```
  ____    __________________________________________________________________________________
 //¯¯\\__//¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\\___
 \\__//¯¯¯ [FightingEntropy(π][2022.10.1]                                                    ___//¯¯\\
  ¯¯¯\\_____________________________________________________________________________________//¯¯\\__//
      ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯    ¯¯¯¯
```

```
\_______________________________________________________________________________________________________/
  [FightingEntropy(π)][2022.10.1] /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

Greetings reader,
This is a preview of an upcoming version of | [FightingEntropy(π)][2022.10.1] |

```
_______________________________________________________________________________________
| 10/28/22 | 2022_1028-([FightingEntropy(π)][2022.10.1]) | https://youtu.be/S7k4lZdPE-I |
¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

This new version is essentially reinventing the installation and removal process, as well as how it obtains the files from the project site, validates the components, and by the time it's ready to roll, it'll begin to filter out the stuff that doesn't work in Linux, PowerShell Core, and Windows PowerShell.

That's basically dancing with death in and of itself, being able to prevent certain functions from loading on various other platforms, but still having all of the files intact.

Effectively, there'll be a different version of the module for each system it is deployed to, for each version that the module detects or is able to run.

So, what I mean is this.
If I'm running PowerShell Core, then the module will only load the components that PowerShell Core can run.
If I switch to PowerShell Desktop, then the module will load all of the components.

If I'm running on a Windows server operating system, well guess what dude...?
NOW I'm gonna have access to the heavy hitting components featured in the module.

I could go on, but ultimately, there's a lot to consider and it all bleeds right back to the manner of how the module installs itself. As well as how it instantiates itself.

The installation function is below, and it is just what I've managed to do over the last couple days or so.
It is including a light version of the Write-Theme function, which I think will win some people over when they go to use it, and then they're like “WHOA, COLORS IN THE CONSOLE AND STUFF...?”

Yeah, buddy.

The classes below are all contained within the function, and they need to be described before the process is fully understood.

```
                                                                                     ________________________________/
\___________________________________________________________________________________/ [FightingEntropy(π)][2022.10.1]
  <FightingEntropy.Module.ThemeBlock> /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

This class is a part of (3) individual classes that are each related to the Write-Theme function within the module itself. Together, they represent a lightweight of that particular function.

This particular class is a “BLOCK”, which represents a single [Object] in (an array/collection) of [Object[]].

Consider this like a Lego block, it has a COLOR, and it also has a HEIGHT, and WIDTH, and they tell the function Write-Host what to display for a particular line of characters.

```
# // ___________________________________________________________________
# // | This is a 1x[track] x 4[char] chunk of information for Write-Host |
# // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

Class ThemeBlock
{
    [UInt32]   $Index
    [Object]  $String
    [UInt32]    $Fore
    [UInt32]    $Back
    [UInt32]    $Last
    ThemeBlock([Int32]$Index,[String]$String,[Int32]$Fore,[Int32]$Back)
    {
        $This.Index  = $Index
        $This.String = $String
        $This.Fore   = $Fore
        $This.Back   = $Back
        $This.Last   = 1
    }
    Write([UInt32]$0,[UInt32]$1,[UInt32]$2,[UInt32]$3)
    {
        $Splat = @{

            Object          = $This.String
            ForegroundColor = @($0,$1,$2,$3)[$This.Fore]
            BackgroundColor = $This.Back
            NoNewLine       = $This.Last
        }

        Write-Host @Splat
    }
    [String] ToString()
    {
        Return "<FightingEntropy.Module.ThemeBlock>"
    }
}
```

```
                                                                                  _____________________________________/
\________________________________________________________________________________/ <FightingEntropy.Module.ThemeBlock>
  <FightingEntropy.Module.ThemeTrack> /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

The above class, is (controlled by/contained within) this particular class, and though this class doesn't DO a whole lot, what it DOES do, is provide a single track within a stack of track objects.

```
    # // _______________________________________________
    # // | Represents a 1x[track] in a stack of tracks |
    # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

    Class ThemeTrack
    {
        [UInt32] $Index
        [Object] $Content
        ThemeTrack([UInt32]$Index,[Object]$Track)
        {
            $This.Index   = $Index
            $This.Content = $Track
        }
        [String] ToString()
        {
            Return "<FightingEntropy.Module.ThemeTrack>"
        }
    }
```

```
                                                                                  _____________________________________/
\________________________________________________________________________________/ <FightingEntropy.Module.ThemeTrack>
  <FightingEntropy.Module.ThemeStack> /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

This particular class effectively controls the above (2) classes, and puts the common denominator of the property `[Object] $Face` into a single scope so that the above classes do not have to recast the same information for each time that the function is run.

```
    # // ___________________________________________
    # // | Generates an actionable write-host object |
    # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

    Class ThemeStack
    {
        Hidden [Object]  $Face
        Hidden [Object] $Track
        ThemeStack([UInt32]$Slot,[String]$Message)
        {
            $This.Main($Message)
            $This.Write($This.Palette($Slot))
        }
        ThemeStack([String]$Message)
        {
            $This.Main($Message)
            $This.Write($This.Palette(0))
        }
        Main([String]$Message)
        {
            $This.Face = $This.Mask()
            $This.Reset()
            $This.Insert($Message)
        }
        [UInt32[]] Palette([UInt32]$Slot)
        {
            If ($Slot -gt 35)
            {
                Throw "Invalid entry"
            }

            Return @( Switch ($Slot)
            {
                00 {10,12,15,00} 01 {12,04,15,00} 02 {10,02,15,00} # Default, R*/Error,   G*/Success
                03 {01,09,15,00} 04 {03,11,15,00} 05 {13,05,15,00} # B*/Info, C*/Verbose, M*/Feminine
                06 {14,06,15,00} 07 {00,08,15,00} 08 {07,15,15,00} # Y*/Warn, K*/Evil,    W*/Host
                09 {04,12,15,00} 10 {12,12,15,00} 11 {04,04,15,00} # R!,      R+,         R-
                12 {02,10,15,00} 13 {10,10,15,00} 14 {02,02,15,00} # G!,      G+,         G-
                15 {09,01,15,00} 16 {09,09,15,00} 17 {01,01,15,00} # B!,      B+,         B-
                18 {11,03,15,00} 19 {11,11,15,00} 20 {03,03,15,00} # C!,      C+,         C-
                21 {05,13,15,00} 22 {13,13,15,00} 23 {05,05,15,00} # M!,      M+,         M-
                24 {06,14,15,00} 25 {14,14,15,00} 26 {06,06,15,00} # Y!,      Y+,         Y-
                27 {08,00,15,00} 28 {08,08,15,00} 29 {00,00,15,00} # K!,      K+,         K-
                30 {15,07,15,00} 31 {15,15,15,00} 32 {07,07,15,00} # W!,      W+,         W-
                33 {11,06,15,00} 34 {06,11,15,00} 35 {11,12,15,00} # Steel*,  Steel!,     C+R+
            })
        }
        [Object] Mask()
        {
            Return ("20202020 5F5F5F5F AFAFAFAF 2020202F 5C202020 2020205C 2F202020 5C5F5F2F "+
            "2FAFAF5C 2FAFAFAF AFAFAF5C 5C5F5F5F 5F5F5F2F 205F5F5F" -Split " ") | % { $This.Convert($_) }
        }
        [String] Convert([String]$Line)
        {
            Return [Char[]]@(0,2,4,6 | % { [Convert]::FromHexString($Line.Substring($_,2)) }) -join ''
        }
        Add([String]$Mask,[String]$Fore)
        {
            # // ____________________________
            # // | Expands the mask strings |
            # // ----------------------------

            $Object        = Invoke-Expression $Mask | % { $This.Face[$_] }
            $FG            = Invoke-Expression $Fore
            $BG            = @(0)*30

            # // ____________________________
            # // | Generates a track object |
            # // ----------------------------

            $Hash          = @{ }
            ForEach ($X in 0..($Object.Count-1))
            {
                $Item      = [ThemeBlock]::New($X,$Object[$X],$FG[$X],$BG[$X])
                If ($X -eq $Object.Count-1)
                {
                    $Item.Last = 0
                }
                $Hash.Add($Hash.Count,$Item)
            }
            $This.Track  += [ThemeTrack]::New($This.Track.Count,$Hash[0..($Hash.Count-1)])
        }
        [Void] Reset()
        {
            $This.Track = @( )

            # // ____________________________
            # // | Generates default tracks |
            # // ----------------------------

            $This.Add("0,1,0+@(1)*25+0,0","@(0)*30")
            $This.Add("3,8,7,9+@(2)*23+10,11,0","0,1,0+@(1)*25+0,0")
            $This.Add("5,7,9,13+@(0)*23+12,8,4","0,1,1+@(2)*24+1,1,0")
            $This.Add("0,10,11+@(1)*23+12+8,7,6","0,0+@(1)*25+0,1,0")
            $This.Add("0,0+@(2)*25+0,2,0","@(0)*30")
        }
        Insert([String]$String)
        {
            $This.Reset()
            $String = " $String"
            Switch ($String.Length)
            {
                {$_ -lt 84}
                {
                    $String += (@(" ") * (84 - ($String.Length+1)) -join '' )
                }
                {$_ -ge 84}
                {
                    $String  = $String.Substring(0,84) + "..."
                }
            }
            $Array = [Char[]]$String
            $Hash  = @{ }
            $Block = ""
            ForEach ($X in 0..($Array.Count-1))
            {
                If ($X % 4 -eq 0 -and $Block -ne "")
                {
                    $Hash.Add($Hash.Count,$Block)
                    $Block = ""
                }
                $Block += $Array[$X]
            }

            ForEach ($X in 0..($Hash.Count-1))
            {
                $This.Track[2].Content[$X+3].String = $Hash[$X]
            }
        }
        [Void] Write([UInt32[]]$Palette)
        {
            $This.Track | % Content | % Write $Palette
        }
        [String] ToString()
        {
            Return "<FightingEntropy.Module.ThemeStack>"
        }
    }
```

```
______________________________________________________________________________________________________________
| ThemeStack([UInt32]$Slot,[String]$Message) | Primary entry into the class, accepts (2) parameters            |
| ThemeStack([String]$Message)               | Secondary entry into the class, accepts (1) parameter           |
| Main([String]$Message)                     | Main method will combine the following methods into one         |
| [UInt32[]] Palette([UInt32]$Slot)          | Returns a mask of colors which allow the theme to work          |
| [Object] Mask()                            | Returns the faces for the blocks                                |
| [String] Convert([String]$Line)            | Converts a string of joined hexadecimal characters to integers  |
| Add([String]$Mask,[String]$Fore)           | Adds a single line of mask information w/ foregroundcolor       |
| [Void] Reset()                             | Clears out any existing information, AND resets the class       |
| Insert([String]$String)                    | Processes the input string into the mask object                 |
| [Void] Write([UInt32[]]$Palette)           | Tells the Write-Host object that it better get to work, or else~! |
    | [String] ToString()                          | Returns the name of the type/class                         |
    ----------------------------------------------------------------------------------------------------------------------
                                                                                       _____________________________________/
\_____________________________________________________________________________/ <FightingEntropy.Module.ThemeStack>
  <FightingEntropy.Module.OSProperty> /----------------------------------------------------------------------------------\
/--------------------------------------
```

This is essentially an extension of the type PSNoteProperty, it allows the source & index to be injected.

```
# // ____________________________________________________
# // | Property object which includes source and index  |
# // ----------------------------------------------------

Class OSProperty
{
    [String] $Source
    Hidden [UInt32] $Index
    [String] $Name
    [Object] $Value
    OSProperty([String]$Source,[UInt32]$Index,[String]$Name,[Object]$Value)
    {
        $This.Source = $Source
        $This.Index  = $Index
        $This.Name   = $Name
        $This.Value  = $Value
    }
    [String] ToString()
    {
        Return "<FightingEntropy.Module.OSProperty>"
    }
}
```

```
                                                                                       _____________________________________/
\_____________________________________________________________________________/ <FightingEntropy.Module.OSProperty>
  <FightingEntropy.Module.OSPropertySet> /-------------------------------------------------------------------------------\
/-----------------------------------------
```

This is a collection of properties for a particular source, and it allows THEM to be indexed and accessed as a collection from a larger parent scope, and allows each source object to remain its' own independent object.

```
# // __________________________________________________________
# // | Container object for indexed OS (property/value) pairs |
# // ----------------------------------------------------------

Class OSPropertySet
{
    Hidden [UInt32] $Index
    [String] $Source
    [Object] $Property
    OSPropertySet([UInt32]$Index,[String]$Source)
    {
        $This.Index     = $Index
        $This.Source    = $Source
        $This.Property  = @( )
    }
    Add([String]$Name,[Object]$Value)
    {
        $This.Property += [OSProperty]::New($This.Source,$This.Property.Count,$Name,$Value)
    }
    [String] ToString()
    {
        $D = ([String]$This.Property.Count).Length
        Return "({0:d$D}) <FightingEntropy.Module.OSPropertySet[{1}]>" -f $This.Property.Count, $This.Source
    }
}
```

```
                                                                                    ________________________________________/
\__________________________________________________________________________/ <FightingEntropy.Module.OSPropertySet>
```

```
  <FightingEntropy.Module.OS> /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

Truncates some information from the automatic variables in relation to the operating system and PSVersion types.

```
# // _______________________________________________________
# // | Collects various details about the operating system |
# // | specifically for cross-platform compatibility       |
# // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

Class OS
{
    [Object]   $Caption
    [Object]  $Platform
    [Object] $PSVersion
    [Object]      $Type
    [Object]    $Output
    OS()
    {
        $This.Output = @( )

        # // _______________
        # // | Environment |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        $This.AddPropertySet("Environment")

        Get-ChildItem Env:              | % { $This.Add(0,$_.Key,$_.Value) }

        # // ____________
        # // | Variable |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯

        $This.AddPropertySet("Variable")

        Get-ChildItem Variable:         | % { $This.Add(1,$_.Name,$_.Value) }

        # // ________
        # // | Host |
        # // ¯¯¯¯¯¯¯¯

        $This.AddPropertySet("Host")

        (Get-Host).PSObject.Properties  | % { $This.Add(2,$_.Name,$_.Value) }

        # // ______________
        # // | PowerShell |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        $This.AddPropertySet("PowerShell")

        (Get-Variable PSVersionTable | % Value).GetEnumerator() | % { $This.Add(3,$_.Name,$_.Value) }

        # // ____________________________________
        # // | Assign hashtable to output array |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        $This.Caption   = $This.Tx("PowerShell","OS")
        $This.Platform  = $This.Tx("PowerShell","Platform")
        $This.PSVersion = $This.Tx("PowerShell","PSVersion")
        $This.Type      = $This.GetOSType()
    }
    [Object] Tx([String]$Source,[String]$Name)
    {
        Return $This.Output | ? Source -eq $Source | % Property | ? Name -eq $Name | % Value
    }
    Add([UInt32]$Index,[String]$Name,[Object]$Value)
    {
        $This.Output[$Index].Add($Name,$Value)
    }
    AddPropertySet([String]$Name)
```

```
        {
            $This.Output += [OSPropertySet]::New($This.Output.Count,$Name)
        }
        [String] GetWinCaption()
        {
            Return "[wmiclass]'Win32_OperatingSystem' | % GetInstances | % Caption"
        }
        [String] GetWinType()
        {
            Return @(Switch -Regex (Invoke-Expression $This.GetWinCaption())
            {
                "Windows (10|11)" { "Win32_Client" } "Windows Server" { "Win32_Server" }
            })
        }
        [String] GetOSType()
        {
            Return @( If ($This.Version.Major -gt 5)
            {
                If (Get-Item Variable:\IsLinux | % Value)
                {
                    (hostnamectl | ? { $_ -match "Operating System" }).Split(":")[1].TrimStart(" ")
                }

                Else
                {
                    $This.GetWinType()
                }
            }

            Else
            {
                $This.GetWinType()
            })
        }
        [String] ToString()
        {
            Return "<FightingEntropy.Module.OS>"
        }
    }
```

```
                                                                                           _____________________________/
\_________________________________________________________________________________________/ <FightingEntropy.Module.OS>
  <FightingEntropy.Module.File> /---------------------------------------------------------------------------------------\
/-------------------------------
```

Encapsulates the entire process of collection, validation, writing to disk, and keeping each file organized.

```
    # // ______________________________________________________________
    # // | Manifest file -> filesystem object (collection/validation) |
    # // --------------------------------------------------------------

    Class File
    {
        Hidden [UInt32]    $Index
        [String]            $Type
        [String]            $Name
        [String]            $Hash
        [UInt32]          $Exists
        Hidden [String] $Fullname
        Hidden [String]   $Source
        Hidden [UInt32]    $Match
        Hidden [Object]  $Content
        File([UInt32]$Index,[String]$Type,[String]$Parent,[String]$Name,[String]$Hash)
        {
            $This.Index    = $Index
            $This.Type     = $Type
            $This.Name     = $Name
            $This.Fullname = "$Parent\$Name"
            $This.Hash     = $Hash
            $This.TestPath()
        }
        [String] FolderName()
        {
            Return @{

                Class    = "Classes"
                Control  = "Control"
                Function = "Functions"
                Graphic  = "Graphics"

            }[$This.Type]
        }
        TestPath()
        {
            If (!$This.Fullname)
            {
                Throw "Exception [!] Resource path not set"
            }

            $This.Exists   = [System.IO.File]::Exists($This.Fullname)
        }
        [Void] Create()
        {
            $This.TestPath()

            If (!$This.Exists)
            {
                [System.IO.File]::Create($This.Fullname).Dispose()
                $This.Exists = 1
            }
        }
        [Void] Delete()
        {
            $This.TestPath()

            If ($This.Exists)
            {
                [System.IO.File]::Delete($This.Fullname)
                $This.Exists = 0
            }
        }
        SetSource([String]$Source)
        {
            $This.Source   = "{0}/blob/main/{1}/{2}?raw=true" -f $Source, $This.FolderName(), $This.Name
        }
        Download()
        {
            Try
            {
                $This.Content = Invoke-WebRequest $This.Source -UseBasicParsing | % Content
            }
            Catch
            {
                Throw "Exception [!] An unspecified error occurred"
            }
        }
        Write()
        {
            If (!$This.Content)
            {
                Throw "Exception [!] Content not assigned, cannot (write/set) content."
            }

            If (!$This.Exists)
            {
                Throw "Exception [!] File does not exist."
            }

            Try
            {
                If ($This.Name -match "\.+(jpg|jpeg|png|bmp|ico)")
                {
```

```
                [System.IO.File]::WriteAllBytes($This.Fullname,[Byte[]]$This.Content)
            }
            Else
            {
                [System.IO.File]::WriteAllText($This.Fullname,
                                               $This.Content,
                                               [System.Text.UTF8Encoding]$False)
            }
        }
        Catch
        {
            Throw "Exception [!] An unspecified error has occurred"
        }
    }
    GetContent()
    {
        If (!$This.Exists)
        {
            Throw "Exception [!] File does not exist, it needs to be created first."
        }

        Try
        {
            If ($This.Name -match "\.+(jpg|jpeg|png|bmp|ico)")
            {
                $This.Content = [System.IO.File]::ReadAllBytes($This.Fullname)
            }
            Else
            {
                $This.Content = [System.IO.File]::ReadAllLines($This.Fullname,
                                                               [System.Text.UTF8Encoding]$False)
            }
        }
        Catch
        {
            Throw "Exception [!] An unspecified error has occurred"
        }
    }
    [String] ToString()
    {
        Return "<FightingEntropy.Module.File>"
    }
}
```

```
                                                                                         _______________________________/
\_______________________________________________________________________________________/ <FightingEntropy.Module.File>
  <FightingEntropy.Module.Folder> /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

Works with the file system as well as the FE module manifest, to orchestrate the process of 1) installing, 2) removing, or 3) validating the module's resource files.

```
# // ________________________________________
# // | Manifest folder -> filesystem object |
# // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

Class Folder
{
    Hidden [UInt32]    $Index
    [String]            $Type
    [String]            $Name
    [String]        $Fullname
    [UInt32]          $Exists
    Hidden [Object]    $Item
    Folder([UInt32]$Index,[String]$Type,[String]$Parent,[String]$Name)
    {
        $This.Index     = $Index
        $This.Type      = $Type
        $This.Name      = $Name
        $This.Fullname  = "$Parent\$Name"
```

```
            $This.Item      = @( )
            $This.TestPath()
        }
        Add([String]$Name,[Object]$Hash)
        {
            $File           = [File]::New($This.Item.Count,$This.Type,$This.Fullname,$Name,$Hash)
            If ($File.Exists)
            {
                $Hash       = Get-FileHash $File.Fullname | % Hash
                If ($Hash -eq $File.Hash)
                {
                    $File.Match = 1
                }
                If ($Hash -ne $File.Hash)
                {
                    $File.Match = 0
                }
            }

            $This.Item     += $File
        }
        TestPath()
        {
            $This.Exists = [System.IO.Directory]::Exists($This.Fullname)
        }
        [Void] Create()
        {
            $This.TestPath()

            If (!$This.Exists)
            {
                [System.IO.Directory]::CreateDirectory($This.Fullname)
                $This.Exists = 1
            }
        }
        [Void] Delete()
        {
            $This.TestPath()

            If ($This.Exists)
            {
                [System.IO.Directory]::Delete($This.Fullname)
                $This.Exists = 0
            }
        }
        [String] ToString()
        {
            $D = ([String]$This.Item.Count).Length
            Return "({0:d$D}) <FightingEntropy.Module.Folder[{1}]>" -f $This.Item.Count, $This.Name
        }
    }
```

```
                                                                                     _________________________________/
\___________________________________________________________________________________/ <FightingEntropy.Module.Folder>
  <FightingEntropy.Module.Manifest> /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

This particular class is formatted slightly differently than in the installation function

```
    # // _____________________________________________________________________
    # // | File manifest container, laid out for hash (insertion+validation) |
    # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

    Class Manifest
    {
        [String]       $Source
        [String]     $Resource
        Hidden [UInt32] $Depth
        Hidden [UInt32] $Total
        [Object]       $Output
        Manifest([String]$Source,[String]$Resource)
        {
            $This.Source   = $Source
            $This.Resource = $Resource
            $This.Output   = @( )

            # // ___________
            # // | Classes |
            # // -----------

            $This.AddFolder("Class","Classes")

            ("_Cache.ps1"                       ,
             "DA2DAF257EB99FFF7E414B83D3659B55646D54DE338D81DB0675D98E76EAE630") ,
            ("_Drive.ps1"                       ,
             "B6CE9795F97896C64C991D7C1314E36939C92C514312C0630C8C5B9A1A972388") ,
            ("_Drives.ps1"                      ,
             "0B1E99226B2345B596B0EC6CF03E85235B43F07862A43C6ACDB74144E176C744") ,
            ("_File.ps1"                        ,
             "26E3C8D2C23A96F75E953603B7C4F028609FBE78444CF8AEAE53FE89B41B5904") ,
            ("_FirewallRule.ps1"                ,
             "4E0B2B849674C2A36D16E32D9914E95B1F18C5645F393CFA321322CB2122EC3") ,
            ("_Icons.ps1"                       ,
             "D469DB44301A69F832417744A58EE60D03AC4C83F8C8AFA3D4D5E765C02BD9F2") ,
            ("_Shortcut.ps1"                    ,
             "FCA9DC157A2BDF9D66AA9A2803D9A419F28D375A7C4ABFFA23A925F544778B6C") ,
            ("_ViperBomb.ps1"                   ,
             "82D3FDBA40360D8E0123CDAEFF3A0A8F0AE0105CC4D6791EBF8B40BD0BF64162") | % {

                $This.Add(0,$_[0],$_[1])
            }

            # // ___________
            # // | Control |
            # // -----------

            $This.AddFolder("Control","Control")

            ("Computer.png"                     ,
             "87EAB4F74B38494A960BEBF69E472AB0764C3C7E782A3F74111F993EA31D1075") ,
            ("DefaultApps.xml"                  ,
             "939CE697246AAC96C6F6A4A285C8EE285D7C5090523DB77831FF76D5D4A31539") ,
            ("failure.png"                      ,
             "59D479A0277CFFDD57AD8B9733912EE1F3095404D65AB630F4638FA1F40D4E99") ,
            ("FEClientMod.xml"                  ,
             "B3EB870C6B4206D11C921E70C6D058777A5F69FD1D9DEA8B6071759CAFCD2593") ,
            ("FEServerMod.xml"                  ,
             "55A881BFE436EF18C104BFA51ECF6D12583076D576BA3276F53A682E056ACA5C") ,
            ("header-image.png"                 ,
             "38F1E2D061218D31555F35C729197A32C9190999EF548BF98A2E2C2217BBCB88") ,
            ("MDT_LanguageUI.xml"               ,
             "100B5CA10BCF99E2A8680C394266042DEA5ECA300FBDA33289F6E4A17E44CBCF") ,
            ("MDTClientMod.xml"                 ,
             "C22C53DAAB87AAC06DC3AC64F66C8F6DF4B7EAE259EC5D80D60E51AF82055231") ,
            ("MDTServerMod.xml"                 ,
             "3724FE189D8D2CFBA17BC2A576469735B1DAAA18A83D1115169EFF0AF5D42A2F") ,
            ("PSDClientMod.xml"                 ,
             "4175C9569C8DFC1F14BADF70395D883BDD983948C2A6633CBBB6611430A872C7") ,
            ("PSDServerMod.xml"                 ,
             "4175C9569C8DFC1F14BADF70395D883BDD983948C2A6633CBBB6611430A872C7") ,
            ("success.png"                      ,
             "46757AB0E2D3FFFFDBA93558A34AC8E36F972B6F33D00C4ADFB912AE1F6D6CE2") ,
            ("vendorlist.txt"                   ,
             "9BD91057A1870DB087765914EAA5057D673CDC33145D804BBF4B024A11D66934") ,
            ("zipcode.txt"                      ,
             "45D5F4B9B50782CEC4767A7660583C68A6643C02FC7CC4F0AE5A79CCABE83021") | % {

                $This.Add(1,$_[0],$_[1])
            }

            # // _____________
            # // | Functions |
# // ------------

$This.AddFolder("Function","Functions")


("Copy-FileStream.ps1"              ,
 "F80662EF865682E3DF17EA8F30E31E3D0F1650C8DD5A129D4F8B9539F92A61B3") ,
("Get-AssemblyList.ps1"             ,
 "1610574E514AAF500FF8CEDCCF2B46EDF28287D9E3EFB612C3C0320320A4E7A3") ,
("Get-ControlExtension.ps1"         ,
 "8CC5D1320C51498AF2BE365F38949926331339E7CB6B3101C4A46FAD05CF2092") ,
("Get-DiskInfo.ps1"                 ,
 "1D68ED1AD277CCF0B860332C1501570B39C49B67D7F9AF7F309ADC9E99B409D0") ,
("Get-EnvironmentKey.ps1"           ,
 "C7C6D0D422A93F803F6F7539C42E057DA213661F9F2212679C6DCF10F5F3AA51") ,
("Get-EventLogArchive.ps1"          ,
 "E411B5B741F98B1F483B1F4E62DF0B64D536EB61ADE427D8793BEC1E99B51021") ,
("Get-EventLogConfigExtension.ps1"  ,
 "CE7FC970662A07DAED28F3E29FD2E449ED691315CB312039D0FDB61E0B587C45") ,
("Get-EventLogController.ps1"       ,
 "B3F1DB7A018A22E378637E170CA016F250572A2DA5113CF1AE3CC393A732091A") ,
("Get-EventLogProject.ps1"          ,
 "825C67A3409669DD26623F24714039613E746D6D7D4F777BEFB219B3307DDFA1") ,
("Get-EventLogRecordExtension.ps1"  ,
 "5722717C4A51D0069DD78FD31E72F239211BC26C10D9CFF5202659B50A026A56") ,
("Get-EventLogXaml.ps1"             ,
 "3D5BCF8F6FAAB73F2CF113C4778A7556C908C055D472E313224FD05BC470D48C") ,
("Get-FEADLogin.ps1"                ,
 "4297453E04EB27552ABD8C3C14104273C60F221A8248CEA1C65A4D30B99C7203") ,
("Get-FEDCPromo.ps1"                ,
 "06CD65C5C5ABDB7A5A625DE510C7CDA4FD9575E7976D7C65FA713714DDC01DFB") ,
("Get-FEHost.ps1"                   ,
 "02904EA751DB13D32FC18577D8780DE8B7E4ADB43EC94FC11621BA6CE5DC2488") ,
("Get-FEImageManifest.ps1"          ,
 "C45D927563C29E8E5AF173ACD635AED75A963582389D39B3A9D2CDF9E9849ECC") ,
("Get-FEInfo.ps1"                   ,
 "2C3E7209FFEE695E7187972B2AB0EF2B50CB5C8F89680ED1E9A14A388B376A59") ,
("Get-FEManifest.ps1"               ,
 "93CD40C06942BCCCBD67ECD950AA3B8F8D9A4162EAE1681C352CA20D7B6CC3F1") ,
("Get-FEModule.ps1"                 ,
 "12FD9079144EFABD7E0ECB923401CB3294C9642004B9259160712E963E99A89B") ,
("Get-FENetwork.ps1"                ,
 "88C28DF03BC1EC0E79E250D3496E6E9C6E26DFCBCEBC4EBA647AA1540BA8C438") ,
("Get-FEOS.ps1"                     ,
 "8C32694CE87CC1CCCB46E106C210F36B84EFCBF4D85C5240BB66D0C0840303BF") ,
("Get-FEProcess.ps1"                ,
 "0CB2B46E14790BA89FC2F60A12B67C9F1E435A8A20DBF011A130D26A291E094D") ,
("Get-FERole.ps1"                   ,
 "1E52DFB5820ACDC711D232DC18E5DCFCF390EF71721E2BE4DDADB885B675A529") ,
("Get-FEService.ps1"                ,
 "3D7F947ADCDCDF0A139AFFA9E92D125A581275BDC2B74247F437BA5A4985C2D6") ,
("Get-FESitemap.ps1"                ,
 "45A571D62EE528F05E0D4EA43995FB0B7C4A1DD7D1839A8D7EEB8754E8AB3009") ,
("Get-MadBomb.ps1"                  ,
 "87550BBEE679E62DF45F44F5BC871030B833FA9DCC8C9956AF422707444EAB68") ,
("Get-MDTModule.ps1"                ,
 "409B59C64ABEBAC3DE884954E40C433B6CDA3145A2EE2D82B503D0ABA1EDBE3D") ,
("Get-PowerShell.ps1"               ,
 "3E2C7F2FBCED55C73F393070F425AA6C66861EDA2D0CDE794F85BA962A3A0348") ,
("Get-PropertyItem.ps1"             ,
 "F9CFE6862B912B4D181A65FAF6BFBC1892ABBEED016FF14FAB3A8AD55B6C9151") ,
("Get-PropertyObject.ps1"           ,
 "A279B9F61B2633DB09D6D574D07B94A2C2D4429BC5DD539412E142F11AB49525") ,
("Get-PSDLog.ps1"                   ,
 "75F2F974CAE0153EB3987389A8EECD88255F58833273F84CC847C14BF80D3269") ,
("Get-PSDLogGUI.ps1"                ,
 "8716E3EC075E03E86BB28C495A359449445BC879F02F47AE5AEBCACCCE5BA679") ,
("Get-PSDModule.ps1"                ,
 "FCD86A877C9F8D5559E6849230AE41E169B31FEB197E0CF722C0CEA95B70CAAB") ,
("Get-SystemDetails.ps1"            ,
 "7B4713132FC595DC85A65286A370822A9F8A68897AD72432FCEC5385BF702EF1") ,
("Get-ThreadController.ps1"         ,
         "8196A7A298364599D72859E761FCD5ED370E99825283C46B83C10DE9E6ACD2DC") ,
        ("Get-ViperBomb.ps1"                ,
         "DF93D10B9C6ACEEEF21EC3AA6B0D32D1130900A363C3D654A529DF46B017541C") ,
        ("Get-WhoisUtility.ps1"             ,
         "9181508E7AE447FE317A50614FB83F1A4BD0B35490A0C5149F50A71D4C4AA451") ,
        ("Install-BossMode.ps1"             ,
         "2739086EB9BCDB520D0B20C17081EF5FB516C2E1387864CC38C9452EA16F0CC3") ,
        ("Install-IISServer.ps1"            ,
         "2D7DEEDB3F844183215609F72D63C24BA5B7C1D0D901120708172164EAA4A4E0") ,
        ("Install-PSD.ps1"                  ,
         "989B34030F75F0A6EFACC574361C170B2D51C6F5FA031032ECCE29119EC3B5A4") ,
        ("Invoke-cimdb.ps1"                 ,
         "4852D60255F5F2715703A38CF82C98B159B588E1B7A1BEA9D0E9AE2EC7530190") ,
        ("Invoke-KeyEntry.ps1"              ,
         "B1300999BF1A6ABEEDCFDEC1B0C150228D7FE03623D22DBCEBBC18C3BAF6C134") ,
        ("New-EnvironmentKey.ps1"           ,
         "A06CFAEAA6DCE65C6E3C6168A3AA2AD9230A81D16706BA82E8D89B0CD376BBE9") ,
        ("New-FEInfrastructure.ps1"         ,
         "966B36D105A6E02299F47B425A32612F17FB2AD16CEC68726D9D6006371206B0") ,
        ("Search-WirelessNetwork.ps1"       ,
         "AF3D312ECA04C87103D5F921F0D35B5B3C3B34EE83E571AA1594DA7C17ECFF5D") ,
        ("Set-ScreenResolution.ps1"         ,
         "FFF86F4CD863BBC59168BDD821362B274C3723A888896F225F6EF04DF5D7C32E") ,
        ("Show-ToastNotification.ps1"       ,
         "0002209685C3D83A4D08E4265B9285DEDD71C381B6EB8A8F7D86F4E949927969") ,
        ("Update-PowerShell.ps1"            ,
         "446878FCADA300B44691053ABF02FF96772B5FCE1A5434FB61A81FE3C1B416E4") ,
        ("Use-Wlanapi.ps1"                  ,
         "1113CEC8BE5E352B09698928995ED840B5EE7A3F90DE1A5537DF339E7D10E5FF") ,
        ("Write-Theme.ps1"                  ,
         "3B385DF0CC0E44AB1C6991A115B0466463AB3FDFA81371278B0D05028AA62703") | % {

            $This.Add(2,$_[0],$_[1])
        }

        # // ____________
        # // | Graphics |
        # // ------------

        $This.AddFolder("Graphic","Graphics")

        ("background.jpg"                   ,
         "94FD6CB32F8FF9DD360B4F98CEAA046B9AFCD717DA532AFEF2E230C981DAFEB5") ,
        ("banner.png"                       ,
         "057AF2EC2B9EC35399D3475AE42505CDBCE314B9945EF7C7BCB91374A8116F37") ,
        ("icon.ico"                         ,
         "594DAAFF448F5306B8B46B8DB1B420C1EE53FFD55EC65D17E2D361830659E58E") ,
        ("OEMbg.jpg"                        ,
         "D4331207D471F799A520D5C7697E84421B0FA0F9B574737EF06FC95C92786A32") ,
        ("OEMlogo.bmp"                      ,
         "98BF79CAE27E85C7722564A3113C52D1E75BD6328398871873072F6B363D1A8") ,
        ("PSDBackground.bmp"                ,
         "05ABBABDC9F67A95D5A4AF466149681C2F5E8ECD68F11433D32F4C0D04446F7E") ,
        ("sdplogo.png"                      ,
         "87C2B016401CA3F8F8FAD5F629AFB3553C4762E14CD60792823D388F87E2B16C") | % {

            $This.Add(3,$_[0],$_[1])
        }

        $This.Total = ($This.Output | % Item).Count
        $This.Depth = ([String]$This.Total).Length
    }
    Add([UInt32]$Index,[String]$Name,[String]$Hash)
    {
        $This.Output[$Index] | % {

            $_.Add($Name,$Hash)
            $_.Item[-1].SetSource($This.Source)
        }
    }
    AddFolder([String]$Type,[String]$Name)
    {
        $This.Output += [Folder]::New($This.Output.Count,$Type,$This.Resource,$Name)
    }
    [String] Status([UInt32]$Rank)
    {
        Return "({0:d$($This.Depth)}/{1})" -f ($Rank+1), $This.Total
    }
    [String] Percent([UInt32]$Rank)
    {
        Return "{0:n2}" -f (($Rank/$This.Total) * 100)
    }
    Refresh()
    {
        $This.Output | % { $_.TestPath(); $_.Item | % TestPath }
    }
    Install()
    {
        $This.Refresh()

        $This.Output | ? Exists -eq 0 | % Create

        $List = $This.Output | % Item
        ForEach ($X in 0..($List.Count-1))
        {
            $File = $List[$X]
            $File.TestPath()
            If (!$File.Exists)
            {
                $File.Create()
                $File.Download()
                $File.Write()
                $File.TestPath()
            }
            Write-Host ("Installed [~] {0} {1}% -> {2}" -f $This.Status($X),
                                                           $This.Percent($X),
                                                           $File.Name)
        }
    }
    Remove()
    {
        $This.Refresh()

        $List  = $This.Output | % Item
        ForEach ($X in 0..($List.Count-1))
        {
            $File = $List[$X]
            $File.TestPath()
            If ($File.Exists)
            {
                $File.Delete()
                $File.TestPath()
            }
            Write-Host ("Removed [+] {0} {1:n2}% -> {2}" -f $This.Status($X),
                                                            $This.Percent($X),
                                                            $File.Name)
        }

        $This.Output | ? Exists -eq 1 | % Delete
    }
    [Object] List()
    {
        Return @(ForEach ($Folder in $This.Output)
        {
            $Folder
            $Folder | % Item
        })
    }
    [Object] Files([UInt32]$Index)
    {
        Return $This.Output[$Index] | % Item
    }
    [String] ToString()
    {
        Return "<FightingEntropy.Module.Manifest>"
    }
}
```

```
                                                                                     ___________________________________/
\___________________________________________________________________________________/ <FightingEntropy.Module.Manifest>
  <FightingEntropy.Module.Template> /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

```
    # // ___________________________________
    # // | Template for registry injection |
    # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

    Class Template
    {
        [String]      $Source
        [String]        $Name
        [String] $Description
        [String]      $Author
        [String]     $Company
        [String]   $Copyright
        [Guid]          $Guid
        [DateTime]      $Date
        [String]     $Caption
        [String]    $Platform
        [String]        $Type
        [String]    $Registry
        [String]    $Resource
        [String]      $Module
        [String]        $File
        [String]    $Manifest
        Template([Object]$Module)
        {
            $This.Source      = $Module.Source
            $This.Name        = $Module.Name
            $This.Description = $Module.Description
            $This.Author      = $Module.Author
            $This.Company     = $Module.Company
            $This.Copyright   = $Module.Copyright
            $This.Guid        = $Module.Guid
            $This.Date        = $Module.Date
            $This.Caption     = $Module.OS.Caption
            $This.Platform    = $Module.OS.Platform
            $This.Type        = $Module.OS.Type
            $This.Registry    = $Module.Root.Registry
            $This.Resource    = $Module.Root.Resource
            $This.Module      = $Module.Root.Module
            $This.File        = $Module.Root.File
            $This.Manifest    = $Module.Root.Manifest
        }
    }
```

```
                                                                                     ___________________________________/
\___________________________________________________________________________________/ <FightingEntropy.Module.Template>
  <FightingEntropy.Module.RootProperty> /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

```
    # // ___________________________________________________
    # // | Represents individual paths to the module root |
    # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

    Class RootProperty
    {
        [String] $Type
        [String] $Name
        [String] $Fullname
        [UInt32] $Exists
        Hidden [String] $Path
        RootProperty([String]$Name,[UInt32]$Type,[String]$Fullname)
        {
            $This.Type     = Switch ($Type) { 0 { "Directory" } 1 { "File" } }
            $This.Name     = $Name
            $This.Fullname = $Fullname
            $This.Path     = $Fullname
            $This.TestPath()
        }
        TestPath()
        {
            $This.Exists   = Test-Path $This.Path
        }
        Create()
        {
            $This.TestPath()

            If (!$This.Exists)
            {
                Switch -Regex ($This.Name)
                {
                    "(Resource|Module)"
                    {
                        [System.IO.Directory]::CreateDirectory($This.Fullname)
                    }
                    "(File|Manifest)"
                    {
                        [System.IO.File]::Create($This.Fullname).Dispose()
                    }
                }

                $This.TestPath()
            }
        }
        Remove()
        {
            $This.TestPath()

            If ($This.Exists)
            {
                Switch -Regex ($This.Name)
                {
                    "(Resource|Module)"
                    {
                        [System.IO.Directory]::Delete($This.Fullname)
                    }
                    "(File|Manifest)"
                    {
                        [System.IO.File]::Delete($This.Fullname)
                    }
                }
                $This.Exists = 0
            }
        }
        [String] ToString()
        {
            Return $This.Path
        }
    }
```

```
                                                                                  ________________________________________/
\_________________________________________________________________________________/ <FightingEntropy.Module.RootProperty>
  <FightingEntropy.Module.Root> /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

```
    # // ________________________________________________________
    # // | Represents a collection of paths for the module root |
    # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

    Class Root
    {
        [Object] $Registry
        [Object] $Resource
        [Object]   $Module
        [Object]     $File
        [Object] $Manifest
        [Object] $Shortcut
        Root([String]$Version,[String]$Resource,[String]$Path)
        {
            $SDP           = "Secure Digits Plus LLC"
            $FE            = "FightingEntropy"
            $This.Registry = $This.Set(0,0,"HKLM:\Software\Policies\$SDP\$FE\$Version")
            $This.Resource = $This.Set(1,0,"$Resource")
            $This.Module   = $This.Set(2,0,"$Path\$FE")
            $This.File     = $This.Set(3,1,"$Path\$FE\$FE.psm1")
            $This.Manifest = $This.Set(4,1,"$Path\$FE\$FE.psd1")
            $This.Shortcut = $This.Set(5,1,"$Env:Public\Desktop\$FE.lnk")
        }
        [String] Slot([UInt32]$Type)
        {
            Return @("Registry","Resource","Module","File","Manifest","Shortcut")[$Type]
        }
        [Object] Set([UInt32]$Index,[UInt32]$Type,[String]$Path)
        {
            Return [RootProperty]::New($This.Slot($Index),$Type,$Path)
        }
        [Void] Refresh()
        {
            $This.List() | % { $_.TestPath() }
        }
        [Object[]] List()
        {
            Return $This.PSObject.Properties.Name | % { $This.$_ }
        }
        [String] ToString()
        {
            Return "<FightingEntropy.Module.Root>"
        }
    }
```

```
                                                                                                      _______________________________/
\_____________________________________________________________________________________________________/ <FightingEntropy.Module.Root>
  <FightingEntropy.Module.RegistryKeyTemp> /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

```
    # // ________________________________________
    # // | Works as a PowerShell Registry provider |
    # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

    Class RegistryKeyTemp
    {
        Hidden [Microsoft.Win32.RegistryKey] $Key
        Hidden [Microsoft.Win32.RegistryKey] $Subkey
        [String]            $Enum
        [String]            $Hive
        [String]            $Path
        [String]            $Name
        Hidden [String] $Fullname
        RegistryKeyTemp([String]$Path)
        {
            $This.Fullname = $Path
            $Split         = $Path -Split "\\"
            $This.Hive     = $Split[0]
            $This.Name     = $Split[-1]
            $This.Enum     = Switch -Regex ($This.Hive)
            {
                HKLM: {"LocalMachine"} HKCU: {"CurrentUser"} HKCR: {"ClassesRoot"}
            }
            $This.Path     = $Path -Replace "$($This.Hive)\\", "" | Split-Path -Parent
        }
        Open()
        {
            $X              = $This.Enum
            $This.Key       = [Microsoft.Win32.Registry]::$X.CreateSubKey($This.Path)
        }
        Create()
        {
            If (!$This.Key)
            {
                Throw "Must open the key first."
            }

            $This.Subkey = $This.Key.CreateSubKey($This.Name)
            Write-Host "Registry [+] Path: [$($This.Fullname)]"
        }
        Add([String]$Name,[Object]$Value)
        {
            If (!$This.Subkey)
            {
                Throw "Must create the subkey first."
            }

            $This.Subkey.SetValue($Name,$Value)
            Write-Host "Key [+] Property: [$Name], Value: [$Value]"
        }
        [Void] Delete()
        {
            If ($This.Key)
            {
                $This.Key.DeleteSubKeyTree($This.Name)
                Write-Host "Registry [-] Path [$($This.Fullname)"
            }
        }
        [Void] Dispose()
        {
            If ($This.Subkey)
            {
                $This.Subkey.Flush()
                $This.Subkey.Dispose()
            }

            If ($This.Key)
            {
                $This.Key.Flush()
                $This.Key.Dispose()
            }
        }
    }
```

```
                                                                                   ___________________________________________/
\_________________________________________________________________________________/ <FightingEntropy.Module.RegistryKeyTemp>
  <FightingEntropy.Module.RegistryKeyProperty> /---------------------------------------------------------------------------\
/---------------------------------------------
```

```
    # // ________________________________________________________
    # // | Represents an individual registry key for the module |
    # // --------------------------------------------------------

    Class RegistryKeyProperty
    {
        Hidden [UInt32] $Index
        [String] $Name
        [Object] $Value
        [UInt32] $Exists
        RegistryKeyProperty([UInt32]$Index,[String]$Name,[Object]$Value)
        {
            $This.Index = $Index
            $This.Name  = $Name
            $This.Value = $Value
        }
        [String] ToString()
        {
            Return "<FightingEntropy.Module.RegistryKeyProperty>"
        }
    }
```

```
                                                                          ______________________________________________/
\________________________________________________________________________/ <FightingEntropy.Module.RegistryKeyProperty>
  <FightingEntropy.Module.RegistryKey> /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

```
    # // ___________________________________________________________
    # // | Represents a collection of registry keys for the module |
    # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

    Class RegistryKey
    {
        [String] $Path
        [UInt32] $Exists
        [Object] $Property
        RegistryKey([Object]$Module)
        {
            $This.Path          = $Module.Root.Registry.Path
            $This.TestPath()
            If ($This.Exists)
            {
                $Object        = Get-ItemProperty $This.Path
                $This.Property = $This.Inject($Object)
            }
            Else
            {
                $Object        = $Module.Template()
                $This.Property = $This.Inject($Object)
            }
        }
        [Object] Inject([Object]$Object)
        {
            $Hash              = @{ }
            $Object.PSObject.Properties | ? Name -notmatch ^PS | % {

                $Item          = $This.Key($Hash.Count,$_.Name,$_.Value)
                $Item.Exists   = $This.Exists
                $Hash.Add($Hash.Count,$Item)
            }

            Return $Hash[0..($Hash.Count-1)]
        }
        TestPath()
        {
            $This.Exists = Test-Path $This.Path
        }
        [String] Status([UInt32]$Rank)
        {
            $D = ([String]$This.Property.Count).Length
            Return "({0:d$D}/{1})" -f $Rank, $This.Property.Count
        }
        Install()
        {
            $This.TestPath()

            If ($This.Exists)
            {
                Throw "Exception [!] Path already exists"
            }

            $Key             = $This.RegistryKeyTemp($This.Path)
            $Key.Open()
            $Key.Create()
            $This.Exists    = 1

            ForEach ($X in 0..($This.Property.Count-1))
            {
                $Item        = $This.Property[$X]
                $Key.Add($Item.Name,$Item.Value)
                $Item.Exists = 1
            }
            $Key.Dispose()
        }
        Remove()
        {
            $This.TestPath()

            If (!$This.Exists)
            {
                Throw "Exception [!] Registry path does not exist"
            }

            $Key             = $This.RegistryKeyTemp($This.Path)
            $Key.Open()
            $Key.Create()
            $Key.Delete()

            ForEach ($Item in $This.Property)
            {
                $Item.Exists = 0
            }

            $This.Exists     =   0
            $Key.Dispose()
        }
        [Object[]] List()
        {
            Return $This.Output
        }
        [Object] Key([UInt32]$Index,[String]$Name,[Object]$Value)
        {
            Return [RegistryKeyProperty]::New($Index,$Name,$Value)
        }
        [Object] RegistryKeyTemp([String]$Path)
        {
            Return [RegistryKeyTemp]::New($Path)
        }
        [String] ToString()
        {
            Return "<FightingEntropy.Module.RegistryKey>"
        }
    }
```

```
                                                                                  ______________________________________/
\_________________________________________________________________________________/ <FightingEntropy.Module.RegistryKey>
  <FightingEntropy.Module.FEVersion> /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

```
    # // ___________________________________________
    # // | Collects/creates versions of the module |
    # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

    Class FEVersion
    {
        [Version]      $Version
        Hidden [DateTime] $Time
        [String]          $Date
        [Guid]            $Guid
        FEVersion([String]$Line)
        {
            $This.Version = $This.Tx(0,$Line)
            $This.Time    = $This.Tx(1,$Line)
            $This.Date    = $This.MilitaryTime()
            $This.Guid    = $This.Tx(2,$Line)
        }
        FEVersion([Switch]$New,[String]$Version)
        {
            $This.Version = $Version
            $This.Time    = [DateTime]::Now
            $This.Date    = $This.MilitaryTime()
            $This.Guid    = [Guid]::NewGuid()
        }
        [String] MilitaryTime()
        {
            Return $This.Time.ToString("MM/dd/yyyy HH:mm:ss")
        }
        [String] Tx([UInt32]$Type,[String]$Line)
        {
            $Pattern = Switch ($Type)
            {
                0 { "\d{4}\.\d{2}\.\d+" }
                1 { "\d{2}\/\d{2}\/\d{4} \d{2}:\d{2}:\d{2}" }
                2 { @(8,4,4,4,12 | % { "[a-f0-9]{$_}" }) -join '-' }
            }

            Return [Regex]::Matches($Line,$Pattern).Value
        }
        [String] ToString()
        {
            Return "| {0} | {1} | {2} |" -f $This.Version,
                                            $This.Date.ToString("MM/dd/yyyy HH:mm:ss"),
                                            $This.Guid
        }
    }
```

```
                                                                                     ____________________________________/
\___________________________________________________________________________________/ <FightingEntropy.Module.FEVersion>
  <FightingEntropy.Module.ValidateFile> /‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾\
/‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾
```

```
# // ________________________________________________________
# // | Specifically used for file hash validation/integrity |
# // ‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾

Class ValidateFile
{
    [String] $Type
    [String] $Name
    Hidden [String] $Fullname
    Hidden [String] $Source
    [String] $Hash
    [UInt32] $Match
    [String] $Compare
    ValidateFile([String]$Leaf,[Object]$File)
    {
        $This.Type     = $File.Type
        $This.Name     = $File.Name
        $This.Fullname = $File.Fullname
        $This.Hash     = $File.Hash
        $This.Source   = $File.Source

        # // _______________________
        # // | Temporary variables |
        # // ‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾

        $Content       = Invoke-WebRequest $This.Source -UseBasicParsing | % Content
        $Target        = "{0}\{1}" -f $Env:Temp, $This.Name

        If ([System.IO.File]::Exists($Target))
        {
            [System.IO.File]::Delete($Target)
        }
```

```powershell
        If ($This.Name -match "\.+(jpg|jpeg|png|bmp|ico)")
        {
            [System.IO.File]::WriteAllBytes($Target,[Byte[]]$Content)
        }
        Else
        {
            [System.IO.File]::WriteAllText($Target,$Content,[System.Text.UTF8Encoding]$False)
        }

        # // ________________________________________
        # // | Get target hash and final comparison |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        $This.Compare  = $This.GetFileHash($Target)
        $This.Match    = $This.Hash -eq $This.Compare

        [System.IO.File]::Delete($Target)
    }
    [String] GetFileHash([String]$Path)
    {
        If (![System.IO.File]::Exists($Path))
        {
            Throw "Invalid path"
        }

        Return Get-FileHash $Path | % Hash
    }
}
```

```
                                                                                 _______________________________________/
\_______________________________________________________________________________/ <FightingEntropy.Module.ValidateFile>
  <FightingEntropy.Module.Validate> /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

```powershell
# // __________________________________________________
# // | Container class for (manifest/file) validation |
# // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

Class Validate
{
    [Object] $Output
    Validate([Object]$Module)
    {
        $Hash     = @{ }
        ForEach ($Branch in $Module.Manifest.Output)
        {
            Write-Host ("Path [~] [{0}]" -f $Branch.Fullname)
            ForEach ($File in $Branch.Item)
            {
                Write-Host "File [~] [$($File.Fullname)]"
                $Hash.Add($Hash.Count,$This.ValidateFile($Branch.Name,$File))
            }
        }

        $This.Output = $Hash[0..($Hash.Count-1)]
    }
    [Object] ValidateFile([String]$Name,[Object]$File)
    {
        Return [ValidateFile]::New($Name,$File)
    }
    [String] BuildManifest()
    {
        $MaxName = ($This.Output.Name | Sort-Object Length)[-1]
        Return @( $This.Output | % {

            "            (`"{0}`"{1}, `"{2}`") ," -f $_.Name,
            (@(" ") * ($MaxName.Length - $_.Name.Length + 1) -join ''),
            $_.Hash
        }) -join "`n"
    }
}
```

```
                                                                                                  ___________________________________/
\_________________________________________________________________________________________________/ <FightingEntropy.Module.Validate>
  <FightingEntropy.Module.Main> /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

```
    # // ______________________________________________________________
    # // | Factory class to control all of the aforementioned classes |
    # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

    Class Main
    {
        [String]      $Source = "https://www.github.com/mcc85s/FightingEntropy"
        [String]        $Name = "[FightingEntropy(π)]"
        [String] $Description = "Beginning the fight against ID theft and cybercrime"
        [String]      $Author = "Michael C. Cook Sr."
        [String]     $Company = "Secure Digits Plus LLC"
        [String]   $Copyright = "(c) 2022 (mcc85s/mcc85sx/sdp). All rights reserved."
        [Guid]          $Guid = "b139e090-db90-4536-95e8-91ea49ab74a9"
        [DateTime]      $Date = "10/27/2022 20:00:08"
        [Version]    $Version = "2022.10.1"
        [Object]          $OS
        [Object]        $Root
        [Object]    $Manifest
        [Object]    $Registry
        Main()
        {
            $This.Write("Loading [~] $($This.Label())")

            $This.OS       = $This.GetOS()
            Write-Host "[+] Operating System"

            $This.Root     = $This.GetRoot()
            Write-Host "[+] Module Root"

            $This.Manifest = $This.GetManifest($This.Source,$This.Root.Resource)
            Write-Host "[+] Module Manifest"

            $This.Registry = $This.GetRegistry()
            Write-Host "[+] Module Registry"
        }
        [Object] NewVersion([String]$Version)
        {
            If ($Version -notmatch "\d{4}\.\d{2}\.\d+")
            {
                Throw "Invalid version entry"
            }

            Return [FEVersion]::New($True,$Version)
        }
        [Object[]] Versions()
        {
            $MD       = Invoke-RestMethod "$($This.Source)/blob/main/README.md?raw=true"
            Return [FEVersion[]]($MD -Split "`n" -match "\d{4}\.\d{2}\.\d+")
        }
        [String] Label()
        {
            Return "{0}[{1}]" -f $This.Name, $This.Version.ToString()
        }
        [Object] Template()
        {
            Return [Template]::New($This)
        }
        [Object] GetOS()
        {
            Return [OS]::New()
        }
```

```
[Object] GetRoot()
{
    $Resource = $Env:ProgramData,
                $This.Company,
                "FightingEntropy",
                $This.Version.ToString() -join "\"
    $Path     = Switch -Regex ($This.OS.Type)
    {
        ^Win32_ { $Env:PSModulePath -Split ";" -match [Regex]::Escape($Env:Windir) }
        Default { $Env:PSModulePath -Split ":" -match "PowerShell"                 }
    }

    Return [Root]::New($This.Version,$Resource,$Path)
}
[Object] GetManifest([String]$Source,[String]$Resource)
{
    Return [Manifest]::New($Source,$Resource)
}
[Object] GetRegistry()
{
    Return [RegistryKey]::New($This)
}
[Void] Write([String]$Message)
{
    [ThemeStack]::New($Message)
}
[Void] Write([UInt32]$Slot,[String]$Message)
{
    [ThemeStack]::New($Slot,$Message)
}
[Object] File([String]$Type,[String]$Name)
{
    Return $This.Manifest.List() | ? Type -eq $Type | ? Name -eq $Name
}
[Object] Class([String]$Name)
{
    Return $This.File("Class",$Name)
}
[Object] Control([String]$Name)
{
    Return $This.File("Control",$Name)
}
[Object] Function([String]$Name)
{
    Return $This.File("Function",$Name)
}
[Object] Graphic([String]$Name)
{
    Return $This.File("Graphic",$Name)
}
[Void] Refresh()
{
    # // ____________________________________________
    # // | Tests all manifest (folder/file) entries |
    # // --------------------------------------------

    $This.Manifest.Output | % { $_.TestPath(); $_.Item | % TestPath }

    $This.Registry.TestPath()
    If ($This.Registry.Exists)
    {
        $This.Root.Registry.Exists = 1
    }
    $This.Root.Manifest.TestPath()
    $This.Root.File.TestPath()
    $This.Root.Module.TestPath()
}
[Void] Remove()
{
    $This.Write(1,"Removing [~] $($This.Label())")

    # // ___________________________________________
```

```
        # // | Removing [Module]: (Manifest/File/Path) |
        # // -------------------------------------------

        "Shortcut","Manifest","File","Module" | % {

            $Item = $This.Root.$_
            $Item.Remove()
            Write-Host "Removed [+] $_ | $($Item.Fullname)"
        }

        # // ________________________________________________
        # // | Removing [Manifest/Registry]: (Content/Path) |
        # // ------------------------------------------------

        "Manifest","Registry" | % {

            Write-Host "Removing [~] $_"
            $This.$_.Remove()
            Write-Host "Removed [+] $_"
        }

        $This.Write(1,"Removed [+] $($This.Label())")
    }
    [Void] Install()
    {
        $This.Write(2,"Installing [~] $($This.Label())")

        $This.Manifest.Install()
        $This.Registry.Install()
        $This.Root.Module.Create()
        $This.Root.File.Create()

        # // ___________________
        # // | PowerShell Full |
        # // -------------------

        If ($This.Root.Resource.Exists)
        {
            # // ______________________________
            # // | Cobble together assemblies |
            # // ------------------------------

            $Bin = "PresentationFramework",
                   "System.Runtime.WindowsRuntime",
                   "System.IO.Compression",
                   "System.IO.Compression.Filesystem",
                   "System.Windows.Forms"

            # // _____________________________________________
            # // | Write the module file to disk using PSM() |
            # // ---------------------------------------------

            [System.IO.File]::WriteAllLines($This.Root.File,
                                            $This.PSM($Bin),
                                            [System.Text.UTF8Encoding]$False)

            # // ____________________________________
            # // | Splat the Module Manifest params |
            # // ------------------------------------

            $Splat = $This.PSDParam($Bin)

            # // ________________________________________________
            # // | Write the PowerShell module manifest to disk |
            # // ------------------------------------------------

            New-ModuleManifest @Splat

            $This.Root.Manifest.TestPath()
        }

        # // ___________________________________________________________________
        # // | Todo | PS Core | PS Server | <- Just a manner of file selection |
        # // -------------------------------------------------------------------

        # // _____________________________________________
        # // | Installs a shortcut to the module console |
        # // ---------------------------------------------

        $Com  = New-Object -ComObject WScript.Shell
        $Item = $Com.CreateShortcut($This.Root.Shortcut.Path)

        # // ___________________________________
        # // | Assigns details to the shortcut |
        # // -----------------------------------

        $Item.TargetPath   = "PowerShell"

        $Command           = 'Add-Type -AssemblyName PresentationFramework',
                             'Import-Module FightingEntropy',
                             '$Module = Get-FEModule',
                             '$Module' -join ";"
        $Item.Arguments    = "-NoExit -ExecutionPolicy Bypass -Command $Command"

        $Item.Description  = $This.Description
        $Item.IconLocation = $This.Graphic("icon.ico").Fullname
        $Item.Save()

        # // _____________________________________________________
        # // | Assigns administrative privileges to the shortcut |
        # // -----------------------------------------------------

        $Bytes             = [System.IO.File]::ReadAllBytes($This.Root.Shortcut)
        $Bytes[0x15]       = $Bytes[0x15] -bor 0x20
                             # Set [byte] (21/0x15) bit 6 (0x20) ON... or else.
        [System.IO.File]::WriteAllBytes($This.Root.Shortcut, $Bytes)

        $This.Root.Shortcut.TestPath()

        $This.Write(2,"Installed [+] $($This.Label())")
    }
    [String] PSM([String[]]$Bin)
    {
        $F  = @( )

        # // __________
        # // | Header |
        # // ----------

        $F += "# Downloaded from {0}" -f $This.Source
        $F += "# {0}" -f $This.Resource
        $F += "# {0}" -f $This.Version.ToString()
        $F += "# <Types>"
        $Bin | % { $F += "Add-Type -AssemblyName $_" }

        # // ____________________________________________
        # // | Classes (To be phased out at some point) |
        # // --------------------------------------------

        $F += "# <Classes>"
        $This.Manifest.Files(0) | % {

            $F += "# <{0}/{1}>" -f $_.Type, $_.Name
            $F += "# {0}" -f $_.Fullname
            If (!$_.Content)
            {
                $_.GetContent()
            }
            $F += $_.Content
            $F += "# </{0}/{1}>" -f $_.Type, $_.Name
        }
        $F += "# </Classes>"

        # // _____________
```

```
            # // | Functions |
            # // -------------

            $F += "# <Functions>"
            $This.Manifest.Files(2)  | % {

                $F += "# <{0}/{1}>" -f $_.Type, $_.Name
                $F += "# {0}" -f $_.Fullname
                If (!$_.Content)
                {
                    $_.GetContent()
                }
                $F += $_.Content
                $F += "# </{0}/{1}>" -f $_.Type, $_.Name
            }
            $F += "# </Functions>"
            $F += "Write-Theme `"Module [+] [$($This.Label())]`" @(10,3,15,0)"

            Return $F -join "`n"
        }
        [Hashtable] PSDParam([String[]]$Bin)
        {
            Return @{

                GUID                 = $This.GUID
                Path                 = $This.Root.Manifest
                ModuleVersion        = $This.Version
                Copyright            = $This.Copyright
                CompanyName          = $This.Company
                Author               = $This.Author
                Description          = $This.Description
                RootModule           = $This.Root.File
                RequiredAssemblies   = $Bin
            }
        }
        [Object] Validation()
        {
            $This.Write(3,"Validation [~] Module manifest")

            $Validate = [Validate]::New($This)
            $Ct       = $Validate.Output | ? Match -eq 0

            Switch ($Ct.Count)
            {
                {$_ -eq 0}
                {
                    $This.Write(3,"Validation [+] All files passed validation")
                }
                {$_ -gt 0}
                {
                    $This.Write(1,"Validation [!] ($($Ct.Count)) files failed validation")
                    $Ct
                }
            }

            Return $Validate
        }
    }
```

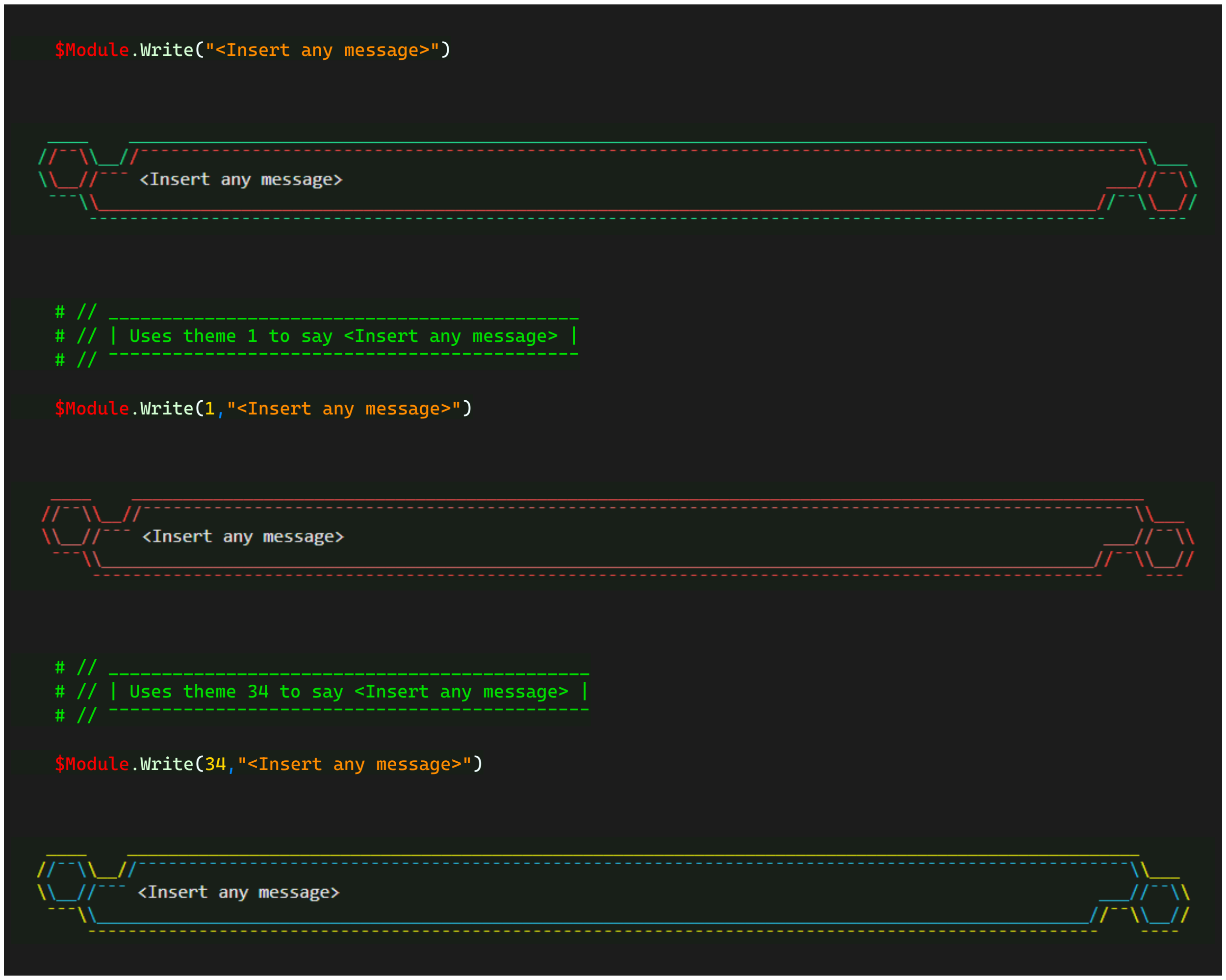
```
                                                                                          _______________________________/
\________________________________________________________________________________________/ <FightingEntropy.Module.Main>
  Function FightingEntropy.Module /--------------------------------------------------------------------------------------\
/---------------------------------
```

I'm not gonna lie.
This is a lot of work, to put this thing together into these files to document where the process 1) IS,
2) what it DOES, and that what is shown 3) WORKS about as well as it 4) LOOKS.

Ya know...? Pretty tall order. Then I gotta make it really (simple/easy to understand). And if I don't succeed at doing that at the tail end of it all...? Then, basically I suck at life...? And, I may as well wave a white flag, surrender, and just... give up. Cause. The challenge was WAY too difficult to complete. End of story.

Yeah, I'm not looking to give up, but... it's a lot harder than it looks.
Here's the final piece of the puzzle.

```
# // ___________________________________________________________________
# // | Installs the PowerShell module, [FightingEntropy(π)][2022.10.1] |
# // -------------------------------------------------------------------

Function FightingEntropy.Module
{

    # // __________________________________________________________
    # // | < ... where all of the above classes actually go ... > |
    # // ----------------------------------------------------------

    # // _______________________________________________________
    # // | Single class that controls all of the above classes |
    # // -------------------------------------------------------

    [Main]::New()
}

$Module = FightingEntropy.Module
#-----1   ---------------------2

# // _________________________________________________________________________________________
# // |---------------------------------------------------------------------------------------|
# // | 1) Single variable that allows all of the methods and properties within EACH class    |
# // |    to be accessible.                                                                  |
# // | 2) Single function being cast to that single variable allowing anybody who uses it,   |
# // |    to essentially be as powerful as god.                                              |
# // |_______________________________________________________________________________________|
# // -----------------------------------------------------------------------------------------

# // ________________________________________________________________________________
# // |------------------------------------------------------------------------------|
# // | Just kidding... nobody can be THAT powerful.                                 |
# // | Imagine if it WAS that powerful, though...?                                  |
# // |                                                                              |
# // | Like, imagine having access to using 'IDDQD' in Doom, but- in real life.     |
# // | The monsters would literally spend the rest of eternity making noises, cause |
# // | they're evil and they see you. So, they'll never stop attacking you.         |
# // |                                                                              |
# // | But- oh boy.                                                                 |
# // | They can't do any damage... cause you're god.                                |
# // | ...in the game.                                                              |
# // | Not real life.                                                               |
# // |______________________________________________________________________________|
# // --------------------------------------------------------------------------------

# // ______________________
# // | Installs the module |
# // ----------------------

$Module.Install()

# // _____________________
# // | Removes the module |
# // ---------------------

$Module.Remove()

# // _____________________________
# // | Validates all of the files |
# // -----------------------------

$Module.Validation()

# // _______________________________________________________
# // | Uses the default theme to say: <Insert any message> |
# // -------------------------------------------------------
```



```
$Module.Write("<Insert any message>")


 ____    ____________________________________________________________________________________________
//¯¯\\__//¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\\___
\\__//¯¯¯ <Insert any message>                                                                   ___//¯¯\\
 ¯¯¯\\___________________________________________________________________________________________//¯¯\\__//
     ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯    ¯¯¯¯


# // ____________________________________________
# // | Uses theme 1 to say <Insert any message> |
# // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

$Module.Write(1,"<Insert any message>")


 ____    ____________________________________________________________________________________________
//¯¯\\__//¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\\___
\\__//¯¯¯ <Insert any message>                                                                   ___//¯¯\\
 ¯¯¯\\___________________________________________________________________________________________//¯¯\\__//
     ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯    ¯¯¯¯


# // _____________________________________________
# // | Uses theme 34 to say <Insert any message> |
# // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

$Module.Write(34,"<Insert any message>")


 ____    ____________________________________________________________________________________________
//¯¯\\__//¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\\___
\\__//¯¯¯ <Insert any message>                                                                   ___//¯¯\\
 ¯¯¯\\___________________________________________________________________________________________//¯¯\\__//
     ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯    ¯¯¯¯
```

```
                                                                                                                    ____________/
\__________________________________________________________________________________________________________________/ Conclusion
```

```
 ______________________________________________________
|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|
|                                  Michael C. Cook Sr. |
|                                    Security Engineer |
|                                 Secure Digits Plus LLC |
|______________________________________________________|
 ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```