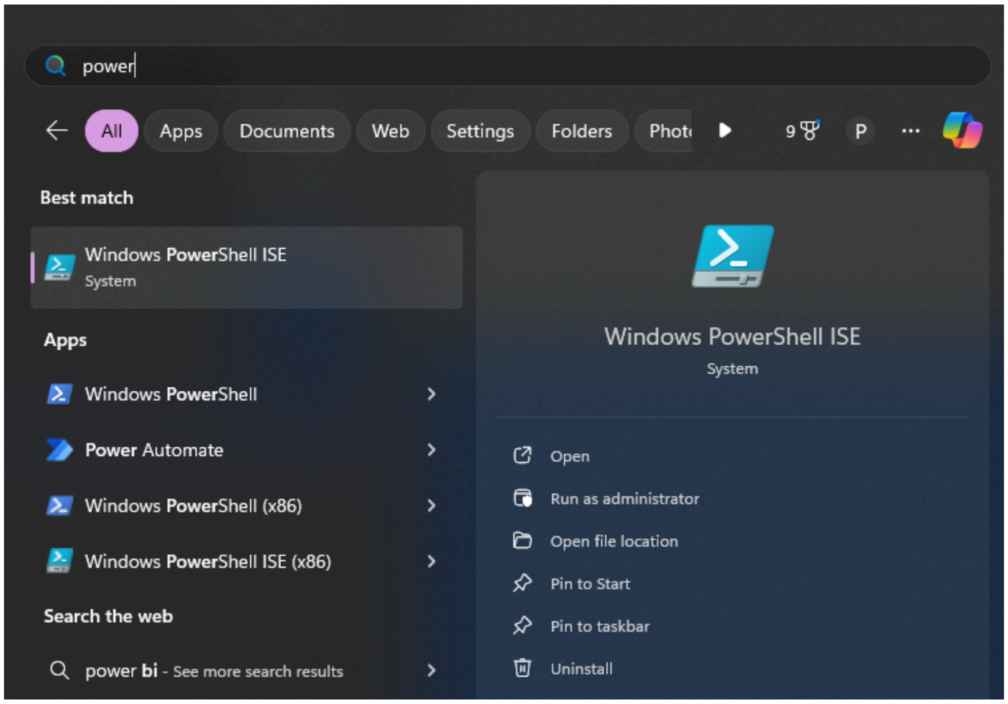
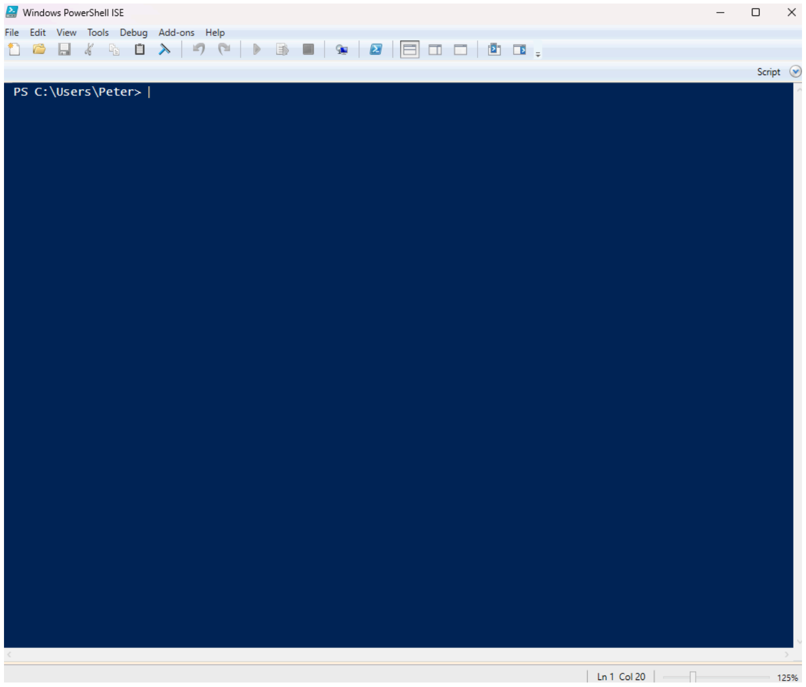
# File Integrity Monitor Documentation

Start PowerShell ISE ***"Run as administrator"***.

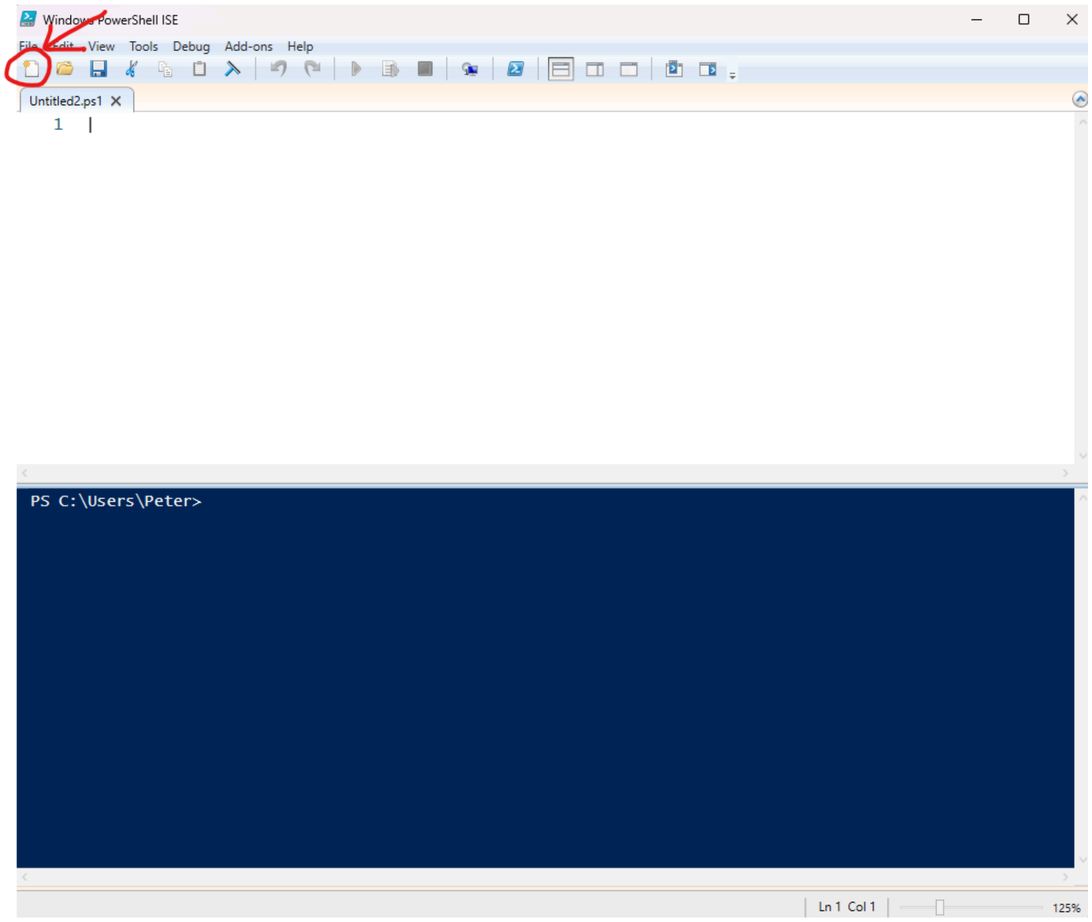


PowerShell ISE application opens as shown below.



```
PS C:\Users\Peter> |
```

Create a ***"New File"*** to create the File Integrity Monitoring scripts for this project.

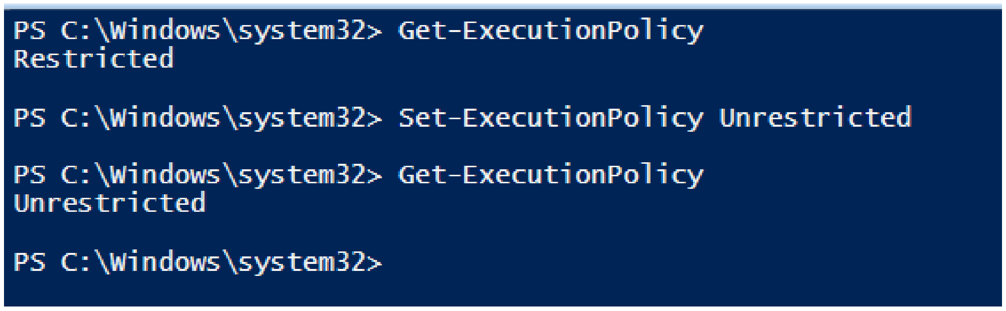


```
PS C:\Users\Peter>
```

Before we get started we must check and allow scripts to be run on this computer. To do so follow the following commands:

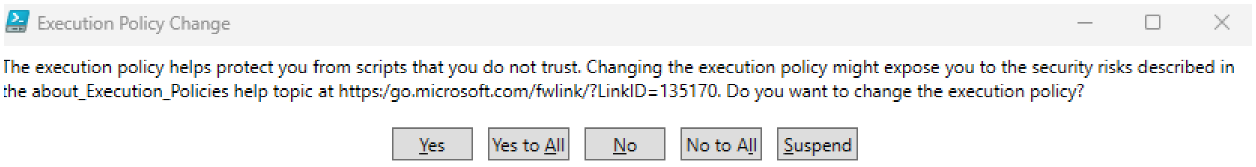
`Get-ExecutionPolicy` - Shows the current policy that is active.

`Set-ExecutionPolicy` - Changes the policy to the new selected one.



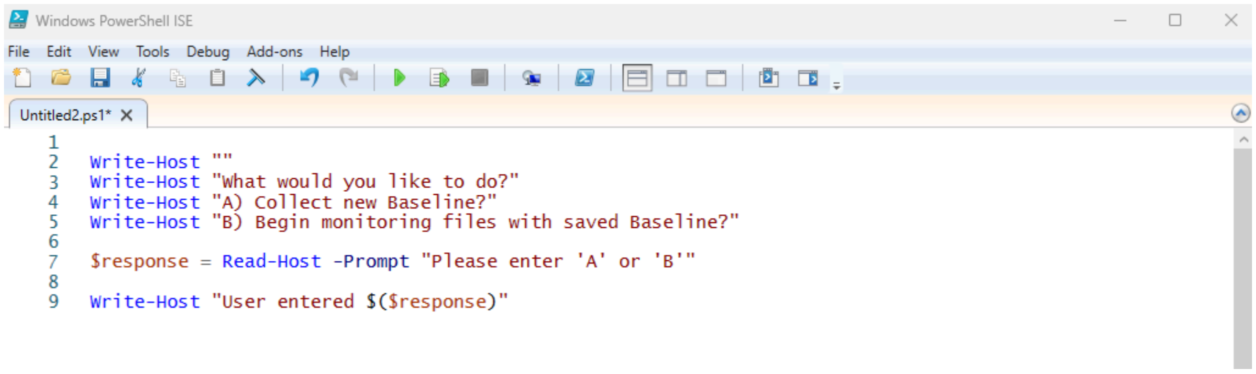
```
PS C:\Windows\system32> Get-ExecutionPolicy
Restricted

PS C:\Windows\system32> Set-ExecutionPolicy Unrestricted

PS C:\Windows\system32> Get-ExecutionPolicy
Unrestricted

PS C:\Windows\system32>
```

The following prompt will show after changing the execution policy hit ***"Yes to All"*** to allow this but **REMEMBER** to Restrict the execution policy after everything is completed.



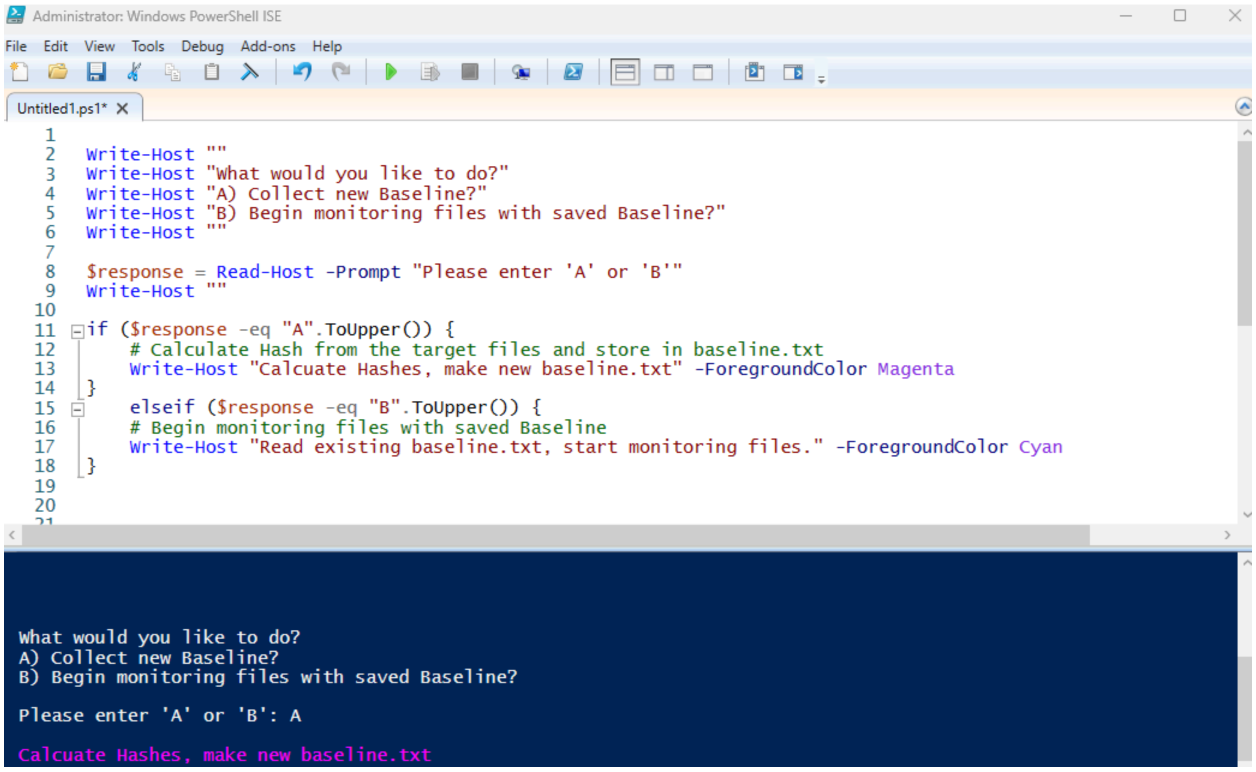
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose you to the security risks described in the about_Execution_Policies help topic at https:/go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?

`Write-Host '____'` outputs text to the console for the user to see. The variable `$response` is intended to store data or the result of a command, and in this case, it will capture the user's choice between A or B.



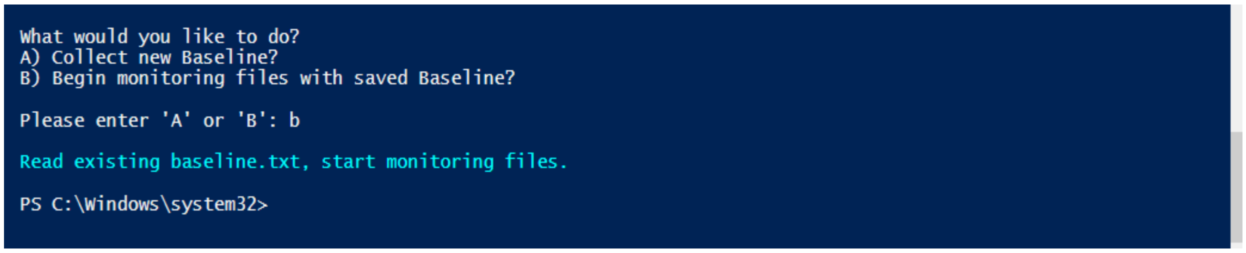
```powershell

Write-Host ""
Write-Host "What would you like to do?"
Write-Host "A) Collect new Baseline?"
Write-Host "B) Begin monitoring files with saved Baseline?"

$response = Read-Host -Prompt "Please enter 'A' or 'B'"

Write-Host "User entered $($response)"
```

Implement the `$response` for option **A** or **B**. If the response is '**A**', the script will execute the corresponding selection. The `.ToUpper()` method allows the script to recognize '**a**' as a valid input as well. Similarly, it will handle '**B**' or '**b**' in the same way. As illustrated in the blue box below, I tested the script to verify the function.


Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1*

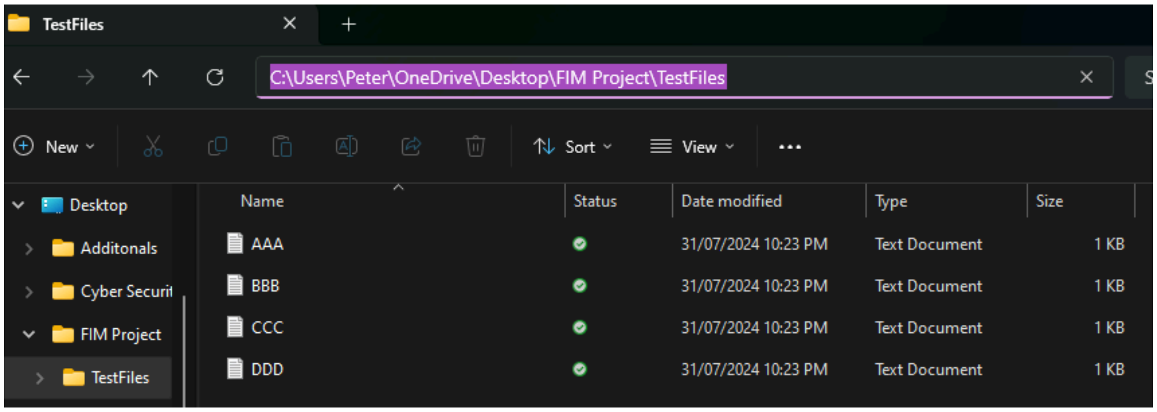
```powershell

Write-Host ""
Write-Host "What would you like to do?"
Write-Host "A) Collect new Baseline?"
Write-Host "B) Begin monitoring files with saved Baseline?"
Write-Host ""

$response = Read-Host -Prompt "Please enter 'A' or 'B'"
Write-Host ""

if ($response -eq "A".ToUpper()) {
    # Calculate Hash from the target files and store in baseline.txt
    Write-Host "Calcuate Hashes, make new baseline.txt" -ForegroundColor Magenta
}
    elseif ($response -eq "B".ToUpper()) {
    # Begin monitoring files with saved Baseline
    Write-Host "Read existing baseline.txt, start monitoring files." -ForegroundColor Cyan
}
```

```
What would you like to do?
A) Collect new Baseline?
B) Begin monitoring files with saved Baseline?

Please enter 'A' or 'B': A

Calcuate Hashes, make new baseline.txt
```



```
What would you like to do?
A) Collect new Baseline?
B) Begin monitoring files with saved Baseline?

Please enter 'A' or 'B': b

Read existing baseline.txt, start monitoring files.

PS C:\Windows\system32>
```

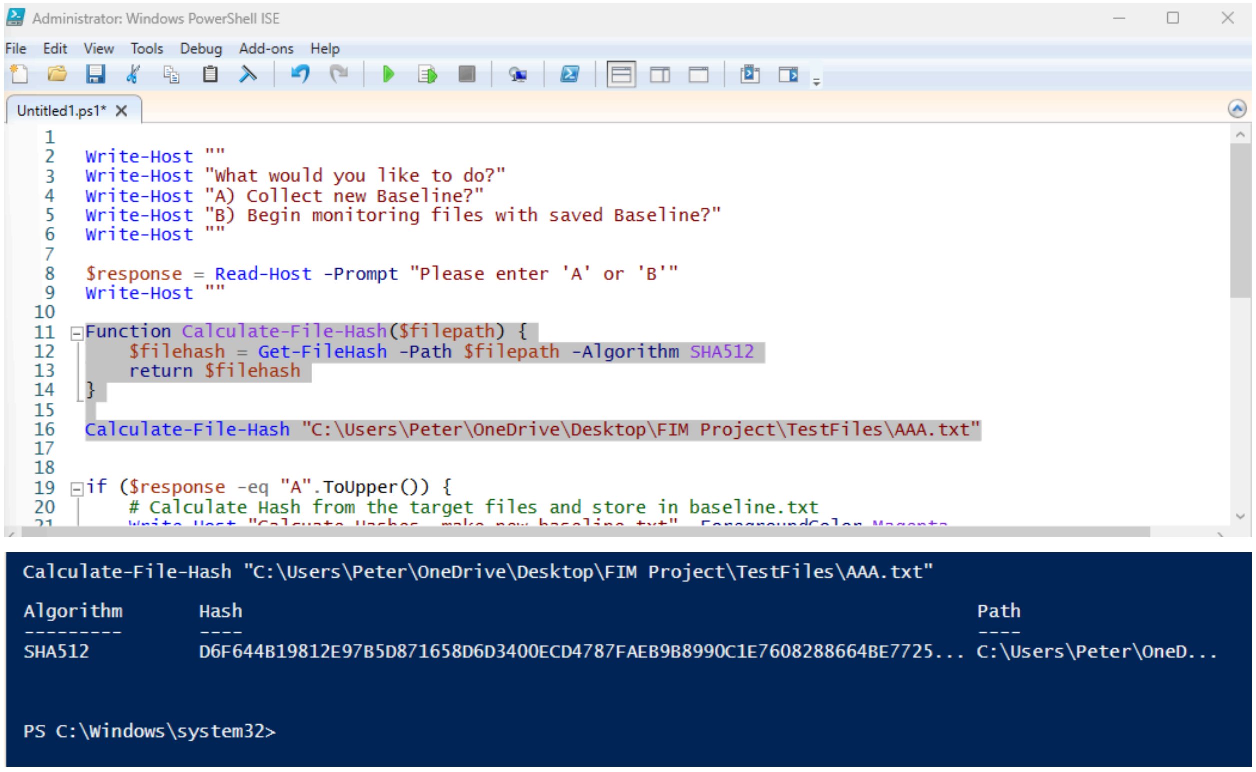
Generate a few text files as samples for monitoring purposes. I've created four files, each containing different content. Be sure to record the directory path for the scripts to access and monitor the files.



C:\Users\Peter\OneDrive\Desktop\FIM Project\TestFiles

| Name | Status | Date modified | Type | Size |
|---|---|---|---|---|
| AAA | ✓ | 31/07/2024 10:23 PM | Text Document | 1 KB |
| BBB | ✓ | 31/07/2024 10:23 PM | Text Document | 1 KB |
| CCC | ✓ | 31/07/2024 10:23 PM | Text Document | 1 KB |
| DDD | ✓ | 31/07/2024 10:23 PM | Text Document | 1 KB |

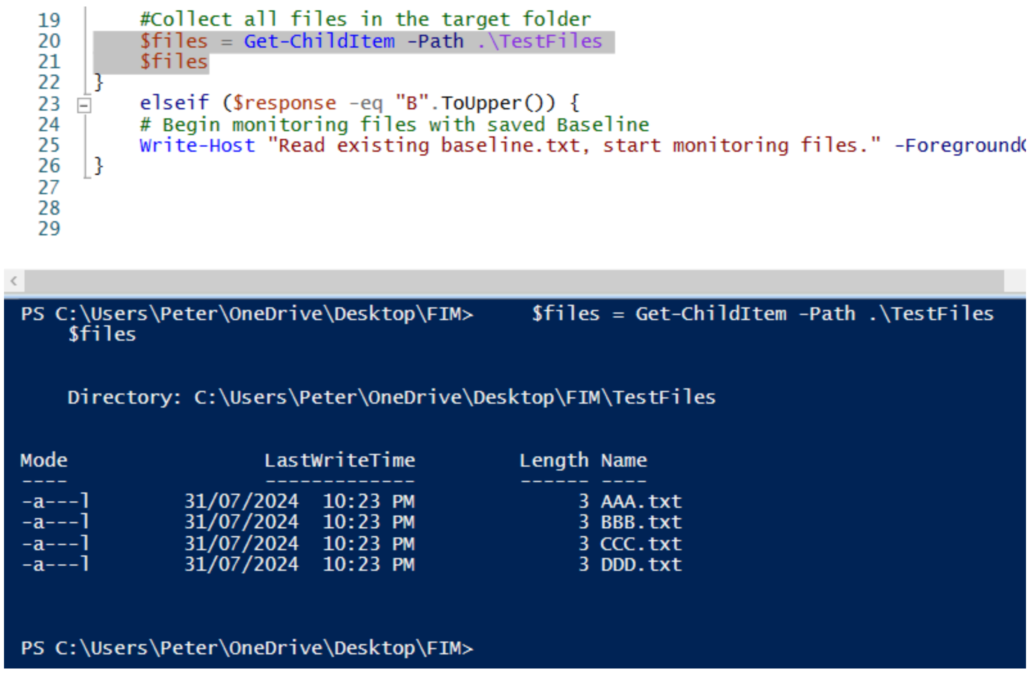
Back in PowerShell ISE, you need to define a function to calculate file hashes.
`Function Calculate-File-Hash($filepath)`: Defines a function named `Calculate-File-Hash` that accepts a single parameter, `$filepath`. This parameter should be the path to the file for which the hash is to be computed.
`Return $filehash`: Returns the value stored in `$filehash`, which is the hash object containing the SHA-512 hash of the file.
*`Calculate-File-Hash "DIRECTORY PATH"`: Calls the `Calculate-File-Hash` function with a specified directory path. This command computes and displays the hashes for files within the provided directory. Note that this line of script will be removed after testing.*


Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1*

```powershell

Write-Host ""
Write-Host "What would you like to do?"
Write-Host "A) Collect new Baseline?"
Write-Host "B) Begin monitoring files with saved Baseline?"
Write-Host ""

$response = Read-Host -Prompt "Please enter 'A' or 'B'"
Write-Host ""

Function Calculate-File-Hash($filepath) {
    $filehash = Get-FileHash -Path $filepath -Algorithm SHA512
    return $filehash
}

Calculate-File-Hash "C:\Users\Peter\OneDrive\Desktop\FIM Project\TestFiles\AAA.txt"


if ($response -eq "A".ToUpper()) {
    # Calculate Hash from the target files and store in baseline.txt
    Write-Host "Calcuate Hashes, make new baseline.txt" -ForegroundColor Magenta
```

```
Calculate-File-Hash "C:\Users\Peter\OneDrive\Desktop\FIM Project\TestFiles\AAA.txt"

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA512          D6F644B19812E97B5D871658D6D3400ECD4787FAEB9B8990C1E7608288664BE7725... C:\Users\Peter\OneD...


PS C:\Windows\system32>
```

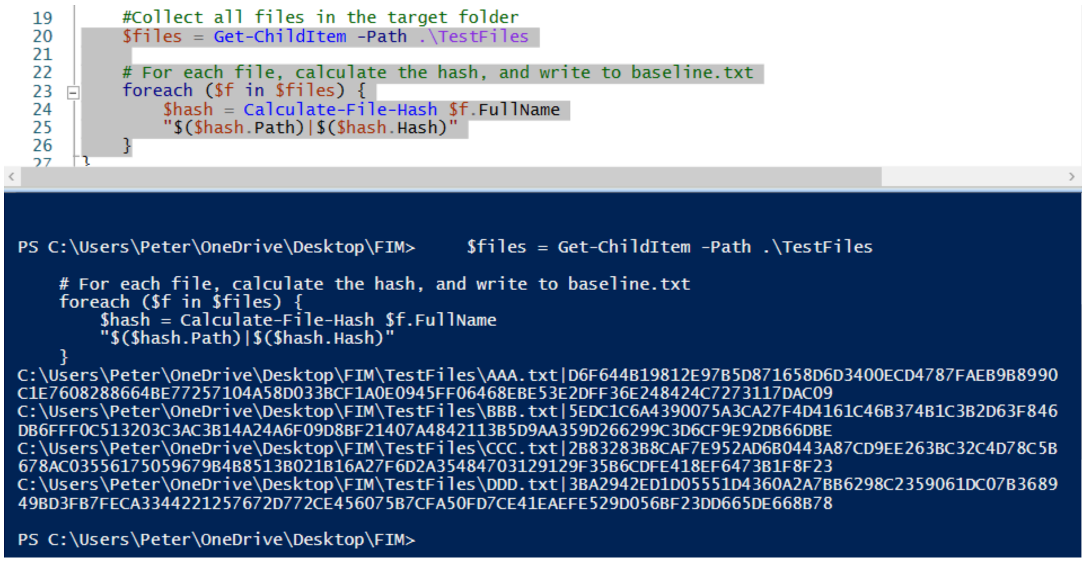
`Get-ChildItem -Path .\TestFiles`: Fetches all items (files and directories) located within the `TestFiles` directory.
`$files`: Holds the output of the `Get-ChildItem` command, which includes the list of files and directories in `TestFiles`.
`$files`: When used by itself, it displays the contents of the `$files` variable, showing the files and directories retrieved.



```
    #Collect all files in the target folder
    $files = Get-ChildItem -Path .\TestFiles
    $files
}
    elseif ($response -eq "B".ToUpper()) {
    # Begin monitoring files with saved Baseline
    Write-Host "Read existing baseline.txt, start monitoring files." -Foreground
}
```

```
PS C:\Users\Peter\OneDrive\Desktop\FIM>     $files = Get-ChildItem -Path .\TestFiles
    $files


    Directory: C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---l        31/07/2024  10:23 PM              3 AAA.txt
-a---l        31/07/2024  10:23 PM              3 BBB.txt
-a---l        31/07/2024  10:23 PM              3 CCC.txt
-a---l        31/07/2024  10:23 PM              3 DDD.txt


PS C:\Users\Peter\OneDrive\Desktop\FIM>
```

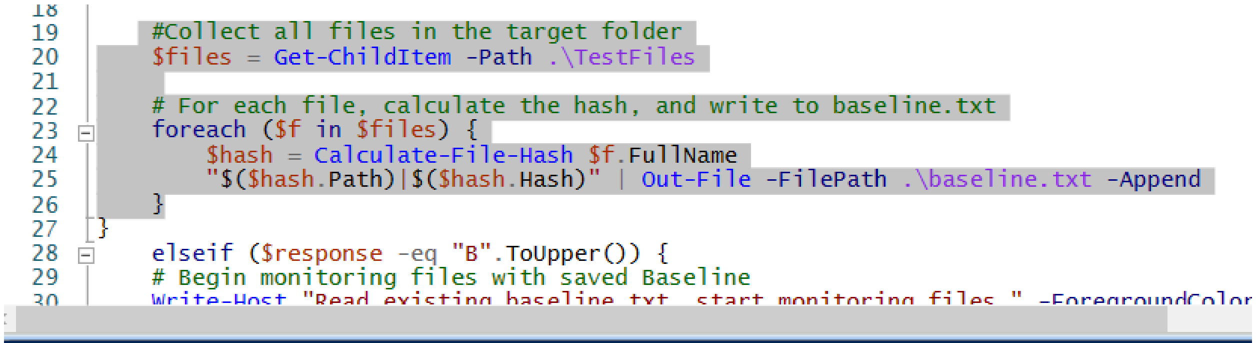
`Foreach ($f in $files)`: Iterates over each item in the `$files` collection, processing each file or folder individually.
`$hash = Calculate-File-Hash $f.FullName`: Computes the hash for the file specified by `$f.FullName` and stores both the directory path and hash value in the `$hash` variable.
`$(hash.Path)|$($hash.Hash)`: Extracts and formats the file's directory path and its SHA-512 hash, presenting them in a `path|hash` format.



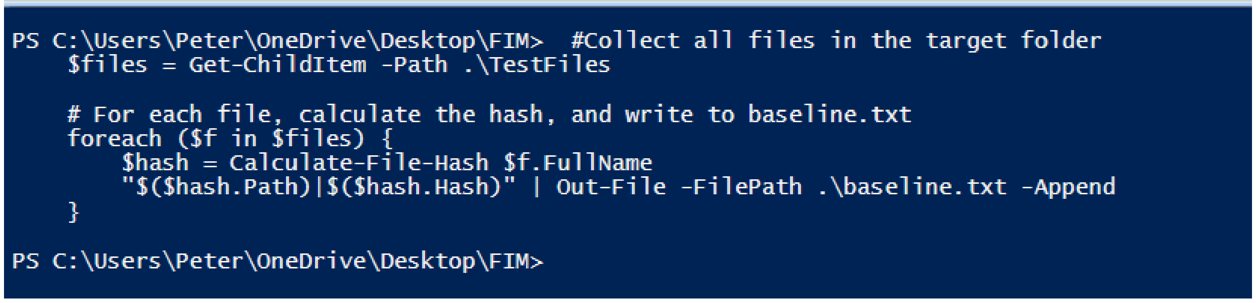
```
    #Collect all files in the target folder
    $files = Get-ChildItem -Path .\TestFiles

    # For each file, calculate the hash, and write to baseline.txt
    foreach ($f in $files) {
        $hash = Calculate-File-Hash $f.FullName
        "$($hash.Path)|$($hash.Hash)"
    }
}
```

```
PS C:\Users\Peter\OneDrive\Desktop\FIM>     $files = Get-ChildItem -Path .\TestFiles

    # For each file, calculate the hash, and write to baseline.txt
    foreach ($f in $files) {
        $hash = Calculate-File-Hash $f.FullName
        "$($hash.Path)|$($hash.Hash)"
    }
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\AAA.txt|D6F644B19812E97B5D871658D6D3400ECD4787FAEB9B8990
C1E7608288664BE77257104A58D033BCF1A0E0945FF06468EBE53E2DFF36E248424C7273117DAC09
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\BBB.txt|5EDC1C6A4390075A3CA27F4D4161C46B374B1C3B2D63F846
DB6FFF0C513203C3AC3B14A24A6F09D8BF21407A4842113B5D9AA359D266299C3D6CF9E92DB66DBE
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\CCC.txt|2B83283B8CAF7E952AD6B0443A87CD9EE263BC32C4D78C5B
678AC03556175059679B4B8513B021B16A27F6D2A35484703129129F35B6CDFE418EF6473B1F8F23
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt|3BA2942ED1D05551D4360A2A7BB6298C2359061DC07B3689
49BD3FB7FECA3344221257672D772CE456075B7CFA50FD7CE41EAEFE529D056BF23DD665DE668B78

PS C:\Users\Peter\OneDrive\Desktop\FIM>
```

By adding the code `| Out-File -FilePath .\baseline.txt -Append`, this command line writes the directory path and the stored hash value to a text file. The `-Append` parameter ensures that the file is updated without duplication.

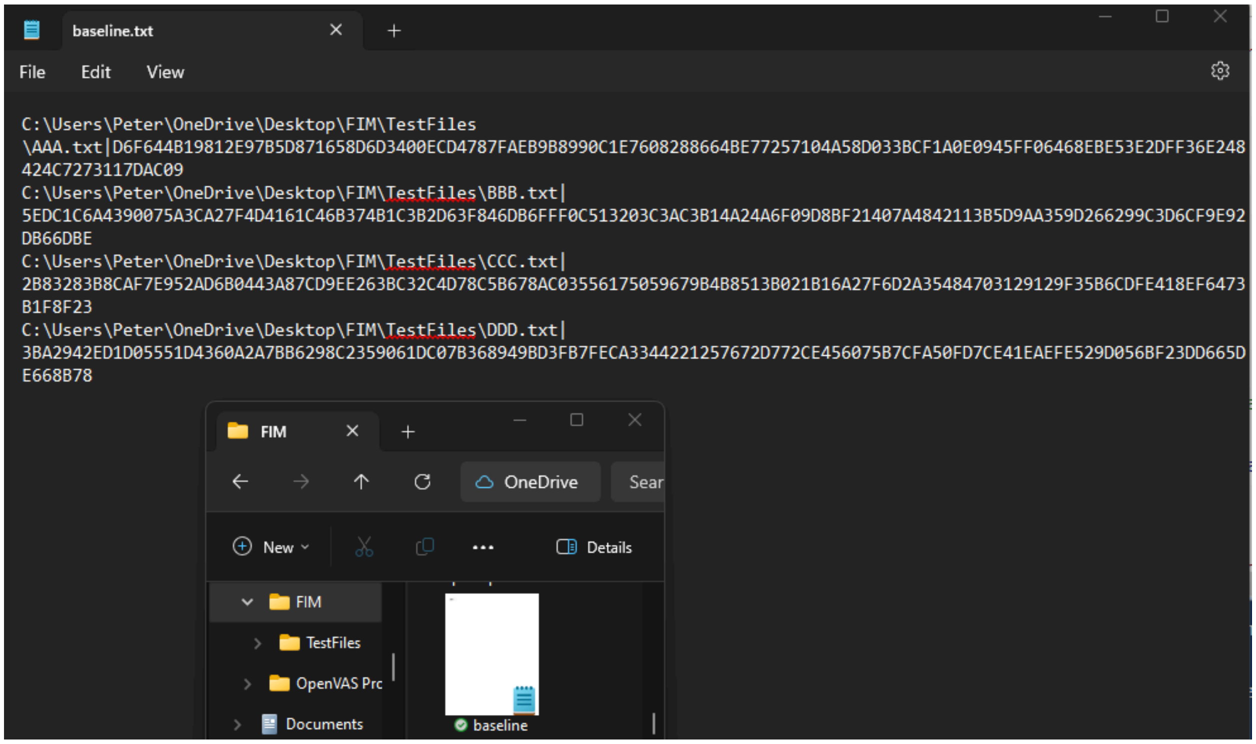


```
    #Collect all files in the target folder
    $files = Get-ChildItem -Path .\TestFiles

    # For each file, calculate the hash, and write to baseline.txt
    foreach ($f in $files) {
        $hash = Calculate-File-Hash $f.FullName
        "$($hash.Path)|$($hash.Hash)" | Out-File -FilePath .\baseline.txt -Append
    }
}

    elseif ($response -eq "B".ToUpper()) {
    # Begin monitoring files with saved Baseline
    Write-Host "Read existing baseline.txt, start monitoring files." -ForegroundColor
```



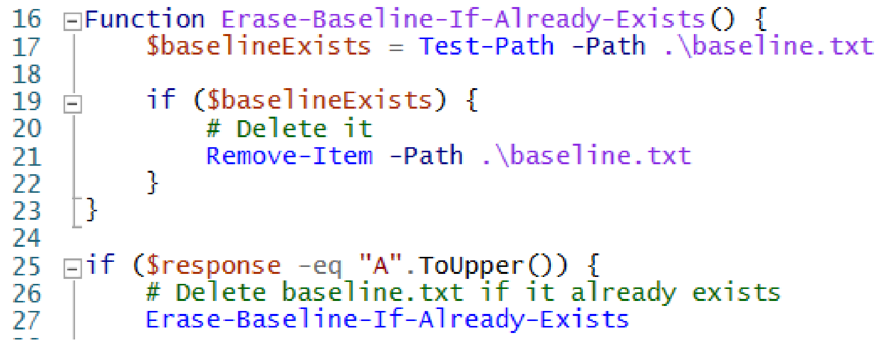
```
PS C:\Users\Peter\OneDrive\Desktop\FIM>  #Collect all files in the target folder
    $files = Get-ChildItem -Path .\TestFiles

    # For each file, calculate the hash, and write to baseline.txt
    foreach ($f in $files) {
        $hash = Calculate-File-Hash $f.FullName
        "$($hash.Path)|$($hash.Hash)" | Out-File -FilePath .\baseline.txt -Append
    }

PS C:\Users\Peter\OneDrive\Desktop\FIM>
```

The output from the command above consists of the following items.



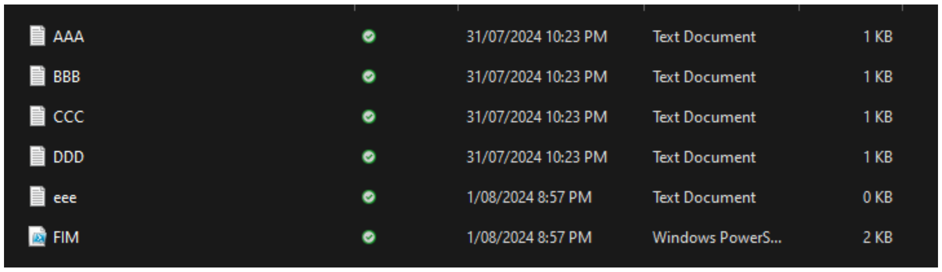
```
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles
\AAA.txt|D6F644B19812E97B5D871658D6D3400ECD4787FAEB9B8990C1E7608288664BE77257104A58D033BCF1A0E0945FF06468EBE53E2DFF36E248
424C7273117DAC09
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\BBB.txt|
5EDC1C6A4390075A3CA27F4D4161C46B374B1C3B2D63F846DB6FFF0C513203C3AC3B14A24A6F09D8BF21407A4842113B5D9AA359D266299C3D6CF9E92
DB66DBE
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\CCC.txt|
2B83283B8CAF7E952AD6B0443A87CD9EE263BC32C4D78C5B678AC03556175059679B4B8513B021B16A27F6D2A35484703129129F35B6CDFE418EF6473
B1F8F23
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt|
3BA2942ED1D05551D4360A2A7BB6298C2359061DC07B368949BD3FB7FECA3344221257672D772CE456075B7CFA50FD7CE41EAEFE529D056BF23DD665D
E668B78
```

Develop a new function for option **A**, called 'Collect New Baseline,' which will erase `baseline.txt` and store a fresh hash value of the selected file.



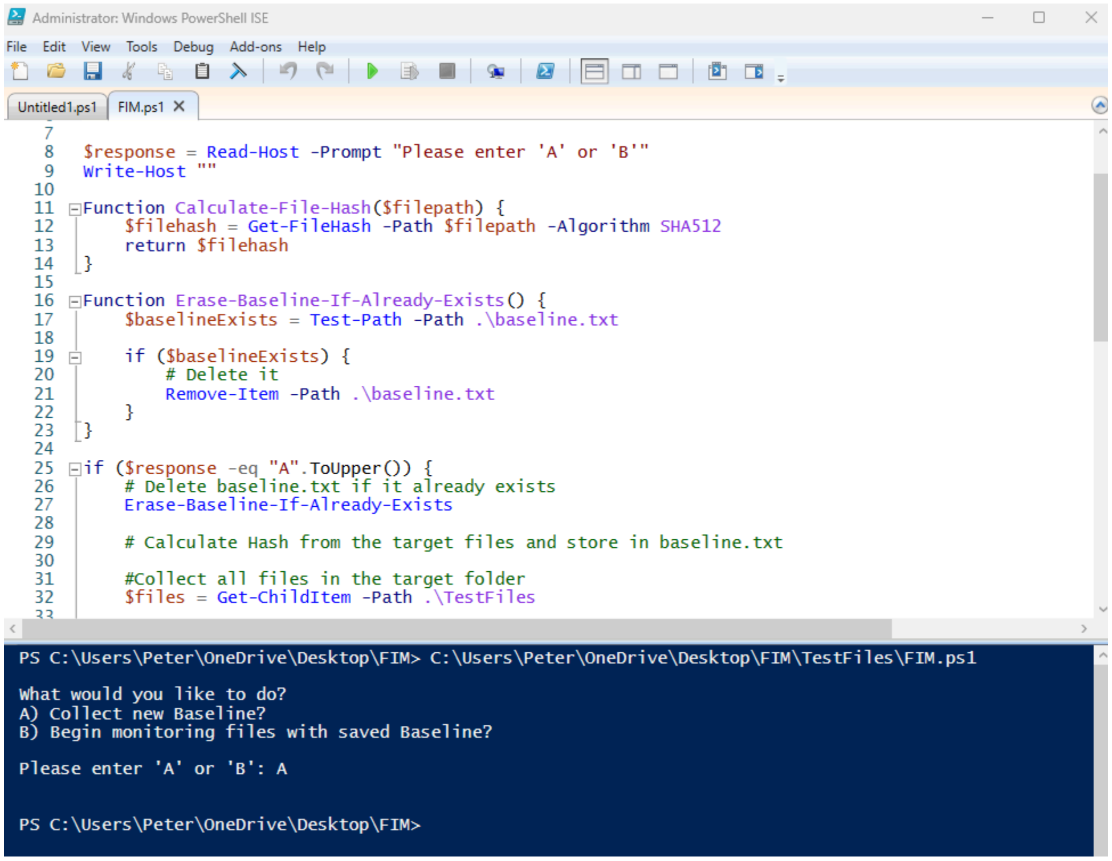
```
Function Erase-Baseline-If-Already-Exists() {
    $baselineExists = Test-Path -Path .\baseline.txt

    if ($baselineExists) {
        # Delete it
        Remove-Item -Path .\baseline.txt
    }
}

if ($response -eq "A".ToUpper()) {
    # Delete baseline.txt if it already exists
    Erase-Baseline-If-Already-Exists
```

Generate a new file in the directory path to verify the functionality of the `Erase-Baseline-If-Already-Exists` function.



|  |  |  |  |  |
|---|---|---|---|---|
| AAA | ✓ | 31/07/2024 10:23 PM | Text Document | 1 KB |
| BBB | ✓ | 31/07/2024 10:23 PM | Text Document | 1 KB |
| CCC | ✓ | 31/07/2024 10:23 PM | Text Document | 1 KB |
| DDD | ✓ | 31/07/2024 10:23 PM | Text Document | 1 KB |
| eee | ✓ | 1/08/2024 8:57 PM | Text Document | 0 KB |
| FIM | ✓ | 1/08/2024 8:57 PM | Windows PowerS... | 2 KB |

Test the **A** selection to ensure it works properly.

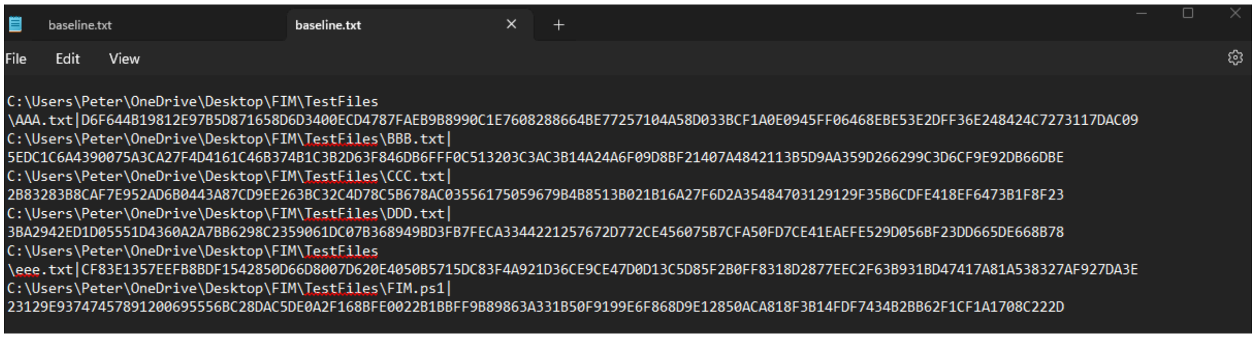


Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1
FIM.ps1

```
$response = Read-Host -Prompt "Please enter 'A' or 'B'"
Write-Host ""

Function Calculate-File-Hash($filepath) {
    $filehash = Get-FileHash -Path $filepath -Algorithm SHA512
    return $filehash
}

Function Erase-Baseline-If-Already-Exists() {
    $baselineExists = Test-Path -Path .\baseline.txt

    if ($baselineExists) {
        # Delete it
        Remove-Item -Path .\baseline.txt
    }
}

if ($response -eq "A".ToUpper()) {
    # Delete baseline.txt if it already exists
    Erase-Baseline-If-Already-Exists

    # Calculate Hash from the target files and store in baseline.txt

    #Collect all files in the target folder
    $files = Get-ChildItem -Path .\TestFiles
```

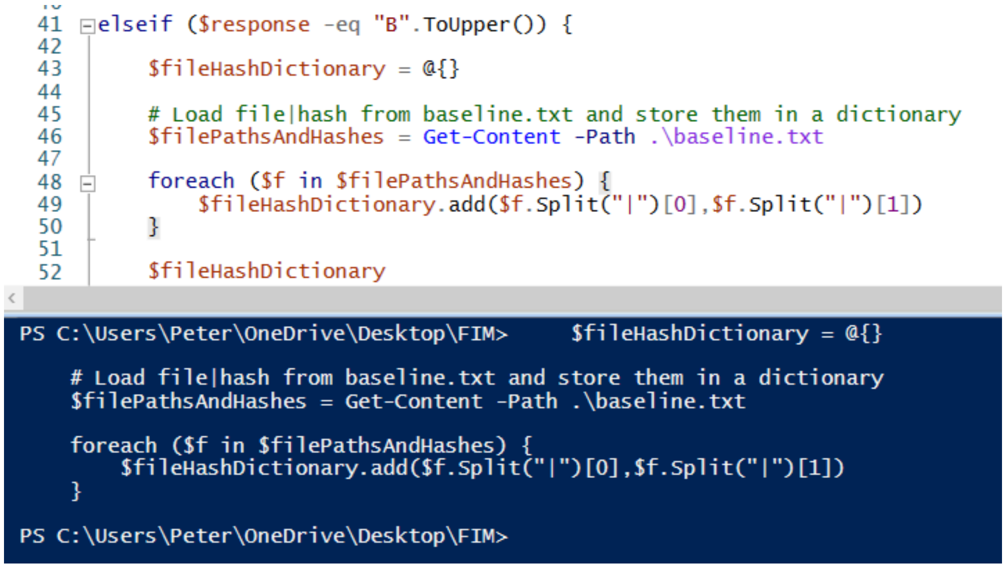
```
PS C:\Users\Peter\OneDrive\Desktop\FIM> C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\FIM.ps1

What would you like to do?
A) Collect new Baseline?
B) Begin monitoring files with saved Baseline?

Please enter 'A' or 'B': A


PS C:\Users\Peter\OneDrive\Desktop\FIM>
```

A new `baseline.txt` file has now been downloaded.



```
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles
\AAA.txt|D6F644B19812E97B5D871658D6D3400ECD4787FAEB9B8990C1E7608288664BE77257104A58D033BCF1A0E0945FF06468EBE53E2DFF36E248424C7273117DAC09
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\BBB.txt|
5EDC1C6A4390075A3CA27F4D4161C46B374B1C3B2D63F846DB6FFF0C513203C3AC3B14A24A6F09D8BF21407A4842113B5D9AA359D266299C3D6CF9E92DB66DBE
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\CCC.txt|
2B83283B8CAF7E952AD6B0443A87CD9EE263BC32C4D78C5B678AC03556175059679B4B8513B021B16A27F6D2A35484703129129F35B6CDFE418EF6473B1F8F23
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt|
3BA2942ED1D05551D4360A2A7BB6298C2359061DC07B368949BD3FB7FECA3344221257672D772CE456075B7CFA50FD7CE41EAEFE529D056BF23DD665DE668B78
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles
\eee.txt|CF83E1357EEFB8BDF1542850D66D8007D620E4050B5715DC83F4A921D36CE9CE47D0D13C5D85F2B0FF8318D2877EEC2F63B931BD47417A81A538327AF927DA3E
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\FIM.ps1|
23129E93747457891200695556BC28DAC5DE0A2F168BFE0022B1BBFF9B89863A331B50F9199E6F868D9E12850ACA818F3B14FDF7434B2BB62F1CF1A1708C222D
```

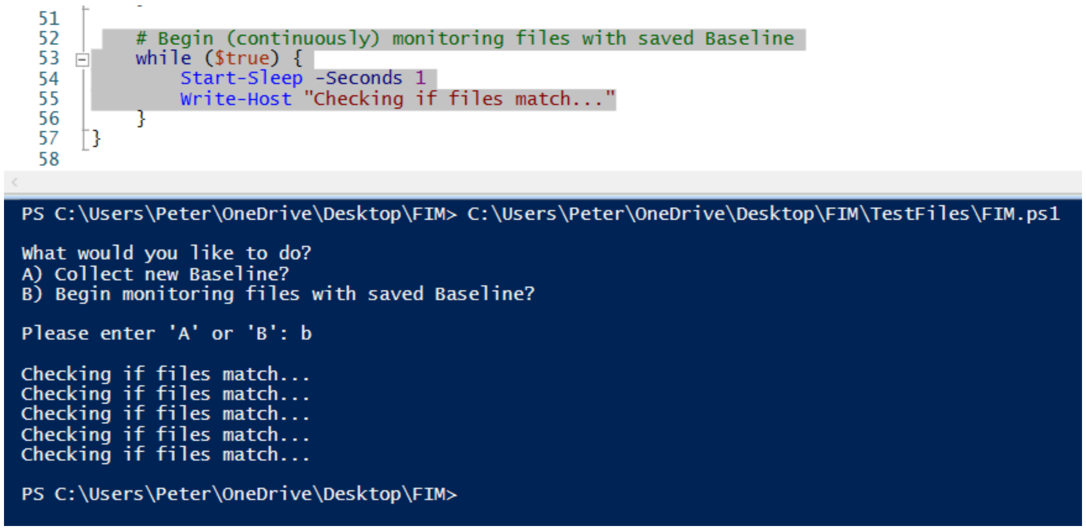
The file and hash data from `baseline.txt` are loaded into a dictionary, where the first element (0) is the directory path and the second element (1) is the file hash. The `Split` command divides these elements, storing element 0 as the key and element 1 as the value in the dictionary.



```
elseif ($response -eq "B".ToUpper()) {

    $fileHashDictionary = @{}

    # Load file|hash from baseline.txt and store them in a dictionary
    $filePathsAndHashes = Get-Content -Path .\baseline.txt

    foreach ($f in $filePathsAndHashes) {
        $fileHashDictionary.add($f.Split("|")[0],$f.Split("|")[1])
    }

    $fileHashDictionary
```

```
PS C:\Users\Peter\OneDrive\Desktop\FIM>     $fileHashDictionary = @{}

    # Load file|hash from baseline.txt and store them in a dictionary
    $filePathsAndHashes = Get-Content -Path .\baseline.txt

    foreach ($f in $filePathsAndHashes) {
        $fileHashDictionary.add($f.Split("|")[0],$f.Split("|")[1])
    }

PS C:\Users\Peter\OneDrive\Desktop\FIM>
```

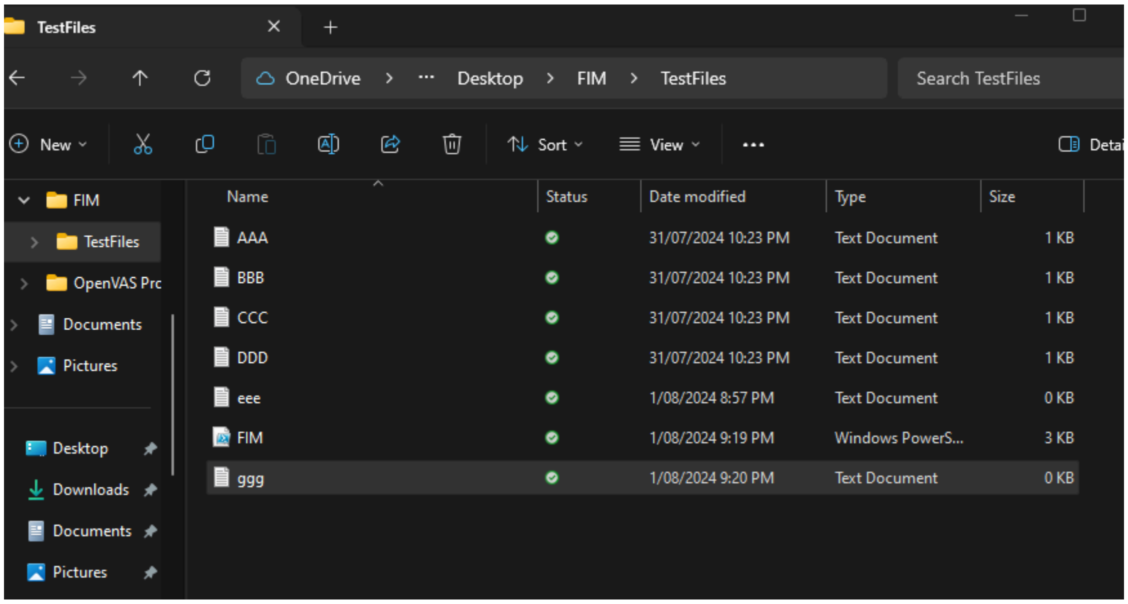
A `while` loop is used to monitor for new file creations and will notify the user if any additional files are detected in the directory path.



```
    # Begin (continuously) monitoring files with saved Baseline
    while ($true) {
        Start-Sleep -Seconds 1
        Write-Host "Checking if files match..."
    }
}
```

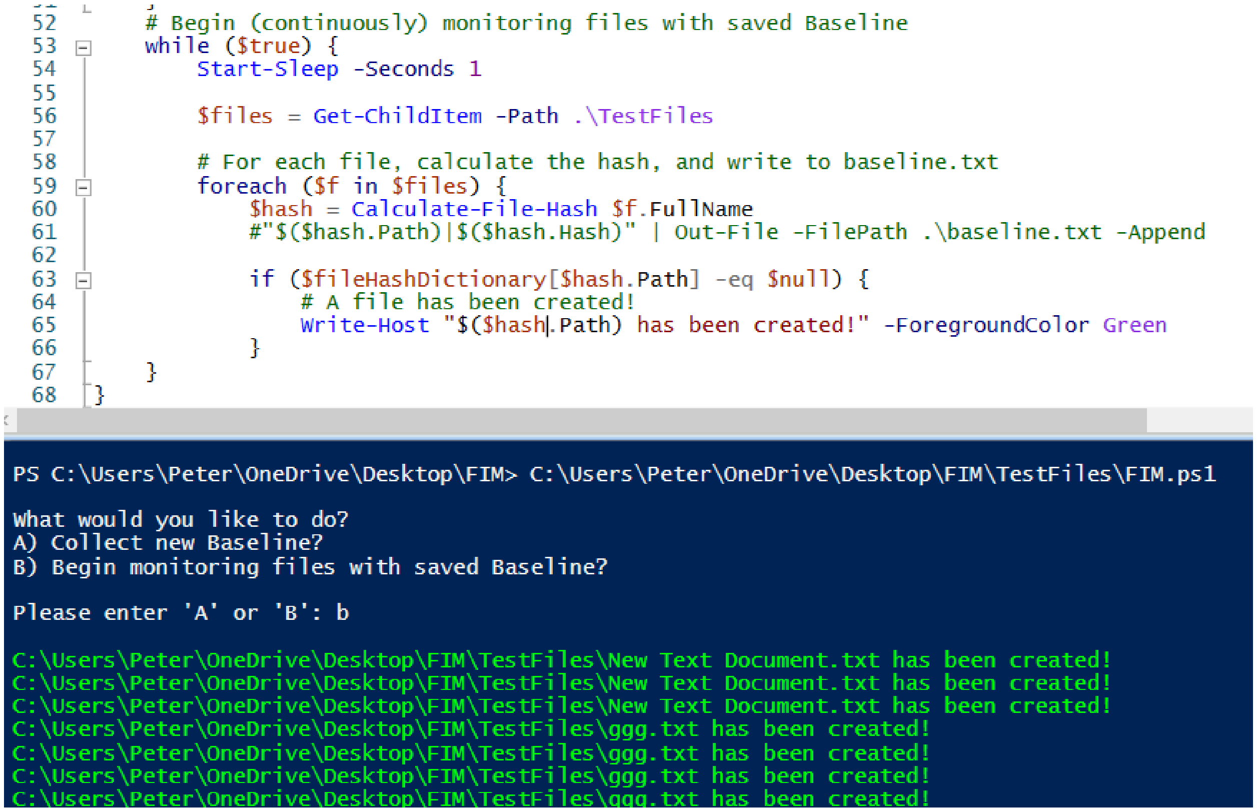
```
PS C:\Users\Peter\OneDrive\Desktop\FIM> C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\FIM.ps1

What would you like to do?
A) Collect new Baseline?
B) Begin monitoring files with saved Baseline?

Please enter 'A' or 'B': b

Checking if files match...
Checking if files match...
Checking if files match...
Checking if files match...
Checking if files match...

PS C:\Users\Peter\OneDrive\Desktop\FIM>
```

Create a new file named `ggg.txt` to test if the system detects unauthorised files. Since this file is not listed in the `baseline.txt`, it should be flagged as an unauthorised file in the folder.



| Name | Status | Date modified | Type | Size |
|---|---|---|---|---|
| AAA | ✓ | 31/07/2024 10:23 PM | Text Document | 1 KB |
| BBB | ✓ | 31/07/2024 10:23 PM | Text Document | 1 KB |
| CCC | ✓ | 31/07/2024 10:23 PM | Text Document | 1 KB |
| DDD | ✓ | 31/07/2024 10:23 PM | Text Document | 1 KB |
| eee | ✓ | 1/08/2024 8:57 PM | Text Document | 0 KB |
| FIM | ✓ | 1/08/2024 9:19 PM | Windows PowerS... | 3 KB |
| ggg | ✓ | 1/08/2024 9:20 PM | Text Document | 0 KB |

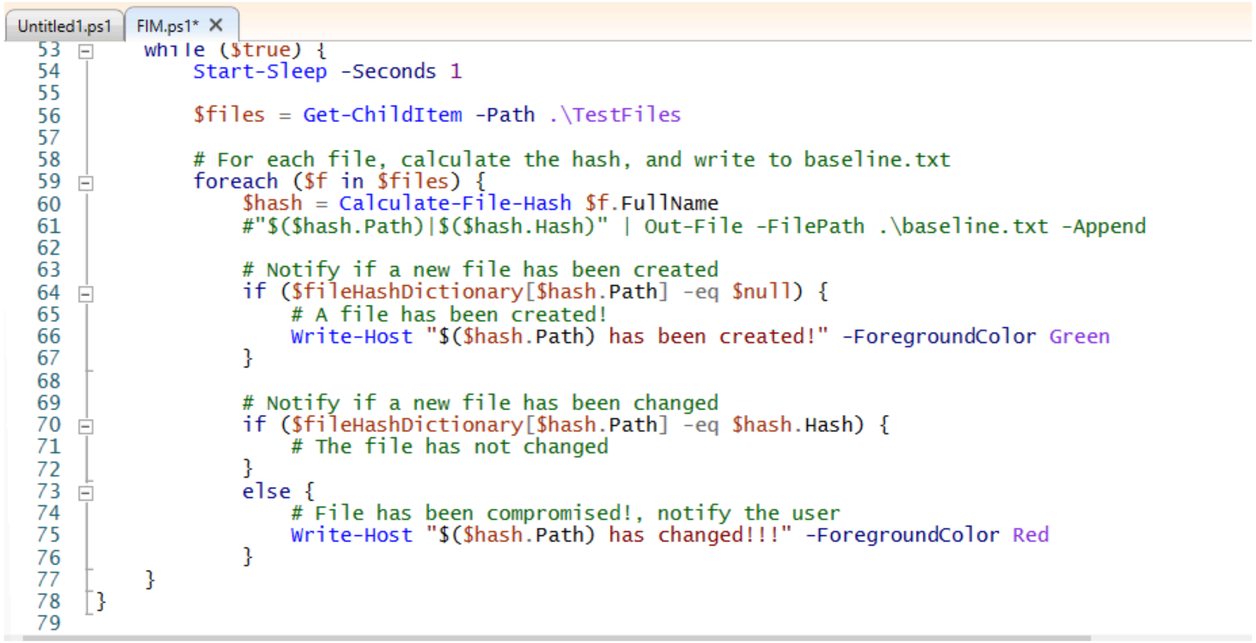
The test results, shown below, indicate that the `ggg.txt` file has been recognised as an unauthorised file.



```
    # Begin (continuously) monitoring files with saved Baseline
    while ($true) {
        Start-Sleep -Seconds 1

        $files = Get-ChildItem -Path .\TestFiles

        # For each file, calculate the hash, and write to baseline.txt
        foreach ($f in $files) {
            $hash = Calculate-File-Hash $f.FullName
            #"$($hash.Path)|$($hash.Hash)" | Out-File -FilePath .\baseline.txt -Append

            if ($fileHashDictionary[$hash.Path] -eq $null) {
                # A file has been created!
                Write-Host "$($hash.Path) has been created!" -ForegroundColor Green
            }
    }
}
```

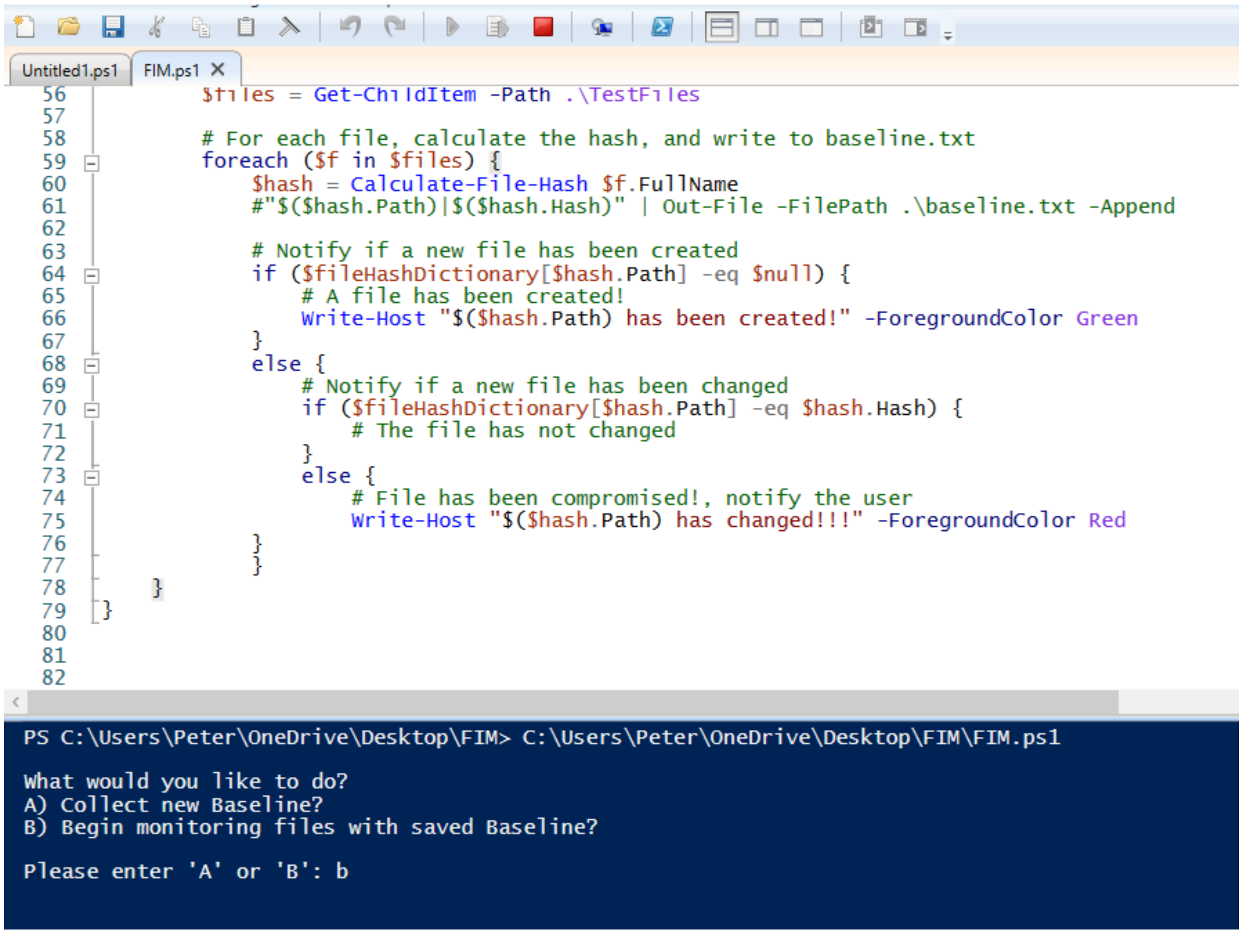
```
PS C:\Users\Peter\OneDrive\Desktop\FIM> C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\FIM.ps1

What would you like to do?
A) Collect new Baseline?
B) Begin monitoring files with saved Baseline?

Please enter 'A' or 'B': b

C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Document.txt has been created!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Document.txt has been created!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Document.txt has been created!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\ggg.txt has been created!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\ggg.txt has been created!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\ggg.txt has been created!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\ggg.txt has been created!
```

Execute the script to verify that it identifies changes in the folder, such as new or modified files.



```
    while ($true) {
        Start-Sleep -Seconds 1

        $files = Get-ChildItem -Path .\TestFiles

        # For each file, calculate the hash, and write to baseline.txt
        foreach ($f in $files) {
            $hash = Calculate-File-Hash $f.FullName
            #"$($hash.Path)|$($hash.Hash)" | Out-File -FilePath .\baseline.txt -Append

            # Notify if a new file has been created
            if ($fileHashDictionary[$hash.Path] -eq $null) {
                # A file has been created!
                Write-Host "$($hash.Path) has been created!" -ForegroundColor Green
            }

            # Notify if a new file has been changed
            if ($fileHashDictionary[$hash.Path] -eq $hash.Hash) {
                # The file has not changed
            }
            else {
                # File has been compromised!, notify the user
                Write-Host "$($hash.Path) has changed!!!" -ForegroundColor Red
            }
    }
}
```

Testing will ensure that the script provides notifications to the user whenever a file is added, created, or modified in the folder, with prompts informing the main user of the changes.

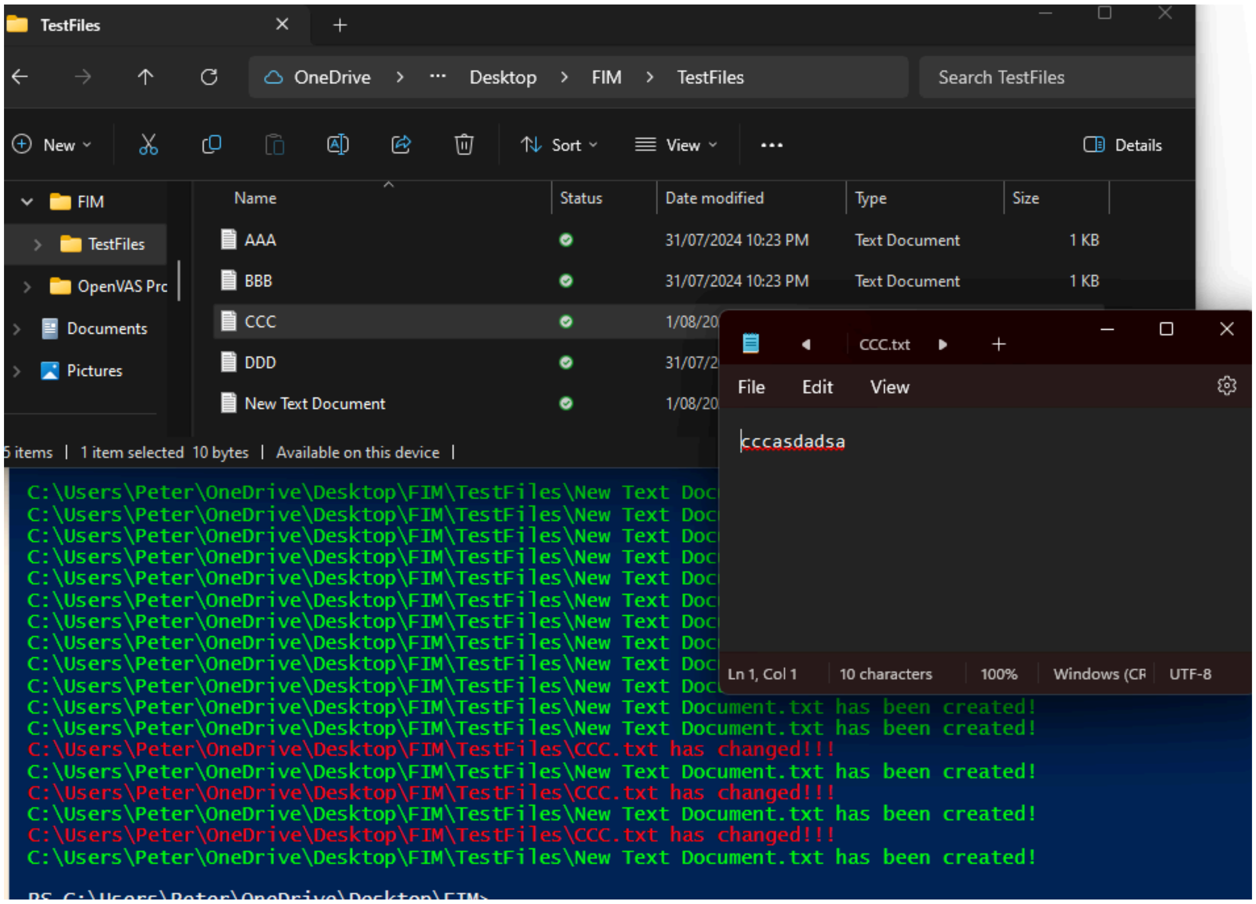


Untitled1.ps1 FIM.ps1

```
        $files = Get-ChildItem -Path .\TestFiles

        # For each file, calculate the hash, and write to baseline.txt
        foreach ($f in $files) {
            $hash = Calculate-File-Hash $f.FullName
            #"$($hash.Path)|$($hash.Hash)" | Out-File -FilePath .\baseline.txt -Append

            # Notify if a new file has been created
            if ($fileHashDictionary[$hash.Path] -eq $null) {
                # A file has been created!
                Write-Host "$($hash.Path) has been created!" -ForegroundColor Green
            }
            else {
                # Notify if a new file has been changed
                if ($fileHashDictionary[$hash.Path] -eq $hash.Hash) {
                    # The file has not changed
                }
                else {
                    # File has been compromised!, notify the user
                    Write-Host "$($hash.Path) has changed!!!" -ForegroundColor Red
            }
            }
    }
}
```

```
PS C:\Users\Peter\OneDrive\Desktop\FIM> C:\Users\Peter\OneDrive\Desktop\FIM\FIM.ps1

What would you like to do?
A) Collect new Baseline?
B) Begin monitoring files with saved Baseline?

Please enter 'A' or 'B': b
```

As indicated earlier, if a new text file is created, the system will notify the user that the file is not present in the original `baseline.txt` hash records. Additionally, if data is added to an existing file, the user will be alerted that the file has been altered, as its hash will no longer match the original.



| Name | Status | Date modified | Type | Size |
|---|---|---|---|---|
| AAA | ✓ | 31/07/2024 10:23 PM | Text Document | 1 KB |
| BBB | ✓ | 31/07/2024 10:23 PM | Text Document | 1 KB |
| CCC | ✓ | 1/08/20 | | |
| DDD | ✓ | 31/07/2 | | |
| New Text Document | ✓ | 1/08/20 | | |

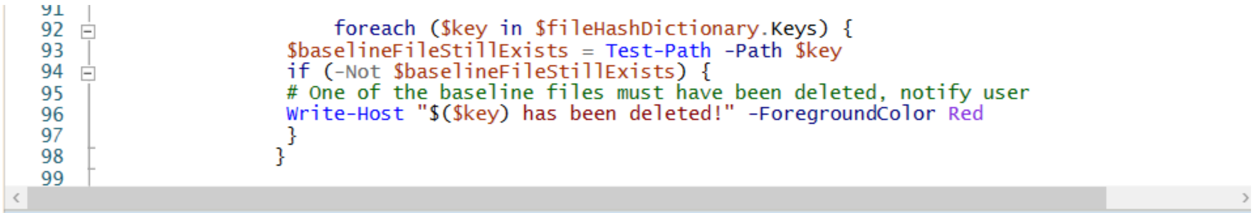
5 items | 1 item selected 10 bytes | Available on this device |

```
cccasdadsa
```

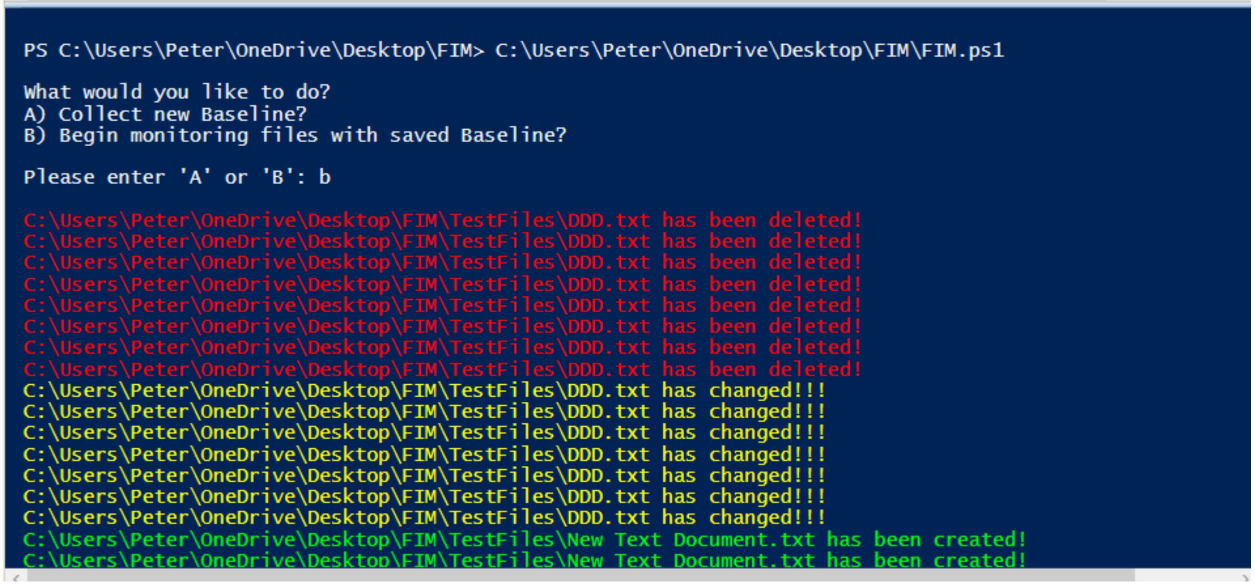
Ln 1, Col 1 | 10 characters | 100% | Windows (CR | UTF-8

```
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Doc
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Doc
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Doc
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Doc
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Doc
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Doc
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Doc
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Doc
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Doc
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Doc
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Document.txt has been created!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Document.txt has been created!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\CCC.txt has changed!!!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Document.txt has been created!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\CCC.txt has changed!!!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Document.txt has been created!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\CCC.txt has changed!!!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Document.txt has been created!

PS C:\Users\Peter\OneDrive\Desktop\FIM>
```

This section of the commands informs users that if a text file is deleted from the folder, the `baseline.txt` will verify all stored hashes. If any files are missing, the user will be notified of the deletion.



```
            foreach ($key in $fileHashDictionary.Keys) {
        $baselineFileStillExists = Test-Path -Path $key
        if (-Not $baselineFileStillExists) {
        # One of the baseline files must have been deleted, notify user
        Write-Host "$($key) has been deleted!" -ForegroundColor Red
        }
      }
```



```
PS C:\Users\Peter\OneDrive\Desktop\FIM> C:\Users\Peter\OneDrive\Desktop\FIM\FIM.ps1

What would you like to do?
A) Collect new Baseline?
B) Begin monitoring files with saved Baseline?

Please enter 'A' or 'B': b

C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt has been deleted!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt has been deleted!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt has been deleted!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt has been deleted!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt has been deleted!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt has been deleted!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt has been deleted!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt has been deleted!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt has changed!!!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt has changed!!!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt has changed!!!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt has changed!!!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt has changed!!!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt has changed!!!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\DDD.txt has changed!!!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Document.txt has been created!
C:\Users\Peter\OneDrive\Desktop\FIM\TestFiles\New Text Document.txt has been created!
```