -ErrorAction. Instead of using SilentlyContinue, I'm going to use Stop.

```
Try {
New-Item -Name "Temp" -ItemType "Directory" -ErrorAction Stop
```

```
}
Catch {
Write-Host “The folder already exists”}
```

What Stop does is tell PowerShell to treat this error as though it were a fatal error even though it's not.

Let me go ahead and save this. Let's run the script one more time.

This time I get the error message that is outlined by the Catch statement: “The folder already exists”. So instead of seeing the big, red, glaring error that PowerShell would normally generate, I see a nice line of formatted text. Then you'll notice that PowerShell does continue after that. It displays the line that says, ‘Do some other stuff.’

Write-Warning

Now there's one more thing that we might do in a situation like this just to make the script a little bit neater. Maybe what we want to do is instead of just displaying a line that says, “The folder already exists”, is to display that as a warning message. That way the text will be highlighted. And the way that we would do that is really easy to do, we would simply change Write-Host to Write-Warning.

```
Catch {
Write-Warning “The folder already exists”}
```

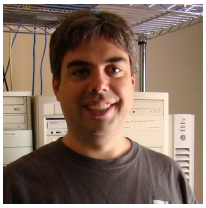
I'll save my file. Let's go ahead and run the script. You can see we have a warning message that's displayed, PowerShell generates a warning that the folder already exists.

```
WARNING: The folder already exists
```

Then, of course, once that warning message is displayed, PowerShell goes on and executes the last line of the script, which is Write-Host ‘Do some other stuff.’

So, those are just a few techniques that you can use to suppress errors within PowerShell scripts. I'm Brien Posey. Thanks for watching.

About the author



Brien Posey is a bestselling technology author, speaker, and 21x Microsoft MVP. In addition to his ongoing work in IT, Posey has trained as a commercial astronaut candidate in preparation to fly on a mission to study polar mesospheric clouds from space.