```
  ____    ____________________________________________________________________________________________________
 //¯¯\\__//¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\\___
 \\__//¯¯¯ Get-CiscoMerakiIpAddress                                                                     ___//¯¯\\
  ¯¯¯\\_____________________________________________________________________________________________//¯¯\\__//
      ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯    ¯¯¯¯
```

```
\________________________________________________________________________________________________________________________/
  Start /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯
```

Develop several classes that reproduces the input/output of this Market 32 Cisco Meraki authentication hyperlink.

https://n137.network-auth.com/splash/?mac=88%3A15%3A44%3AA3%3AB7%3A10&real_ip=192.168.0.81&client_ip=10.201.240.180&client_mac=9C:B7:0D:20:08:FE&vap=0&a=a17554a0d2b15a664c0e7390018454f19e70227&b=17468474&auth_version=5&key=834c46c40a4248fae0dec59501aef3f0327e6738&acl_ver=P4903858V2&continue_url=http%3A%2F%2Fwww.msftconnecttest.com%2Fredirect

If you would like to refer to the actual SCRIPT that is exhibited below...?

Cisco Meraki IP	https://github.com/mcc85s/FightingEntropy/blob/main/Scripts/Get-CiscoMerakiIPAddress.ps1

```
                                                                                                                  _______/
\________________________________________________________________________________________________________________/ Start
  Overview /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯
```

That's such a LONG URL that there's really next to no way for a HUMAN to be able to REMEMBER all of that info.
Let's break it down. Cast it to a variable, $String

```
# // ____________________________________________________________________
# // | Get the Cisco Meraki IP Address from the wireless access point, |
# // | or else we won't receive any birthday party invitations          |
# // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

$String = ( "https://n137.network-auth.com/splash/?mac=88%3A15%3A44%3AA3%3AB7"+
           "%3A10&real_ip=192.168.0.81&client_ip=10.201.240.180&client_mac=9"+
           "C:B7:0D:20:08:FE&vap=0&a=a17554a0d2b15a664c0e7390018454f19e7022"+
           "7&b=17468474&auth_version=5&key=834c46c40a4248fae0dec59501aef3f0"+
           "327e6738&acl_ver=P4903858V2&continue_url=http%3A%2F%2Fwww.msftco"+
           "nnecttest.com%2Fredirect" -join '')
```

Alright, let's use the color formatting from Visual Studio Code, to proceed with the lesson plan.

```
# // ______________________________________________________________________________________________________
# // |¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|
# // | $String is a really long string. Let's SPLIT it, so that it makes MORE SENSE                       |
# // |____________________________________________________________________________________________________|
# // |¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|¯¯¯¯¯¯¯¯¯|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|
# // | Property              | Type    | Value                                                            |
# // |_______________________|_________|__________________________________________________________________|
# // |¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|¯¯¯¯¯¯¯¯¯|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|
# // | Base                  | String  | https://n137.network-auth.com/splash/?                           |
# // | Network Mac Address   | String  | mac=88%3A15%3A44%3AA3%3AB7%3A10&                                 |
# // | Network IP Address    | String+ | real_ip=192.168.0.81&                                            |
# // | Client IP  Address    | String+ | client_ip=10.201.240.180&                                        |
# // | Client Mac Address    | String  | client_mac=9C:B7:0D:20:08:FE&                                    |
# // | Virtual Access Point  | Integer | vap=0&                                                           |
# // | Private A Variable    | String+ | a=a17554a0d2b15a664c0e7390018454f19e70227&                       |
# // | Private B Variable    | Integer | b=17468474&                                                      |
# // | Auth. Version         | Integer | auth_version=5&                                                  |
# // | Auth. Key             | String+ | key=834c46c40a4248fae0dec59501aef3f0327e6738&                    |
# // | Access Control List   | String+ | acl_ver=P4903858V2&                                              |
# // | Continue URL          | String+ | continue_url=http%3A%2F%2Fwww.msftconnecttest.com%2Fredirect     |
# // |_______________________|_________|__________________________________________________________________|
# // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

```
                                                                                                             __________/
 \___________________________________________________________________________________________________________/ Overview
```

GolubCorpNetworkAuth /---\
/-----------------------

```
# // ______________________________________________________________________
# // | Create a verbatim copy of the class represented by the URL string |
# // ----------------------------------------------------------------------

Class GolubCorpNetworkAuth
{
    Hidden [String] $base
    [String] $mac
    [String] $real_ip
    [String] $client_ip
    [String] $client_mac
    [UInt32] $vap
    [String] $a
    [UInt32] $b
    [UInt32] $auth_version
    [String] $key
    [String] $acl_ver
    [String] $continue_url
    GolubCorpNetworkAuth([String]$String)
    {
        # https://n137.network-auth.com/splash/?
        # mac=88%3A15%3A44%3AA3%3AB7%3A10&
        # real_ip=192.168.0.81&
        # client_ip=10.201.240.180&
        # client_mac=9C:B7:0D:20:08:FE&
        # vap=0&
        # a=a17554a0d2b15a664c0e73900184544f19e70227&
        # b=17468474&
        # auth_version=5&
        # key=834c46c40a4248fae0dec59501aef3f0327e6738&
        # acl_ver=P4903858V2&
        # continue_url=http%3A%2F%2Fwww.msftconnecttest.com%2Fredirect

        # // ______________________________________________________________
        # // | Use the Regex base class to catch+trim+split the URL query |
        # // --------------------------------------------------------------

        $E              = [Regex]::Matches($String,"\?.+").Value.TrimStart("?").Split("&")

        # // _______________________________________________________________________
        # // | Test the input, if it does not have (11) components, throw an error |
        # // -----------------------------------------------------------------------

        If ($E.Count -ne 11)
        {
            Throw "Invalid entry"
        }

        $This.mac          = $This.Tx($E[0])
        $This.real_ip      = $This.Tx($E[1])
        $This.client_ip    = $This.Tx($E[2])
        $This.client_mac   = $This.Tx($E[3])
        $This.vap          = $This.Tx($E[4])
        $This.a            = $This.Tx($E[5])
        $This.b            = $This.Tx($E[6])
        $This.auth_version = $This.Tx($E[7])
        $This.key          = $This.Tx($E[8])
        $This.acl_ver      = $This.Tx($E[9])
        $This.continue_url = $This.Tx($E[10])

        $This.base         = [Regex]::Matches($String,".+\?").Value.TrimEnd("?")
    }
    [String] Tx([String]$Entry)
    {
        # Slice assignment
        $0, $1 = $Entry -Split "="

        # Property message
```

```
        Write-Host "Setting [~] Property: [$0], Value: [$1]" -ForegroundColor 10

        # Property assignment
        Return $1
    }
    [String] Out()
    {
        Return @( $This.PSObject.Properties | % { $_.Name, $_.Value -join "=" } ) -join "&"
    }
    [String] ToString()
    {
        Return "{0}?{1}" -f $This.Base, $This.Out()
    }
}
```

```
# // ------------------------------------------------------------------------------------------------------------
# // | Now create an instantiation of the above class with the variable $String as it's only parameter |
# // ------------------------------------------------------------------------------------------------------------

$Test = [GolubCorpNetworkAuth]$String

# OR... you can use
# $Test = [GolubCorpNetworkAuth]::New($String)
```

```
PS Prompt:\> $Test = [GolubCorpNetworkAuth]$String
Setting [~] Property: [mac], Value: [88%3A15%3A44%3AA3%3AB7%3A10]
Setting [~] Property: [real_ip], Value: [192.168.0.81]
Setting [~] Property: [client_ip], Value: [10.201.240.180]
Setting [~] Property: [client_mac], Value: [9C:B7:0D:20:08:FE]
Setting [~] Property: [vap], Value: [0]
Setting [~] Property: [a], Value: [a17554a0d2b15a664c0e739001845 44f19e70227]
Setting [~] Property: [b], Value: [17468474]
Setting [~] Property: [auth_version], Value: [5]
Setting [~] Property: [key], Value: [834c46c40a4248fae0dec59501aef3f0327e6738]
Setting [~] Property: [acl_ver], Value: [P4903858V2]
Setting [~] Property: [continue_url], Value: [http%3A%2F%2Fwww.msftconnecttest.com%2Fredirect]

PS Prompt:\> $Test


mac          : 88%3A15%3A44%3AA3%3AB7%3A10
real_ip      : 192.168.0.81
client_ip    : 10.201.240.180
client_mac   : 9C:B7:0D:20:08:FE
vap          : 0
a            : a17554a0d2b15a664c0e739001845 44f19e70227
b            : 17468474
auth_version : 5
key          : 834c46c40a4248fae0dec59501aef3f0327e6738
acl_ver      : P4903858V2
continue_url : http%3A%2F%2Fwww.msftconnecttest.com%2Fredirect
```

```
# // ------------------------------------------------------------------------------------------------------------
# // |----------------------------------------------------------------------------------------------------------|
# // | Cool. Now some of that information has to be processed into an object.                                   |
# // | It IS an object right now, but- it actually has some issues such as ...                                  |
# // | ---------------------                                                                                    |
# // | | %3A = : | %2F = / |                                                                                    |
# // | ---------------------                                                                                    |
# // | Uh-oh. Those are gonna cause problems if we use them VERBATIM. Because...                                |
# // | ...those are actually CHARACTER CODES so that the browser can process the input string.                  |
# // | If we were to REPLACE EVERY "%" symbol with a [char]0x, we can get back the actual character.            |
# // |__________________________________________________________________________________________________________|
# // ------------------------------------------------------------------------------------------------------------
```

```
# // ____________________________________________________________________________________________________
# // |¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|
# // | PS Prompt:\> [char]0x2f                                                                          |
# // | /                                                                                                |
# // | PS Prompt:\> [char]0x3a                                                                          |
# // | :                                                                                                |
# // |__________________________________________________________________________________________________|
# // |¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|
# // | So, now what I'm going to do, is create MULTIPLE CLASSES so that I can cleanly organize the     |
# // | components of the input string, in order to get a more complex object BACK from it.             |
# // |__________________________________________________________________________________________________|
# // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

```
                                                                                                             ______________________/
\___________________________________________________________________________________________________/ GolubCorpNetworkAuth
  IPInterface /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯
```

This class will be for handling EACH of the NETWORK and the CLIENT addresses, (IP + Mac Address)

```
Class IpInterface
{
    [String] $Type
    [Object]   $IP
    [String]  $Mac
    IpInterface([UInt32]$Type,[String]$IPAddress,[String]$MacAddress)
    {
        # // ____________________________________________________________
        # // | Tests whether we're specifying a NETWORK or CLIENT interface |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        If ($Type -notin 0..1)
        {
            Throw "Invalid address type"
        }

        # // ______________________________________________________________________
        # // | Assign the type based on the input integer (0: Network, 1: Client) |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        $This.Type = @("Network","Client")[$Type]

        # // _________________________________________________________
        # // | Tests whether the IP address matches IPv4 conventions |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        If ($IPAddress -notmatch "(\d+\.\d+\.\d+\.\d+)")
        {
            Throw "Invalid IP Address"
        }

        # // ____________________________________________________________________________
        # // | Assigns the property "Ip", while also converting the string to an object |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        $This.Ip   = [IPAddress]$IPAddress

        # // _____________________________________________________
        # // | Tests whether the Mac address matches conventions |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        If ($MacAddress -notmatch (@("[A-F0-9]{2}")*6 -join ":"))
        {
            Throw "Invalid Mac Address"
        }
```

```
        # // ______________________________
        # // | Assigns the property "Mac" |
        # // ------------------------------

        $This.Mac  = $MacAddress
    }
    [String] ToString()
    {
        Return $This.Type, $This.Ip, $This.Mac -join "&"
    }
}
```

```
                                                                                                                         ____________/
\__________________________________________________________________________________________________________________/ IPInterface
  ApAuthenticationToken /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

This class will handle the information that is being sent/received by the Cisco Meraki Wireless Lan Controller on the backend, which MAY be in the store...? Or, it may be at Pchop HQ...

```
__________________________________________________________
| Golub Corporation | 461 Nott St. Schenectady, NY 12308 |
¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

Then again, maybe it's still sitting over at:

```
_________________________________________________________
| Nfrastructure | 5 Enterprise Lane, Halfmoon, NY 12065 |
¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

Look, I have no idea where it is specifically, or whether they work with Nfrastructure still...
I just know that they both extensively use CISCO equipment.
It's what EXPERTS use.

```
Class ApAuthenticationToken
{
    [UInt32] $Index
    [String] $A
    [UInt32] $B
    [UInt32] $Version
    [String] $Key
    [String] $Acl
    ApAuthenticationToken([UInt32]$Index,[String]$A,[UInt32]$B,[UInt32]$Version,[String]$Key,[String]$Acl)
    {
        # // _____________________________________________________________________________________
        # // | Access Point Index is essentially "which access point is this being issued to...?" |
        # // | on the CISCO MERAKI WIRELESS LAN CONTROLLER...                                      |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        $This.Index = $Index

        # // ____________________________________________________________________________
        # // | A is a 40-digit HEXADECIMAL address, which is 8 digits longer than a GUID |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        If ($A -notmatch "[a-f0-9]{40}")
        {
            Throw "Not a valid A"
        }

        # // ____________________________________
        # // | Now we can ASSIGN the property A |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        $This.A = $A

        # // ___________________________________________________________________________________
        # // | B in the example, appears to be an 8-digit Integer/[UInt32], though it could be  |
        # // | LARGER or SMALLER. The parameter input will automatically test whether it is the |
        # // | correct type. Now we can ASSIGN the property B                                   |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        $This.B = $B
```

```
        # // ____________________________________________________________________________________
        # // | Auth. Version in the example is a 1-digit integer, but perhaps it could be larger |
        # // | The parameter input will automatically test whether it is the correct type.       |
        # // | Now we can ASSIGN the property Version                                            |
        # // ------------------------------------------------------------------------------------

        $This.Version = $Version

        # // ______________________________________________________________________________
        # // | Key is a 40-digit HEXADECIMAL address, which is 8 digits longer than a GUID |
        # // ------------------------------------------------------------------------------

        If ($Key -notmatch "[a-f0-9]{40}")
        {
            Throw "Not a valid key"
        }

        # // __________________
        # // | Assigns the key |
        # // ------------------

        $This.Key = $Key

        # // __________________________________________________________________________________
        # // | ACL stands for access control list, it's apparently a string, since P+V are NOT |
        # // | hexadecimal characters                                                          |
        # // ----------------------------------------------------------------------------------

        $This.Acl = $Acl
    }
}
```

```
                                                                                                 _______________________/
\_______________________________________________________________________________________________/ ApAuthenticationToken
  GolubCorpNetworkAuth2 /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

So, this will be a more complex class that has main “branch” properties to make it APPEAR to be much simpler. It'll essentially have all of the same information as the original GolubCorpNetworkAuth class, but it will be FORMATTED differently.

```
Class GolubCorpNetworkAuth2
{
    [Object] $Network
    [Object] $Client
    [Object] $Token
    [String] $Continue
    [String] Cx([String]$String)
    {
        $A       = [Char[]]$String
        $X       = 0
        Return @( Do
        {
            Switch -Regex ($A[$X])
            {
                "\%"
                {
                    ("[Char]0x{0}{1}" -f $A[$X+1],$A[$X+2]) | Invoke-Expression; $X ++; $X ++
                }
                Default
                {
                    $A[$X]
                }
            }
            $X ++
        }
        Until ($X -eq $String.Length)) -join ''
    }
    GolubCorpNetworkAuth2([String]$String)
```

```
        {
            # // ------------------------------------------------------------------
            # // | Here's the input, but split...                                 |
            # // | https://n137.network-auth.com/splash/?                         |
            # // | mac=88%3A15%3A44%3AA3%3AB7%3A10&                               |
            # // | real_ip=192.168.0.81&                                          |
            # // | client_ip=10.201.240.180&                                      |
            # // | client_mac=9C:B7:0D:20:08:FE&                                  |
            # // | vap=0&                                                         |
            # // | a=a17554a0d2b15a664c0e73900184544f19e70227&                    |
            # // | b=17468474&                                                    |
            # // | auth_version=5&                                                |
            # // | key=834c46c40a4248fae0dec59501aef3f0327e6738&                  |
            # // | acl_ver=P4903858V2&                                            |
            # // | continue_url=http%3A%2F%2Fwww.msftconnecttest.com%2Fredirect |
            # // ------------------------------------------------------------------

            # // ----------------------------------------------------------------
            # // | Use the method Cx to convert any % input to the full character |
            # // ----------------------------------------------------------------

            $E = ForEach ($Item in [Regex]::Matches($String,"\?.+").Value.TrimStart("?").Split("&"))
            {
                $This.Cx($Item.Split("=")[1])
            }

            # // ----------------------------------
            # // | Assign the network information |
            # // ----------------------------------

            $This.Network      = [IPInterface]::New(0,$E[1],$E[0])

            # // ----------------------------------
            # // | Assign the client information |
            # // ----------------------------------

            $This.Client       = [IPInterface]::New(1,$E[2],$E[3])

            # // ----------------------------------
            # // | Compile the token information |
            # // ----------------------------------

            $This.Token        = [ApAuthenticationToken]::New($E[4],$E[5],$E[6],$E[7],$E[8],$E[9])

            # // -------------------------
            # // | Set the continue URL |
            # // -------------------------

            $This.Continue     = $E[10]
        }
    }
```

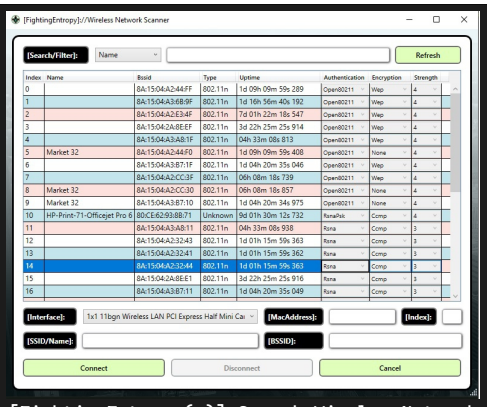
_____/
_____/ GolubCorpNetworkAuth2
Output (1) /_____\
/_____

Alright, now's the time to scope out the output of the above information. The information is NOT exactly the same, however, the NETWORK and CLIENT strings are EASIER to UNDERSTAND. That doesn't necessarily mean that will be the INTENDED result of the output string. There's plenty more to do with these classes.

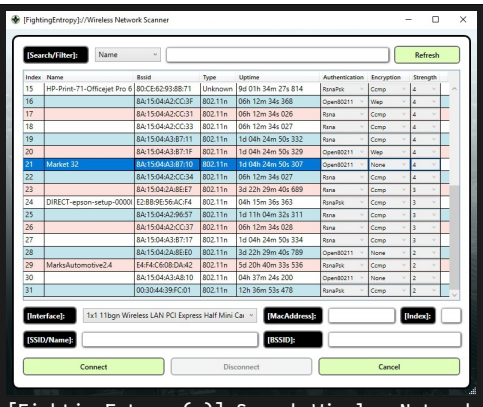
```
PS Prompt:\> $Test2 | Format-List

Network  : Network&192.168.0.81&88:15:44:A3:B7:10
Client   : Client&10.201.240.180&9C:B7:0D:20:08:FE
Token    : ApAuthenticationToken
Continue : http://www.msftconnecttest.com/redirect
```

```
# // ------------------------------------------------------
# // | Let's look at the SUBPROPERTIES of these properties  |
# // ------------------------------------------------------


PS Prompt:\> $Test2.Network

Type    IP           Mac
----    --           ---
Network 192.168.0.81 88:15:44:A3:B7:10

PS Prompt:\> $Test2.Client

Type    IP           Mac
----    --            ---
Client 10.201.240.180 9C:B7:0D:20:08:FE


PS Prompt:\> $Test2.Token

Index   : 0
A       : a17554a0d2b15a664c0e739001845444f19e70227
B       : 17468474
Version : 5
Key     : 834c46c40a4248fae0dec59501aef3f0327e6738
Acl     : P4903858V2

PS Prompt:\> $Test2.Continue
http://www.msftconnecttest.com/redirect
```

There is additional work that needs to be done with this particular function to have it reproduce the ORIGINAL information, however, this code is effectively what is being done on the BACKEND of the server. The Cisco Meraki WLAN Controller has to maintain an Open WiFi authentication, with a WEP encrypted connection INTERNALLY.



[FightingEntropy(π)] Search-WirelessNetwork Golub Corporation Cisco Meraki Results (1/2) (Click to expand)



[FightingEntropy(π)] Search-WirelessNetwork Golub Corporation Cisco Meraki Results (2/2) (Click to expand)

"88:15:44:A3:B7:10" is in the second results box. The function Search-WirelessNetwork is available in the PowerShell module, [FightingEntropy(π)]. The instructions to install the module are located at...

| [FightingEntropy(π)] | https://github.com/mcc85s/FightingEntropy |

```
# // ------------------------------------------------------------------------------------------------
# // | Now, we have some stuff to play around with, particularly the assemblies and types           |
# // | available in the function Search-WirelessNetwork in [FightingEntropy(π)]                     |
# // ------------------------------------------------------------------------------------------------
```

_____/ Output (1)

Correlations /----------\
/----------

```
# // ------------------------------------------------------------------------------------------
# // | What I'm going to do, is to call the Use-Wlanapi function, in order to pull some of the  |
# // | type information so I can access the RADIOS from the PowerShell console.                 |
# // | Calling the function only assembles the code, it doesn't instantiate it.                 |
# // |                                                                                          |
# // | So, we will do that below with the function Add-Type with various additional parameters. |
# // |                                                                                          |
# // | I'm going to pick apart the function Search-WirelessNetwork, and paste the content       |
# // | of its' several classes, and then use those classes to examine some of input/output.     |
# // |                                                                                          |
# // | By the time the lesson plan is over with...? Some people will probably say to themselves |
# // | "This dude is a legitimate expert that knows what the hell he's doing..."                |
# // |                                                                                          |
# // | That's the point of all of this. If I want to CONVINCE somebody who's a PHILANTHROPIST,  |
# // | that a couple dudes like CHRISTOPHER MURPHY and DANIEL PICKETT have been REMOTELY        |
# // | INTERACTING with my DEVICES...? Then I have to find a way to BYPASS people in society   |
# // | who think that stuff sounds INSANE. Right...? Cause, I gotta tell ya...                  |
# // |------------------------------------------------------------------------------------------|
# // |------------------------------------------------------------------------------------------|
# // | (Michael Edward Cook + Jesse Pickett + Husdon Valley Community College)    [Correlation] |
# // | https://github.com/mcc85s/FightingEntropy/blob/main/Docs/2021_0414-(Jesse%20Pickett).pdf |
# // |------------------------------------------------------------------------------------------|
# // |------------------------------------------------------------------------------------------|
# // | (Nfrastructure + FBI Investigator Murphy + Golub Corporation)              [Correlation] |
# // | 05/23/20 0133 | https://youtu.be/3twiZEsyQf0                                             |
# // | 05/23/20 0203 | https://youtu.be/V-_YqedKZb8                                             |
# // |------------------------------------------------------------------------------------------|
# // |------------------------------------------------------------------------------------------|
# // | (Nfrastructure + Shenendehowa + Golub Corporation + etc.)                                |
# // | 05/23/20 1200 | https://youtu.be/HT4p28bRhqc                                             |
# // |------------------------------------------------------------------------------------------|
# // |------------------------------------------------------------------------------------------|
# // | Uh-oh. I'm an actual expert that knows what he's saying/doing, whether it's with the     |
# // | TECHNOLOGY or it's the PEOPLE I just mentioned. Sometimes people such as myself are SO   |
# // | INTELLIGENT, that when I SAY STUFF THAT HAS SUPPORTING EVIDENCE for my THEORIES...?       |
# // | They'll be like "But such-and-such told me...", like LAURA HUGHES from SARATOGA COUNTY.  |
# // |                                                                                          |
# // | That's the power of PEOPLE WHO LIE...                                                    |
# // | If such-and-such told you (1) thing...                                                   |
# // | But the SUPPORTING EVIDENCE tells you (1) OTHER thing...?                                |
# // | *squinting* ...what do you think that means...?                                          |
# // | It means this: such-and-such lied to you, and you just blindly followed along.           |
# // | Oh boy. Whatever will we do if everybody lies to one another...?                         |
# // |                                                                                          |
# // | People think that what I say sounds INSANE, because they don't make what's otherwise     |
# // | referred to as an "EFFORT", to review the SUPPORTING EVIDENCE, in order to UNDERSTAND my |
# // | "THEORIES".                                                                              |
# // |                                                                                          |
# // | That's pretty important, ESPECIALLY if you're a POLICE OFFICER or an INVESTIGATOR.       |
# // | Unfortunately, the INVESTIGATORS at (SCSO/NYSP), they are not that intelligent.          |
# // | Neither are half of the people who work for SARATOGA COUNTY.                             |
# // |------------------------------------------------------------------------------------------|
# // |------------------------------------------------------------------------------------------|
# // | (SCSO Captain Jeffrey Brown)                                                             |
# // | 02/02/21 | https://drive.google.com/file/d/1JECZXhwpXFO5B8fvFnLftESp578PFVF8              |
# // |------------------------------------------------------------------------------------------|
# // |------------------------------------------------------------------------------------------|
# // | If they WERE...? They would have been able to make the SAME CORRELATIONS I have made,    |
# // | over the last 2.5 years since I recorded that content up above. The CORRELATIONS are...  |
# // |                                                                                          |
# // | My dad + Jesse Pickett -> studied PROGRAMMING @ HVCC in 88/89 -> Nfrastructure 92 ->     |
# // | My dad murdered by GANG/MAFIA/KGB in 95 -> also Sammy Santa Cassaro ** in 96 ->          |
# // | Same cab company + Same radio dispatcher + Lived in the same neighborhood + Both murders |
# // | APPEAR to have been MONEY/DRUG related, but APPEARNCES CAN BE DECEIVING -> I could go on |
# // |                                                                                          |
# // | How those 26-27 year old murder cases became RELEVANT AGAIN, is because:                 |
# // | I believe that someone had my father murdered, and that they tried to have ME murdered,  |
# // | after my NETWORK at (Computer Answers - 1602 US-9, Clifton Park NY 12065) was subjected  |
```

```
# // | to an extremely sophisticated cyberattack on 01/15/2019 involving:                  |
# // |                                                                                     |
# // | CVE-2019-8936, DDOS, WannaCry derivative, and I also believe that my Apple iPhone 8+ had |
# // | a DANGEROUS PROGRAM CALLED PHANTOM/PEGASUS DEPLOYED TO IT TO SPY ON ME...           |
# // | ...and that AFTER I recorded those ABOVE exhibits from 05/23/20...?                 |
# // | 2x 25-30 year old white males attempted to murder me outside of COMPUTER ANSWERS between |
# // | 05/25/20 2343 -> 05/26/20 0130 leading to SCSO-2020-028501.                        |
# // |                                                                                     |
# // | I visually confirmed that I RECORDED A VIDEO, of these 2 guys, using Pegasus/Phantom, |
# // | that appeared to be KNEE DEEP in attempting to: MURDER ME.                          |
# // | I told SCSO SCOTT SCHELLING about this (event/video) and my 2x 911 calls were subjected |
# // | to what's otherwise known as a TELEPHONY DENIAL OF SERVICE attack, (feature of Pegasus), |
# // | and for whatever reason...? Someone in the government disabled my iPhone 8+ after this. |
# // | Oh boy. Whatever will I do...?                                                      |
# // |                                                                                     |
# // | At some point in history, planet Earth was overrun by morons who can't think real hard. |
# // | I think I've been saying that for like *checks watch* 2.5 years now.               |
# // |                                                                                     |
# // | I'm just, surrounded by people that think I'm making ALL of that stuff up, right...? |
# // | And that's ok. Sometimes people are SLOW ON THE UPTAKE.                             |
# // |                                                                                     |
# // | I'm not gonna go around and shake my finger in people's faces if they don't UNDERSTAND |
# // | what I've been saying...? However, uh- given the detail of what I've discussed so far in |
# // | this LESSON PLAN, as well as links to the several pieces of "SUPPORTING EVIDENCE", I'm |
# // | fairly certain that ANYBODY WITH AN ACTUAL BRAIN, may be able to LOGICALLY DEDUCE that |
# // | what I'm SUGGESTING has a PLAUSIBILITY FACTOR AFTER ALL.                            |
# // |                                                                                     |
# // | That said, let's proceed with the lesson plan.                                      |
# // |_____________________________________________________________________________________|
# // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

# // __________________________________________________________________________
# // | Load the types from the function Use-Wlanapi                            |
# // | Since the command is rather long, I'll perform what's called "SPLATTING"... |
# // | by assigning a variable named $Splat to a hashtable.                    |
# // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

$Splat               = @{

    MemberDefinition = Use-Wlanapi       # <- Install [FightingEntropy(π)] to use this
    Using            = "System.Text"
    Namespace        = "WiFi"
    Name             = "ProfileManagement"
}

Add-Type @Splat -Passthru | Out-Null
```

```
                                                                                                    ______________/
\___________________________________________________________________________________________________/ Correlations
  Ssid /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯
```

```
# // ____________________________________________________________________________________
# // | Provides an accurate representation of the information collected by the wireless radio(s) |
# // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

Class Ssid
{
    [UInt32] $Index
    Hidden [Object] $Ssid
    [String] $Name
    [Object] $Bssid
    [String] $Type
    Hidden [UInt32] $TypeSlot
    Hidden [String] $TypeDescription
    [Object] $Uptime
    [String] $NetworkType
    [String] $Authentication
    Hidden [UInt32] $AuthenticationSlot
    Hidden [String] $AuthenticationDescription
```

```
        [String] $Encryption
        Hidden [UInt32] $EncryptionSlot
        Hidden [String] $EncryptionDescription
        [UInt32] $Strength
        [String] $BeaconInterval
        [Double] $ChannelFrequency
        [Bool]   $IsWifiDirect
        Ssid([UInt32]$Index,[Object]$Object)
        {
            $This.Index              = $Index
            $This.Ssid               = $Object
            $This.Name               = $Object.Ssid
            $This.Bssid              = $Object.Bssid.ToUpper()
            $This.GetPhyType($Object.PhyKind)
            $This.Uptime             = $This.GetUptime($Object.Uptime)
            $This.NetworkType        = $Object.NetworkKind
            $This.Authentication     = $Object.SecuritySettings.NetworkAuthenticationType
            $This.GetNetAuthType($This.Authentication)
            $This.Encryption         = $Object.SecuritySettings.NetworkEncryptionType
            $This.GetNetEncType($This.Encryption)
            $This.Strength           = $Object.SignalBars
            $This.BeaconInterval     = $Object.BeaconInterval
            $This.ChannelFrequency   = $Object.ChannelCenterFrequencyInKilohertz
            $This.IsWiFiDirect       = $Object.IsWiFiDirect
        }
        [String] ToString()
        {
            Return $This.Name
        }
        [String] GetUptime([String]$Uptime)
        {
            $Slot      = @( )
            $Total     = $Uptime -Split "(\:|\.)" | ? { $_ -match "\d+" }
            $Ticks     = $Total[-1].Substring(0,3)
            $Seconds   = "{0}s" -f $Total[-2]
            $Minutes   = "{0}m" -f $Total[-3]
            $Hours     = "{0}h" -f $Total[-4]
            If ($Total[-5])
            {
                $Days  = "{0}d" -f $Total[-5]
                $Slot += $Days
            }

            If ($Total[-4])
            {
                $Slot += $Hours
            }

            If ($Total[-3])
            {
                $Slot += $Minutes
            }

            If ($Total[-2])
            {
                $Slot += $Seconds
            }

            If ($Total[-1])
            {
                $Slot += $Ticks
            }
            Return @( $Slot -join " " )
        }
        GetPhyType([String]$PhyKind)
        {
            $Types         = "Unknown","Fhss","Dsss","IRBaseband","Ofdm",
                             "Hrdsss","Erp","HT","Vht","Dmg","HE"
            $This.TypeSlot = $Types.IndexOf($PhyKind)
            $This.TypeDescription = Switch ($PhyKind)
            {
                Unknown     { "Unspecified physical type"                    }
```

```
            Fhss        { "(FHSS/Frequency-Hopping Spread-Spectrum)"       }
            Dsss        { "(DSSS/Direct Sequence Spread-Spectrum)"         }
            IRBaseband  { "(IR/Infrared baseband)"                         }
            Ofdm        { "(OFDM/Orthogonal Frequency Division Multiplex)" }
            Hrdsss      { "(HRDSSS/High-rated DSSS)"                       }
            Erp         { "(ERP/Extended Rate)"                            }
            HT          { "(HT/High Throughput [802.11n])"                 }
            Vht         { "(VHT/Very High Throughput [802.11ac])"          }
            Dmg         { "(DMG/Directional Multi-Gigabit [802.11ad])"     }
            HE          { "(HEW/High-Efficiency Wireless [802.11ax])"      }
        }
        $Regex          = [Regex]::Matches($this.TypeDescription,"(802\.11\w+)").Value
        $This.Type      = @("Unknown",$Regex)[$This.TypeDescription -match 802.11]
    }
    GetNetAuthType([String]$Auth)
    {
        $Types          = "None","Unknown","Open80211","SharedKey80211","Wpa",
                          "WpaPsk","WpaNone","Rsna","RsnaPsk","Ihv","Wpa3",
                          "Wpa3Enterprise192Bits","Wpa3Sae","Owe","Wpa3Enterprise"
        $This.AuthenticationSlot = $Types.IndexOf($Auth)
        $This.AuthenticationDescription = Switch -Regex ($Auth)
        {
            ^None$
            {
                "No authentication enabled."
            }
            ^Unknown$
            {
                "Authentication method unknown."
            }
            ^Open80211$
            {
                "Open authentication over 802.11 wireless.",("Devices are authenticated and can connect"+
                " to an access point."),("Communication w/ network requires matching (WEP/Wired Equival"+
                "ent Privacy) key.")
            }
            ^SharedKey80211$
            {
                "Specifies an IEEE 802.11 Shared Key authentication algorithm.",("Requires pre-shared ("+
                "WEP/Wired Equivalent Privacy) key for 802.11 authentication.")
            }
            ^Wpa$
            {
                "Specifies a (WPA/Wi-Fi Protected Access) algorithm.",("IEEE 802.1X port authorization "+
                "is performed by the supplicant, authenticator, and authentication server."),("Cipher k"+
                "eys are dynamically derived through the authentication process.")
            }
            ^WpaPsk$
            {
                "Specifies a (WPA/Wi-Fi Protected Access) algorithm that uses (PSK/pre-shared key).",
                "IEEE 802.1X port authorization is performed by the supplicant and authenticator.",
                ("Cipher keys are dynamically derived through a PSK that is used on both the supplicant"+
                " and authenticator.")
            }
            ^WpaNone$
            {
                "Wi-Fi Protected Access."
            }
            ^Rsna$
            {
                "Specifies an IEEE 802.11i (RSNA/Robust Security Network Association) algorithm.",("IEE"+
                "E 802.1X port authorization is performed by the supplicant, authenticator, and authent"+
                "ication server."),"Cipher keys are dynamically derived through the auth. process."
            }
            ^RsnaPsk$
            {
                "Specifies an IEEE 802.11i RSNA algorithm that uses (PSK/pre-shared key).",
                "IEEE 802.1X port authorization is performed by the supplicant and authenticator.",
                ("Cipher keys are dynamically derived through a PSK that is used on both the supplican"+
                "t and authenticator.")
            }
            ^Ihv$
```

```
            {
                "Specifies an authentication type defined by an (IHV/Independent Hardware Vendor)."
            }
            "(^Wpa3$|^Wpa3Enterprise192Bits$)"
            {
                ("Specifies a 192-bit encryption mode for (WPA3-Enterprise/Wi-Fi Protected Access 3 Ent"+
                "erprise) networks.")
            }
            ^Wpa3Sae$
            {
                ("Specifies (WPA3 SAE/Wi-Fi Protected Access 3 Simultaneous Authentication of Equals) "+
                "algorithm."),("WPA3 SAE is the consumer version of WPA3. SAE is a secure key establis"+
                "hment protocol between devices;"),("SAE provides: synchronous authentication, and str"+
                "onger protections for users against password-guessing attempts by third parties.")
            }
            ^Owe$
            {
                "Specifies an (OWE/Opportunistic Wireless Encryption) algorithm.",
                "OWE provides opportunistic encryption over 802.11 wireless networks.",
                "Cipher keys are dynamically derived through a (DH/Diffie-Hellman) key exchange-",
                "Enabling data protection without authentication."
            }
            ^Wpa3Enterprise$
            {
                "Specifies a (WPA3-Enterprise/Wi-Fi Protected Access 3 Enterprise) algorithm.","WPA3-En"+
                "terprise uses IEEE 802.1X in a similar way as (RSNA/Robust Security Network Associatio"+
                "n)-"),("However, it provides increased security through the use of mandatory certifica"+
                "te validation and protected management frames.")
            }
        }
    }
    GetNetEncType([String]$Enc)
    {
        $Types = "None","Unknown","Wep","Wep40","Wep104","Tkip","Ccmp","WpaUseGroup","RsnUseGroup",
                 "Ihv","Gcmp","Gcmp256"
        $This.EncryptionSlot = $Types.IndexOf($Enc)
        $This.EncryptionDescription = Switch ($Enc)
        {
            None
            {
                "No encryption enabled."
            }
            Unknown
            {
                "Encryption method unknown."
            }
            Wep
            {
                "Specifies a WEP cipher algorithm with a cipher key of any length."
            }
            Wep40
            {
                ("Specifies an RC4-based (WEP/Wired Equivalent Privacy) algorithm specified in IEEE 802"+
                ".11-1999."),"This enumerator specifies the WEP cipher algorithm with a 40-bit cipher key."
            }
            Wep104
            {
                "Specifies a (WEP/Wired Equivalent Privacy) cipher algorithm with a 104-bit cipher key."
            }
            Tkip
            {
                "Specifies an RC4-based cipher (TKIP/Temporal Key Integrity Protocol) algorithm",("This"+
                " cipher suite that is based on algorithms defined in WPA + IEEE 802.11i-2004 standards."),
                "This cipher also uses the (MIC/Message Integrity Code) algorithm for forgery protection."
            }
            Ccmp
            {
                "Specifies an [IEEE 802.11i-2004 & RFC 3610] AES-CCMP algorithm standard.",
                "(AES/Advanced Encryption Standard) is the encryption algorithm defined in FIPS PUB 197."
            }
            WpaUseGroup
            {
```

```
                "Specifies a (WPA/Wifi Protected Access) Use Group Key cipher suite.",
                "For more information about the Use Group Key cipher suite, refer to:",
                "Clause 7.3.2.25.1 of the IEEE 802.11i-2004 standard."
            }
            RsnUseGroup
            {
                "Specifies a (RSN/Robust Security Network) Use Group Key cipher suite.",
                "For more information about the Use Group Key cipher suite, refer to:",
                "Clause 7.3.2.25.1 of the IEEE 802.11i-2004 standard."
            }
            Ihv
            {
                "Specifies an encryption type defined by an (IHV/Independent Hardware Vendor)."
            }
            Gcmp
            {
                "Specifies an [IEEE 802.11-2016] AES-GCMP algorithm w/ 128-bit key.",
                "(AES/Advanced Encryption Standard) is the encryption algorithm defined in FIPS PUB 197."
            }
            Gcmp256
            {
                "Specifies an [IEEE 802.11-2016] AES-GCMP algorithm w/ 256-bit key.",
                "(AES/Advanced Encryption Standard) is the encryption algorithm defined in FIPS PUB 197."
            }
        }
    }
}
```

So, that's the Ssid (type/class) object, which we will use shortly.
If you're wondering, this information is from the Microsoft website.

Some of these properties actually do the same thing as an Enum, but I don't really like using Enums because they need to be STATICALLY WRITTEN. Though the information is statically written, I think that this approach also works.

In actuality, I would like to further develop this particular class so that it has subclasses in a similar manner to the classes I wrote above within the GolubCorpNetworkAuth2 class for the IpInterface, and and the ApAuthenticationToken classes.

Maybe I WILL at SOME POINT... but- not right now.

```
                                                                                                     ______/
\___________________________________________________________________________________________________/ Ssid
  WiFiProfile /‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾\
/‾‾‾‾‾‾‾‾‾‾‾‾‾
```

```
# // ______________________________
# // | Handles the profile objects |
# // ‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾

Class WiFiProfile
{
    [Object] $Interface
    [String] $Name
    [String] $Flags
    [Object] $Detail
    WiFiProfile([Object]$Interface,[Object]$ProfileInfo)
    {
        $This.Interface = $Interface
        $This.Name      = $ProfileInfo.strProfileName
        $This.Flags     = $ProfileInfo.ProfileFlags
        $This.Detail    = $Null
    }
    [String[]] ToString()
    {
        Return @(
        " ",
        "Interface        : $($This.Interface.Name)",
        "Guid             : $($This.Interface.Guid)",
        "Description      : $($This.Interface.Description)",
        "IfIndex          : $($This.Interface.ifIndex)",
```

```
            "Status          : $($This.Interface.Status)",
            "MacAddress      : $($This.Interface.MacAddress)",
            "LinkSpeed       : $($This.Interface.LinkSpeed)",
            "State           : $($This.Interface.State)",
            "----"
            "SSID            : $($This.Name)",
            "Flags           : $($This.Flags)",
            "ProfileName     : $($This.Detail.ProfileName)",
            "ConnectionMode  : $($This.Detail.ConnectionMode)",
            "Authentication  : $($This.Detail.Authentication)",
            "Encryption      : $($This.Detail.Encryption)",
            "Password        : $($This.Detail.Password)",
            "HiddenSSID      : $($This.Detail.ConnectHiddenSSID)",
            "EAPType         : $($This.Detail.EAPType)",
            "ServerNames     : $($This.Detail.ServerNames)",
            "TrustedRootCA   : $($This.Detail.TrustedRootCA)",
            "----",
            "XML             : $($This.Detail.Xml)",
            " "
            )
        }
    }
```

```
                                                                                                       ____________/
\_____________________________________________________________________________________________________/ WiFiProfile
  InterfaceObject /----------------------------------------------------------------------------------------------\
/----------------
```

In order to instruct a particular detected radio, to scan for wireless networks, AND use that above class...? We'll need an Interface object. That's what these (2) classes below, are for. This will pull the available interfaces from ONE of the following commands...

```
-----------------------------------------------------------------------------------------------
| Get-NetAdapter | Get-WmiObject Win32_NetworkAdapter | netsh interface/wlan show interface |
-----------------------------------------------------------------------------------------------
```

These commands, though SIMILAR, are NOT the same. Apparently the version of PowerShell being used has an impact on it's success... so these two classes below, are meant to capture the input/output of various ways to go about getting the adapter information. Adapters and Interfaces are pretty similar, however- an adapter is a PHYSICAL component, whereas an interface is the LOGICAL component. The terms are often used interchangeably.

```
    # // ____________________________________________________________
    # // | Represents an individual wireless interface on the host. |
    # // ------------------------------------------------------------

    Class InterfaceObject
    {
        [String] $Name
        [String] $Guid
        [String] $Description
        [UInt32] $ifIndex
        [String] $Status
        [String] $MacAddress
        [String] $LinkSpeed
        [String] $State
        InterfaceObject([Object]$Info,[Object]$Interface)
        {
            $This.Name        = $Interface.Name
            $This.Guid        = $Info.Guid
            $This.Description = $Info.Description
            $This.ifIndex     = $Interface.ifIndex
            $This.Status      = $Interface.Status
            $This.MacAddress  = $Interface.MacAddress.Replace("-",":")
            $This.LinkSpeed   = $Interface.LinkSpeed
            $This.State       = $Info.State
        }
    }
```

```
                                                                                                   ________________/
\_________________________________________________________________________________________________/ InterfaceObject
```

WlanInterface /----------\
/---------------

```powershell
# // ------------------------------------------------------------
# // | Parses WLAN adapter information returned from the netsh. |
# // | Not nearly as CLEAN as accessing wlanapi.dll...?         |
# // | But- it is included as a FALLBACK MECHANISM.             |
# // ------------------------------------------------------------

Class WlanInterface
{
    Hidden [String[]] $Select
    [String] $Name
    [String] $Description
    [String] $Guid
    [String] $MacAddress
    [String] $InterfaceType
    [String] $State
    [String] $Ssid
    [String] $Bssid
    [String] $NetworkType
    [String] $RadioType
    [String] $Authentication
    [String] $Cipher
    [String] $Connection
    [String] $Band
    [UInt32] $Channel
    [Float]  $Receive
    [Float]  $Transmit
    [String] $Signal
    [String] $Profile
    WlanInterface([String[]]$Select)
    {
        $This.Select                = $Select
        $This.Name                  = $This.Find("Name")
        $This.Description           = $This.Find("Description")
        $This.GUID                  = $This.Find("GUID")
        $This.MacAddress            = $This.Find("Physical address")
        $This.InterfaceType         = $This.Find("Interface type")
        $This.State                 = $This.Find("State")
        $This.Ssid                  = $This.Find("SSID")
        $This.Bssid                 = $This.Find("BSSID") | % ToUpper
        $This.NetworkType           = $This.Find("Network type")
        $This.RadioType             = $This.Find("Radio type")
        $This.Authentication        = $This.Find("Authentication")
        $This.Cipher                = $This.Find("Cipher")
        $This.Connection            = $This.Find("Connection mode")
        $This.Band                  = $This.Find("Band")
        $This.Channel               = $This.Find("Channel")
        $This.Receive               = $This.Find("Receive rate \(Mbps\)")
        $This.Transmit              = $This.Find("Transmit rate \(Mbps\)")
        $This.Signal                = $This.Find("Signal")
        $This.Profile               = $This.Find("Profile")
    }
    [String] Find([String]$String)
    {
        Return @(($This.Select | ? { $_ -match "(^\s+$String\s+\:)" }).Substring(29))
    }
}
```

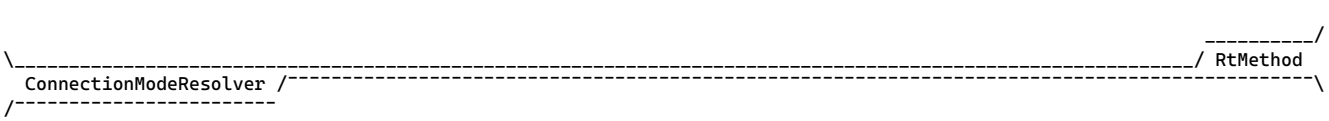
_____________/
_____________/ WlanInterface

RtMethod /----------\
/----------

This particular class here is meant to make the Task() method in the MAIN class, SMALLER and MORE COMPACT.
I had to create this for that SINGLE line alone, since it spanned too wide for the formatting of this document.

Anytime you see the text and the code blocks with colored code, or just plain text, I am MANUALLY formatting the content to fit the dimensions of the document. This ALSO requires me to have to WRITE the code a SPECIFIC way.

```
# // ------------------------------------------------------------
# // | Specifically for selecting/filtering a Runtime IAsyncTask |
# // ------------------------------------------------------------

Class RtMethod
{
    [String] $Name
    [Object] $Params
    [Object] $Count
    [Object] $Object
    RtMethod([Object]$Object)
    {
        $This.Object = $Object
        $This.Params = $Object.GetParameters()
        $This.Count  = $This.Params.Count
        $This.Name   = $This.Params[0].ParameterType.Name
    }
}
```




```
# // ---------------------------
# // | Better than a hashtable |
# // ---------------------------

Class ConnectionModeResolver
{
    [String] $Profile           = "WLAN_CONNECTION_MODE_PROFILE"
    [String] $TemporaryProfile  = "WLAN_CONNECTION_MODE_TEMPORARY_PROFILE"
    [String] $DiscoverySecure   = "WLAN_CONNECTION_MODE_DISCOVERY_SECURE"
    [String] $Auto              = "WLAN_CONNECTION_MODE_AUTO"
    [String] $DiscoveryUnsecure = "WLAN_CONNECTION_MODE_DISCOVERY_UNSECURE"
}
```

Alright, this is where the magic happens. Things are about to get pretty real. I'm going to heavily modify the original class, so that I can use it explicitly in the PowerShell console, without having to post the Xaml classes for the Windows Presentation Framework. That is a very heavy component of this utility...

```
# // ------------------------------------------------------------------------------
# // | Controller class for the function, this encapsulates the XAML/GUI, as well as  |
# // | ALL of the various classes and functions necessary to access the radios.       |
# // ------------------------------------------------------------------------------

Class Wireless
{
    Hidden [Object] $Module
    Hidden [String] $OEMLogo
    [Object] $Adapters
    [Object] $Request
    [Object] $Radios
    [Object] $List
    [Object] $Output
    [Object] $Selected
    [Object] $Connected
    [Object] Task()
    {
        Return [System.WindowsRuntimeSystemExtensions].GetMethods() | ? Name -eq AsTask | % {
```

```
            [RtMethod]$_ } | ? Count -eq 1 | ? Name -eq IAsyncOperation``1 | % Object
    }
    [Object] RxStatus()
    {
        Return [Windows.Devices.Radios.RadioAccessStatus]
    }
    [Object[]] RxAsync()
    {
        Return [Windows.Devices.Radios.Radio]::RequestAccessAsync()
    }
    [Object] RsList()
    {
        Return [System.Collections.Generic.IReadOnlyList[Windows.Devices.Radios.Radio]]
    }
    [Object[]] RsAsync()
    {
        Return [Windows.Devices.Radios.Radio]::GetRadiosAsync()
    }
    [Object] RaList()
    {
        Return [System.Collections.Generic.IReadOnlyList[Windows.Devices.WiFi.WiFiAdapter]]
    }
    [Object[]] RaAsync()
    {
        Return [Windows.Devices.WiFi.WiFiAdapter]::FindAllAdaptersAsync()
    }
    [Object] RadioRequestAccess()
    {
        Return $This.Task().MakeGenericMethod($This.RxStatus()).Invoke($Null,$This.RxAsync())
    }
    [Object] RadioSynchronization()
    {
        Return $This.Task().MakeGenericMethod($This.RsList()).Invoke($Null, $This.RsAsync())
    }
    [Object] RadioFindAllAdaptersAsync()
    {
        Return $This.Task().MakeGenericMethod($This.RaList()).Invoke($Null, $This.RaAsync())
    }
    [Object] NetshShowInterface([String]$Name)
    {
        Return [WlanInterface]::New((netsh wlan show interface $Name))
    }
    [String] Win32Exception([UInt32]$RC)
    {
        # // ------------------
        # // | RC: ReasonCode |
        # // ------------------

        Return "[System.ComponentModel.Win32Exception]::new($RC)" | IEX
    }
    [Object] WlanReasonCodeToString([UInt32]$RC,[UInt32]$BS,[Object]$SB,[IntPtr]$Res)
    {
        # // ------------------------------------------------------------------------
        # // | RC: ReasonCode | BS: BufferSize | SB: StringBuilder | Res: Reserved |
        # // ------------------------------------------------------------------------

        Return "[WiFi.ProfileManagement]::WlanReasonCodeToString($RC,$BS,$SB,$Res)" | IEX
    }
    [Void] WlanFreeMemory([IntPtr]$P)
    {
        # // --------------
        # // | P: Pointer |
        # // --------------

        "[WiFi.ProfileManagement]::WlanFreeMemory($P)" | IEX
    }
    [Object] WlanOpenHandle([UInt32]$CV,[IntPtr]$PR,[UInt32]$NV,[IntPtr]$CH)
    {
        # // ------------------------------------------------------------------------------
        # // | CV: ClientVersion | PR: pReserved | NV: NegotiatedVersion | CH: ClientHandle |
        # // ------------------------------------------------------------------------------
```

```
        Return "[WiFi.ProfileManagement]::WlanOpenHandle($CV, $PR, $NV, $CH)" | IEX
    }
    [Object] WlanCloseHandle([IntPtr]$CH,[IntPtr]$Res)
    {
        # // ------------------------------------
        # // | CH: ClientHandle | Res: Reserved |
        # // ------------------------------------

        Return "[WiFi.ProfileManagement]::WlanCloseHandle($CH, $Res)" | IEX
    }
    [Object] WlanEnumInterfaces([IntPtr]$CH,[IntPtr]$PR,[IntPtr]$IL)
    {
        # // ------------------------------------------------------------
        # // | CH: ClientHandle | PR: pReserved | IL: ppInterfaceList |
        # // ------------------------------------------------------------

        Return "[WiFi.ProfileManagement]::WlanEnumInterfaces($CH, $PR, $IL)" | IEX
    }
    [Object] WlanInterfaceList([IntPtr]$IIL)
    {
        # // ----------------------------
        # // | IIL: ppInterfaceInfoList |
        # // ----------------------------

        Return "[WiFi.ProfileManagement+WLAN_INTERFACE_INFO_LIST]::new($IIL)" | IEX
    }
    [Object] WlanInterfaceInfo([Object]$II)
    {
        # // -------------------------
        # // | II: WlanInterfaceInfo |
        # // -------------------------

        Return "[WiFi.ProfileManagement+WLAN_INTERFACE_INFO]$II" | IEX
    }
    [Object] WlanGetProfileList([IntPtr]$CH,[guid]$IG,[IntPtr]$PR,[IntPtr]$PL)
    {
        # // ------------------------------------------------------------------------
        # // | CH: ClientHandle | IG: InterfaceGuid | PR: pReserved | PL: ProfileList |
        # // ------------------------------------------------------------------------

        Return "[WiFi.ProfileManagement]::WlanGetProfileList($CH,$IG,$PR,$PL)" | IEX
    }
    [Object[]] WlanGetProfileListFromPtr([IntPtr]$PLP)
    {
        # // ---------------------------
        # // | PLP: ProfileListPointer |
        # // ---------------------------

        Return "[WiFi.ProfileManagement+WLAN_PROFILE_INFO_LIST]::new($PLP).ProfileInfo" | IEX
    }
    [Object] WlanGetProfile([IntPtr]$CH,[Guid]$IG,[String]$PN,[IntPtr]$PR,[String]$X,[UInt32]$F,[UInt32]$A)
    {
        # // ------------------------------------------------------------
        # // | CH: ClientHandle | IG: InterfaceGuid | PN: ProfileName |
        # // | PR: pReserved | X: Xml | F: Flags | A: Access          |
        # // ------------------------------------------------------------

        Return "[WiFi.ProfileManagement]::WlanGetProfile($CH,$IG,$PN,$PR,$X,$F,$A)" | IEX
    }
    [Object] WlanProfileInfoObject()
    {
        Return "[WiFi.ProfileManagement+ProfileInfo]::New()" | IEX
    }
    [Object] WlanConnectionParams()
    {
        Return "[WiFi.ProfileManagement+WLAN_CONNECTION_PARAMETERS]::new()" | IEX
    }
    [Object] WlanConnectionMode([String]$CM)
    {
        # // ----------------------
        # // | CM: ConnectionMode |
        # // ----------------------
```

```
        Return "[WiFi.ProfileManagement+WLAN_CONNECTION_MODE]::$CM" | IEX
    }
    [Object] WlanDot11BssType([String]$D)
    {
        # // ____________________
        # // | D: Dot11BssType |
        # // --------------------

        Return "[WiFi.ProfileManagement+DOT11_BSS_TYPE]::$D" | IEX
    }
    [Object] WlanConnectionFlag([String]$F)
    {
        # // ____________
        # // | F: Flag |
        # // ------------

        Return "[WiFi.ProfileManagement+WlanConnectionFlag]::$F" | IEX
    }
    [Object] WlanSetProfile([UInt32]$CH,[Guid]$IG,[UInt32]$F,[IntPtr]$PX,[IntPtr]$PS,
                            [Bool]$O,[IntPtr]$PR,[IntPtr]$pdw)
    {
        # // ____________________________________________________________________________
        # // | CH: ClientHandle | IG: InterfaceGuid | F: Flags | PX: ProfileXml         |
        # // | PS: ProfileSecurity | O: Overwrite | PR: pReserved | PDW: pdwReasonCode |
        # // ----------------------------------------------------------------------------

        Return "[WiFi.ProfileManagement]::WlanSetProfile($CH,$IG,$F,$PX,$PS,$O,$PR,$PDW)" | IEX
    }
    [Void] WlanDeleteProfile([IntPtr]$CH,[Guid]$IG,[String]$PN,[IntPtr]$PR)
    {
        # // ____________________________________________________________________
        # // | CH: ClientHandle | IG: InterfaceGuid | PN: ProfileName | PR: pReserved |
        # // --------------------------------------------------------------------

        "[WiFi.ProfileManagement]::WlanDeleteProfile($CH,$IG,$PN,$PR)" | IEX
    }
    [Void] WlanDisconnect([IntPtr]$HCH,[Guid]$IG,[IntPtr]$PR)
    {
        # // ____________________________________________________
        # // | HCH: hClientHandle | IG: InterfaceGuid | PR: pReserved |
        # // ----------------------------------------------------

        "[WiFi.ProfileManagement]::WlanDisconnect($HCH,$IG,$PR)" | IEX
    }
    [Void] WlanConnect([IntPtr]$HCH,[Guid]$IG,[Object]$CP,[IntPtr]$PR)
    {
        # // ____________________________________________________________________________________________
        # // | HCH: hClientHandle | IG: InterfaceGuid | CP: ConnectionParameters | PR: pReserved |
        # // --------------------------------------------------------------------------------------------

        "[WiFi.ProfileManagement]::WlanConnect($HCH,$IG,$CP,$PR" | IEX
    }
    [String] WiFiReasonCode([IntPtr]$RC)
    {
        # // ___________________
        # // | RC: ReasonCode |
        # // -------------------

        $SB          = [Text.StringBuilder]::New(1024)
        $result      = $This.WlanReasonCodeToString($RC.ToInt32(),$SB.Capacity,$SB,[IntPtr]::zero)

        If ($result -ne 0)
        {
            Return $This.Win32Exception($result)
        }

        Return $SB.ToString()
    }
    [IntPtr] NewWifiHandle()
    {
```

```
        # // ------------------------------------------------------------
        # // | MC: MaxClient | NV: NegotiatedVersion | CH: ClientHandle |
        # // ------------------------------------------------------------

        $MC       = 2
        [Ref] $NV = 0
        $CH       = [IntPtr]::zero
        $result   = $This.WlanOpenHandle($MC,[IntPtr]::Zero,$NV,[Ref]$CH)

        If ($result -eq 0)
        {
            Return $CH
        }
        Else
        {
            Throw $This.Win32Exception($Result)
        }
    }
    [Void] RemoveWifiHandle([IntPtr]$CH)
    {
        # // --------------------
        # // | CH: ClientHandle |
        # // --------------------

        $Result = $This.WlanCloseHandle($CH,[IntPtr]::zero)

        If ($Result -ne 0)
        {
            Throw $This.Win32Exception($Result)
        }
    }
    [Object] GetWiFiInterfaceGuid([String]$WFAN)
    {
        # // ------------------------------------------------------------------------
        # // | WFAN: WiFiAdapterName | IG: InterfaceGuid | WFAI: WiFiAdapterInfo |
        # // ------------------------------------------------------------------------

        $IG   = $Null
        Switch ([Environment]::OSVersion.Version -ge [Version]6.2)
        {
            $True
            {
                $IG   = Get-NetAdapter -Name $WFAN -EA 0 | % InterfaceGuid
            }
            $False
            {
                $WFAI = Get-WmiObject Win32_NetworkAdapter | ? NetConnectionID -eq $WFAN
                $IG   = Get-WmiObject Win32_NetworkAdapterConfiguration | ? {

                    $_.Description -eq $WFAI.Name | % SettingID
                }
            }
        }

        Return [System.Guid]$IG
    }
    [Object[]] GetWiFiInterface()
    {
        # // ------------------------------------------------------------------------------------------
        # // | IL: InterfaceListPtr | CH: ClientHandle | WFIL: WiFiInterfaceList | IF: Interface |
        # // ------------------------------------------------------------------------------------------

        $IL            = 0
        $CH            = $This.NewWiFiHandle()
        $This.Adapters = $This.RefreshAdapterList()
        $Return        = @( )
        Try
        {
            [Void]$This.WlanEnumInterfaces($CH,[IntPtr]::zero,[ref]$IL)
            $WFIL = $This.WlanInterfaceList($IL)
            ForEach ($wlanInterfaceInfo in $WFIL.wlanInterfaceInfo)
```

```
            {
                $Info      = $this.WlanInterfaceInfo($wlanInterfaceInfo)
                $Interface = $This.Adapters | ? InterfaceDescription -eq $Info.Description
                $Return   += [InterfaceObject]::New($Info,$Interface)
            }
        }
        Catch
        {
            Write-Host "No wireless interface(s) found"
            $Return += $Null
        }
        Finally
        {
            $This.RemoveWiFiHandle($CH)
        }

        Return @($Return)
    }
    [Object[]] GetWiFiProfileList([String]$Name)
    {
        # // ------------------------------------------------------------------------------
        # // | PLP: ProfileListPointer | IF: Interface | CH: ClientHandle | PL: ProfileList |
        # // ------------------------------------------------------------------------------

        $PLP    = 0
        $IF     = $This.GetWifiInterface() | ? Name -match $Name
        $CH     = $This.NewWifiHandle()
        $Return = @( )

        $This.WlanGetProfileList($CH,$IF.GUID,[IntPtr]::zero,[Ref]$PLP)

        $PL     = $This.WlanGetProfileListFromPtr($PLP)

        ForEach ($ProfileName in $PL)
        {
            $Item           = [WiFiProfile]::New($IF,$ProfileName)
            $Item.Detail    = $This.GetWiFiProfileInfo($Item.Name,$IF.Guid)
            $Return        += $Item
        }

        $This.RemoveWiFiHandle($CH)

        Return $Return
    }
    [Object] GetWiFiProfileInfo([String]$PN,[Guid]$IG,[Int16]$WPF)
    {
        # // ------------------------------------------------------------------------------------
        # // | PN: ProfileName | IG: InterfaceGuid | WPF: WlanProfileFlags | CH: ClientHandle |
        # // ------------------------------------------------------------------------------------

        [IntPtr]$CH              = $This.NewWifiHandle()
        $WlanProfileFlagsInput   = $WPF
        $Return                  = $This.WiFiProfileInfo($PN,$IG,$CH,$WlanProfileFlagsInput)
        $This.RemoveWiFiHandle($CH)
        Return $Return
    }
    [Object] GetWifiProfileInfo([String]$PN,[Guid]$IG)
    {
        # // ------------------------------------------------------------
        # // | PN: ProfileName | IG: InterfaceGuid | CH: ClientHandle |
        # // ------------------------------------------------------------

        [IntPtr]$CH            = $This.NewWifiHandle()
        $WlanProfileFlagsInput = 0
        $Return                = $This.WiFiProfileInfo($PN,$IG,$CH,$WlanProfileFlagsInput)
        $This.RemoveWiFiHandle($CH)
        Return $Return
    }
    [Object] WiFiProfileInfo([String]$PN,[Guid]$IG,[IntPtr]$CH,[Int16]$WPFI)
    {
        # // ------------------------------------------------------------------------------------
        # // | PN: ProfileName | IG: IntGuid | CH: ClientHandle | WPFI: WlanProfileFlagsInput |
```

```powershell
        # // | PS: pstrProfileXml | WA: WlanAccess | WlanPF: WlanProfileFlags | PW: Password  |
        # // | CHSSID: ConnectHiddenSSID | EAP: EapType | X: XmlPtr | SN: ServerNames         |
        # // | TRCA: TrustedRootCA | WP: WlanProfile
        # // ---------------------------------------------------------------------------------------

        [String] $PS = $null
        $WA          = 0
        $WlanPF      = $WPFI
        $result      = $This.WlanGetProfile($CH,$IG,$PN,[IntPtr]::Zero,[Ref]$PS,[Ref]$WlanPF,[Ref]$WA)
        $PW          = $Null
        $CHSSID      = $Null
        $Eap         = $Null
        $xmlPtr      = $Null
        $SN          = $Null
        $TRCA        = $Null
        $Return      = $Null

        If ($result -ne 0)
        {
            Return $This.Win32Exception($Result)
        }

        $WP          = [Xml]$PS

        # // ------------------
        # // | Parse password |
        # // ------------------

        If ($WPFI -eq 13)
        {
            $PW      = $WP.WLANProfile.MSM.security.sharedKey.keyMaterial
        }
        If ($WPFI -ne 13)
        {
            $PW            = $Null
        }

        # // ---------------------------
        # // | Parse nonBroadcast flag |
        # // ---------------------------

        If ([bool]::TryParse($WP.WLANProfile.SSIDConfig.nonBroadcast,[Ref]$null))
        {
            $CHSSID = [bool]::Parse($WP.WLANProfile.SSIDConfig.nonBroadcast)
        }
        Else
        {
            $CHSSID = $false
        }

        # // ------------------
        # // | Parse EAP type |
        # // ------------------

        If ($WP.WLANProfile.MSM.security.authEncryption.useOneX -eq $true)
        {
            $WP.WLANProfile.MSM.security.OneX.EAPConfig.EapHostConfig.EapMethod.Type.InnerText | % {

                $EAP   = Switch ($_) { 13 { 'TLS'  } 25 { 'PEAP' }  Default { 'Unknown' } }
                                        # 13: EAP-TLS | 25: EAP-PEAP (MSCHAPv2)
            }
        }
        Else
        {
            $EAP = $null
        }

        # // --------------------------------
        # // | Parse Validation Server Name |
        # // --------------------------------

        If (!!$Eap)
```

```
        {
            $Cfg = $WP.WLANProfile.MSM.security.OneX.EAPConfig.EapHostConfig.Config
            Switch ($Eap)
            {
                PEAP
                {

                    $SN   = $Cfg.Eap.EapType.ServerValidation.ServerNames
                }

                TLS
                {
                    $Node = $Cfg.SelectNodes("//*[local-name()='ServerNames']")
                    $SN   = $Node[0].InnerText
                }
            }
        }

        # // ___________________________________
        # // | Parse Validation TrustedRootCA |
        # // -----------------------------------

        If (!!$EAP)
        {
            $Cfg = $WP.WLANProfile.MSM.security.OneX.EAPConfig.EapHostConfig.Config
            Switch ($EAP)
            {
                PEAP
                {
                    $TRCA = $Cfg.Eap.EapType.ServerValidation.TrustedRootCA.Replace(' ','') | % ToLower
                }
                TLS
                {
                    $Node = $Cfg.SelectNodes("//*[local-name()='TrustedRootCA']")
                    $TRCA = $Node[0].InnerText.Replace(' ','') | % ToLower
                }
            }
        }

        $Return                   = $This.WlanProfileInfoObject()
        $Return.ProfileName       = $WP.WlanProfile.SSIDConfig.SSID.name
        $Return.ConnectionMode    = $WP.WlanProfile.ConnectionMode
        $Return.Authentication    = $WP.WlanProfile.MSM.Security.AuthEncryption.Authentication
        $Return.Encryption        = $WP.WlanProfile.MSM.Security.AuthEncryption.Encryption
        $Return.Password          = $PW
        $Return.ConnectHiddenSSID = $CHSSID
        $Return.EAPType           = $EAP
        $Return.ServerNames       = $SN
        $Return.TrustedRootCA     = $TRCA
        $Return.Xml               = $PS

        $xmlPtr                   = [System.Runtime.InteropServices.Marshal]::StringToHGlobalAuto($PS)
        $This.WlanFreeMemory($xmlPtr)

        Return $Return
    }
    [Object] GetWiFiConnectionParameter([String]$PN,[String]$CM,[String]$D,[String]$F)
    {
        # // ___________________________________________________________________
        # // | PN: ProfileName | CM: ConnectionMode | D: Dot11BssType | F: Flag |
        # // -------------------------------------------------------------------

        Return $This.WifiConnectionParameter($PN,$CM,$D,$F)
    }
    [Object] GetWiFiConnectionParameter([String]$PN,[String]$CM,[String]$D)
    {
        # // ___________________________________________________
        # // | PN: ProfileName | CM: ConnectionMode | D: Dot11BssType |
        # // ---------------------------------------------------

        Return $This.WifiConnectionParameter($PN,$CM,$D,"Default")
    }
```

```
        [Object] GetWiFiConnectionParameter([String]$PN,[String]$CM)
        {
            # // ----------------------------------------
            # // | PN: ProfileName | CM: ConnectionMode |
            # // ----------------------------------------

            Return $This.WifiConnectionParameter($PN,$CM,"Any","Default")
        }
        [Object] GetWiFiConnectionParameter([String]$PN)
        {
            # // --------------------
            # // | PN: ProfileName |
            # // --------------------

            Return $This.WifiConnectionParameter($PN,"Profile","Any","Default")
        }
        [Object] WifiConnectionParameter([String]$PN,[String]$CM,[String]$D,[String]$F)
        {
            # // ------------------------------------------------------------
            # // | PN: ProfileName | CM: ConnectionMode | D: Dot11BssType |
            # // | F: Flag | CMR: ConnectionModeResolver | P: Profile     |
            # // | CP: ConnectionParameters                                 |
            # // ------------------------------------------------------------

            Try
            {

                $CMR                    =  [ConnectionModeResolver]::New()

                $CP                     = $This.WlanConnectionParams()
                $CP.StrProfile          = $PN
                $CP.WlanConnectionMode  = $This.WlanConnectionMode($CMR[$CM])
                $CP.Dot11BssType        = $This.WlanDot11BssType($D)
                $CP.dwFlags             = $This.WlanConnectionFlag($F)
            }
            Catch
            {
                Throw "An error occurred while setting connection parameters"
            }

            Return $CP
        }
        [Object] FormatXml([Object]$Content)
        {
            $StringWriter           = [System.IO.StringWriter]::New()
            $XmlWriter              = [System.Xml.XmlTextWriter]::New($StringWriter)
            $XmlWriter.Formatting   = "Indented"
            $XmlWriter.Indentation = 4
            ([Xml]$Content).WriteContentTo($XmlWriter)
            $XmlWriter.Flush()
            $StringWriter.Flush()
            Return $StringWriter.ToString()
        }
        [Object] XmlTemplate([UInt32]$Type)
        {
            $xList = (0,"Personal"),(1,"EapPeap"),(2,"EapTls") | % { "($($_[0]): $($_[1]))" }

            If ($Type -notin 0..2)
            {
                Throw "Select a valid type: [$($xList -join ", ")]"
            }

            $P = "http://www.microsoft.com/provisioning"

            $xProfile = Switch ($Type)
            {
                0 # WiFiProfileXmlPersonal
                {
                    '<?xml version="1.0"?>',('<WLANProfile xmlns="http://www.microsoft.com/networking/WLAN/pr'+
                    'ofile/v1">'),'<name>{0}</name>','<SSIDConfig>','<SSID>','<hex>{1}</hex>',('<name>{0}</na'+
                    'me>'),'</SSID>','</SSIDConfig>','<connectionType>ESS</connectionType>',('<connectionMode'+
                    '>{2}</connectionMode>'),'<MSM>','<security>','<authEncryption>',('<authentication>{3}</a'+
```

```
                'uthentication>'),'<encryption>{4}</encryption>','<useOneX>false</useOneX>',('</authEncry'+
                'ption>'),'<sharedKey>','<keyType>passPhrase</keyType>','<protected>false</protected>',
                '<keyMaterial>{5}</keyMaterial>','</sharedKey>','</security>','</MSM>',('<MacRandomizatio'+
                'n xmlns="http://www.microsoft.com/networking/WLAN/profile/v3">'),('<enableRandomization>'+
                'false</enableRandomization>'),"</MacRandomization>",'</WLANProfile>'
            }
            1 # WiFiProfileXmlEapPeap
            {
                '<?xml version="1.0"?>',('<WLANProfile xmlns="http://www.microsoft.com/networking/WLAN/pr'+
                'ofile/v1">'),'<name>{0}</name>','<SSIDConfig>','<SSID>','<hex>{1}</hex>',('<name>{0}</na'+
                'me>'),'</SSID>',('</SSIDConfig>'),'<connectionType>ESS</connectionType>',('<connectionMo'+
                'de>{2}</connectionMode>'),'<MSM>','<security>','<authEncryption>',('<authentication>{3}<'+
                '/authentication>'),'<encryption>{4}</encryption>','<useOneX>true</useOneX>',('</authEncr'+
                'yption>'),'<PMKCacheMode>enabled</PMKCacheMode>','<PMKCacheTTL>720</PMKCacheTTL>',('<PMK'+
                'CacheSize>128</PMKCacheSize>'),'<preAuthMode>disabled</preAuthMode>',('<OneX xmlns="http'+
                '://www.microsoft.com/networking/OneX/v1">'),'<authMode>machineOrUser</authMode>',('<EAPC'+
                'onfig>'),"<EapHostConfig xmlns='$P/EapHostConfig'>",'<EapMethod>',("<Type xmlns='$P/EapH"+
                "ostConfig'>25</Type>"),"<VendorId xmlns='$P/EapCommon'>0</VendorId>",("<VendorType xmlns"+
                "='$P/EapCommon'>0</VendorType>"),"<AuthorId xmlns='$P/EapCommon'>0</AuthorId>",('</EapMe'+
                'thod>'),"<Config xmlns='$P/EapHostConfig'>",("<Eap xmlns='$P/BaseEapConnectionProperties"+
                "V1'>"),'<Type>25</Type>',"<EapType xmlns='$P/MsPeapConnectionPropertiesV1'>",('<ServerVa'+
                'lidation>'),('<DisableUserPromptForServerValidation>false</DisableUserPromptForServerVal'+
                'idation>'),'<ServerNames></ServerNames>','<TrustedRootCA></TrustedRootCA>',('</ServerVal'+
                'idation>'),'<FastReconnect>true</FastReconnect>',('<InnerEapOptional>false</InnerEapOpti'+
                'onal>'),"<Eap xmlns='$P/BaseEapConnectionPropertiesV1'>",'<Type>26</Type>',("<EapType xm"+
                "lns='$P/MsChapV2ConnectionPropertiesV1'>"),('<UseWinLogonCredentials>false</UseWinLogonC'+
                'redentials>'),'</EapType>','</Eap>',('<EnableQuarantineChecks>false</EnableQuarantineChe'+
                'cks>'),'<RequireCryptoBinding>false</RequireCryptoBinding>','<PeapExtensions>',("<Perfor"+
                "mServerValidation xmlns='$P/MsPeapConnectionPropertiesV2'>true</PerformServerValidation>"+
                ""),"<AcceptServerName xmlns='$P/MsPeapConnectionPropertiesV2'>true</AcceptServerName>",
                "<PeapExtensionsV2 xmlns='$P/MsPeapConnectionPropertiesV2'>",("<AllowPromptingWhenServerC"+
                "ANotFound xmlns='$P/MsPeapConnectionPropertiesV3'>true</AllowPromptingWhenServerCANotFou"+
                "nd>"),'</PeapExtensionsV2>','</PeapExtensions>','</EapType>','</Eap>','</Config>',('</Ea'+
                'pHostConfig>'),'</EAPConfig>','</OneX>','</security>','</MSM>',('<MacRandomization xmlns'+
                '="http://www.microsoft.com/networking/WLAN/profile/v3">'),("<enableRandomization>false</"+
                "enableRandomization>"),"</MacRandomization>",'</WLANProfile>'
            }
            2 # WiFiProfileXmlEapTls
            {
                '<?xml version="1.0"?>',('<WLANProfile xmlns="http://www.microsoft.com/networking/WLAN/pr'+
                'ofile/v1">'),'<name>{0}</name>','<SSIDConfig>','<SSID>','<hex>{1}</hex>',('<name>{0}</na'+
                'me>'),'</SSID>','</SSIDConfig>','<connectionType>ESS</connectionType>',('<connectionMode'+
                '>{2}</connectionMode>'),'<MSM>','<security>','<authEncryption>',('<authentication>{3}</a'+
                'uthentication>'),'<encryption>{4}</encryption>','<useOneX>true</useOneX>',('</authEncryp'+
                'tion>'),'<PMKCacheMode>enabled</PMKCacheMode>','<PMKCacheTTL>720</PMKCacheTTL>',('<PMKCa'+
                'cheSize>128</PMKCacheSize>'),'<preAuthMode>disabled</preAuthMode>',('<OneX xmlns="http:/'+
                '/www.microsoft.com/networking/OneX/v1">'),'<authMode>machineOrUser</authMode>',('<EAPCon'+
                'fig>'),"<EapHostConfig xmlns='$P/EapHostConfig'>",'<EapMethod>',("<Type xmlns='$P/EapHos"+
                "tConfig'>13</Type>"),"<VendorId xmlns='$P/EapCommon'>0</VendorId>",("<VendorType xmlns='"+
                "$P/EapCommon'>0</VendorType>"),"<AuthorId xmlns='$P/EapCommon'>0</AuthorId>",('</EapMeth'+
                'od>'),("<Config xmlns:baseEap='$P/BaseEapConnectionPropertiesV1' xmlns:eapTls='$P/EapTls"+
                "ConnectionPropertiesV1'>"),'<baseEap:Eap>','<baseEap:Type>13</baseEap:Type>',('<eapTls:E'+
                'apType>'),'<eapTls:CredentialsSource>','<eapTls:CertificateStore />',('</eapTls:Credenti'+
                'alsSource>'),'<eapTls:ServerValidation>',('<eapTls:DisableUserPromptForServerValidation>'+
                'false</eapTls:DisableUserPromptForServerValidation>'),('<eapTls:ServerNames></eapTls:Ser'+
                'verNames>'),'<eapTls:TrustedRootCA></eapTls:TrustedRootCA>','</eapTls:ServerValidation>',
                '<eapTls:DifferentUsername>false</eapTls:DifferentUsername>','</eapTls:EapType>',('</base'+
                'Eap:Eap>'),'</Config>','</EapHostConfig>','</EAPConfig>','</OneX>','</security>','</MSM>',
                '<MacRandomization xmlns="http://www.microsoft.com/networking/WLAN/profile/v3">',("<enabl"+
                "eRandomization>false</enableRandomization>"),"</MacRandomization>",'</WLANProfile>'
            }
        }

        Return $This.FormatXml($xProfile)
    }
    [String] Hex([String]$PN)
    {
        # // ------------------
        # // | PN: ProfileName |
        # // ------------------

        Return ([Char[]]$PN | % { '{0:X}' -f [Int]$_ }) -join ''
```

```
    }
    [String] NewWiFiProfileXmlPsk([String]$PN,[String]$CM='Auto',[String]$A='WPA2PSK',[String]$E='AES',
                                  [SecureString]$PW)
    {
        # // ------------------------------------------------------------------------------------------
        # // | PN: ProfileName | CM: ConnectionMode | A: Authentication | E: Encryption | PW: Password |
        # // | PP: PlainPassword | PX: ProfileXml | SS: SecureStringToBstr | RN: RefNode | XN: XmlNode |
        # // ------------------------------------------------------------------------------------------

        $PP           = $Null
        $PX           = $Null
        $Hex          = $This.Hex($PN)
        Try
        {
            If ($PW)
            {
                $SS   = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($PW)
                $PW   = [System.Runtime.InteropServices.Marshal]::PtrToStringAuto($SS)
            }

            $PX       = [XML]($This.XmlTemplate(0) -f $PN, $Hex, $CM, $A, $E, $PP)
            If (!$PP)
            {
                $Null = $PX.WLANProfile.MSM.security.RemoveChild($PX.WLANProfile.MSM.security.sharedKey)
            }

            If ($A -eq 'WPA3SAE')
            {
                # // ---------------------------------------------
                # // | Set transition mode as true for WPA3-SAE |
                # // ---------------------------------------------

                $N  = [System.Xml.XmlNamespaceManager]::new($PX.NameTable)
                $N.AddNamespace('WLANProfile',$PX.DocumentElement.GetAttribute('xmlns'))
                $RN = $PX.SelectSingleNode('//WLANProfile:authEncryption', $N)
                $XN = $PX.CreateElement('transitionMode',
                                        'http://www.microsoft.com/networking/WLAN/profile/v4')
                $XN.InnerText = 'True'
                $null         = $RN.AppendChild($XN)
            }

            Return $This.FormatXml($PX.OuterXml)
        }
        Catch
        {
            Throw "Failed to create a new profile"
        }
    }
    [String] NewWifiProfileXmlEap([String]$PN,[String]$CM='Auto',[String]$A='WPA2PSK',[String]$E='AES',
                                  [String]$Eap,[String[]]$SN,[String]$TRCA)
    {
        # // ------------------------------------------------------------------------------------------
        # // | PN: ProfileName | CM: ConnectionMode | A: Authentication | E: Encryption | EAP: EapType |
        # // | SN: ServerNames | TRCA: TrustedRootCa | PX: ProfileXml |  RN: RefNode | XN: XmlNode     |
        # // ------------------------------------------------------------------------------------------

        $Px  = $Null
        $Hex = $This.Hex($PN)
        Try
        {
            If ($Eap -eq 'PEAP')
            {
                $Px = [Xml]($This.XmlTemplate(1) -f $PN, $Hex, $CM, $A, $E)
                $Cfg = $PX.WLANProfile.MSM.security.OneX.EAPConfig.EapHostConfig.Config

                If ($SN)
                {
                    $Cfg.Eap.EapType.ServerValidation.ServerNames = $SN
                }

                If ($TRCA)
                {
```

```
                    $Cfg.Eap.EapType.ServerValidation.TrustedRootCA = $TRCA.Replace('..','$& ')
                }
            }
            ElseIf ($Eap -eq 'TLS')
            {
                $PX  = [Xml]($This.XmlTemplate(2) -f $PN, $Hex, $CM, $A, $E)
                $Cfg = $PX.WLANProfile.MSM.security.OneX.EapConfig.EapHostConfig.Config

                If ($SN)
                {
                    $Node = $Cfg.SelectNodes("//*[local-name()='ServerNames']")
                    $Node[0].InnerText = $SN
                }

                If ($TRCA)
                {
                    $Node = $Cfg.SelectNodes("//*[local-name()='TrustedRootCA']")
                    $Node[0].InnerText = $TRCA.Replace('..','$& ')
                }
            }

            If ($A -eq 'WPA3SAE')
            {
                # // ------------------------------------------
                # // | Set transition mode as true for WPA3-SAE |
                # // ------------------------------------------

                $N = [System.Xml.XmlNamespaceManager]::new($PX.NameTable)
                $N.AddNamespace('WLANProfile', $PX.DocumentElement.GetAttribute('xmlns'))
                $RN = $PX.SelectSingleNode('//WLANProfile:authEncryption', $N)
                $XN = $PX.CreateElement('transitionMode',
                                        'http://www.microsoft.com/networking/WLAN/profile/v4')
                $XN.InnerText = 'true'
                $null = $RN.AppendChild($XN)
            }

            Return $This.FormatXml($PX.OuterXml)
        }
        Catch
        {
            Throw "Failed to create a new profile"
        }
    }
    [Object] NewWiFiProfilePsk([String]$PN,[String]$PW,[String]$WFAN)
    {
        # // ------------------------------------------------------------------------------
        # // | PN: ProfileName | PW: Password | WFAN: WiFiAdapterName | CM: ConnectionMode |
        # // | A: Authentication | E: Encryption | PT: ProfileTemp                         |
        # // ------------------------------------------------------------------------------

        $CM = 'auto'
        $A  = 'WPA2PSK'
        $E  = 'AES'
        $PT = $This.NewWifiProfileXmlPsk($PN,$CM,$A,$E,$PW)
        Return $This.NewWifiProfile($PT,$WFAN)
    }
    [Object] NewWiFiProfilePsk([String]$PN,[String]$PW,[String]$CM,[String]$WFAN)
    {
        # // ------------------------------------------------------------------------------
        # // | PN: ProfileName | PW: Password | WFAN: WiFiAdapterName | CM: ConnectionMode |
        # // | A: Authentication | E: Encryption | PT: ProfileTemp                         |
        # // ------------------------------------------------------------------------------

        $A  = 'WPA2PSK'
        $E  = 'AES'
        $PT = $This.NewWifiProfileXmlPsk($PN,$CM,$A,$E)
        Return $This.NewWifiProfile($PT,$WFAN)
    }
    [Object] NewWiFiProfilePsk([String]$PN,[String]$PW,[String]$CM,[String]$A,[String]$WFAN)
    {
        # // ------------------------------------------------------------------------------
        # // | PN: ProfileName | PW: Password | WFAN: WiFiAdapterName | CM: ConnectionMode |
```

```
            # // | A: Authentication | E: Encryption | PT: ProfileTemp                          |
            # // ------------------------------------------------------------------------------

            $E  = 'AES'
            $PT = $This.NewWifiProfileXmlPsk($PN,$CM,$A,$E,$WFAN)
            Return $This.NewWifiProfile($PT,$WFAN)
        }
        [Object] NewWiFiProfilePsk([String]$PN,[String]$PW,[String]$CM,[String]$A,[String]$E,[String]$WFAN)
        {
            # // ------------------------------------------------------------------------------
            # // | PN: ProfileName | PW: Password | CM: ConnectionMode | A: Authentication |
            # // | E: Encryption | WFAN: WiFiAdapterName | PT: ProfileTemp                  |
            # // ------------------------------------------------------------------------------

            $PT     = $This.NewWifiProfileXmlPsk($PN,$CM,$A,$E,$WFAN)
            Return $This.NewWifiProfile($PT,$WFAN)
        }
        [Object] NewWifiProfileEap([String]$PN,[String]$EAP,[String]$WFAN)
        {
            # // ------------------------------------------------------------------------------
            # // | PN: ProfileName | EAP: EapType | WFAN: WiFiAdapterName | CM: ConnectionMode  |
            # // | A: Authentication | E: Encryption | SN: ServerNames | TRCA: TrustedRootCA    |
            # // | PT: ProfileTemp                                                              |
            # // ------------------------------------------------------------------------------

            $CM   = 'Auto'
            $A    = 'WPA2PSK'
            $E    = 'AES'
            $SN   = ''
            $TRCA = $Null
            $PT   = $This.NewWifiProfileXmlEap($PN,$CM,$A,$E,$EAP,$SN,$TRCA)
            Return $This.NewWifiProfile($PT,$WFAN)
        }
        [Object] NewWifiProfileEap([String]$PN,[String]$CM,[String]$EAP,[String]$WFAN)
        {
            # // ------------------------------------------------------------------------------
            # // | PN: ProfileName | EAP: EapType | WFAN: WiFiAdapterName | CM: ConnectionMode  |
            # // | A: Authentication | E: Encryption | SN: ServerNames | TRCA: TrustedRootCA    |
            # // | PT: ProfileTemp                                                              |
            # // ------------------------------------------------------------------------------

            $A    = 'WPA2PSK'
            $E    = 'AES'
            $SN   = ''
            $TRCA = $Null
            $PT   = $This.NewWifiProfileXmlEap($PN,$CM,$A,$E,$EAP,$SN,$TRCA)
            Return $This.NewWifiProfile($PT,$WFAN)
        }
        [Object] NewWifiProfileEap([String]$PN,[String]$CM,[String]$A,[String]$EAP,[String]$WFAN)
        {
            # // ------------------------------------------------------------------------------
            # // | PN: ProfileName | EAP: EapType | WFAN: WiFiAdapterName | CM: ConnectionMode  |
            # // | A: Authentication | E: Encryption | SN: ServerNames | TRCA: TrustedRootCA    |
            # // | PT: ProfileTemp                                                              |
            # // ------------------------------------------------------------------------------

            $E    = 'AES'
            $SN   = ''
            $TRCA = $Null
            $PT   = $This.NewWifiProfileXmlEap($PN,$CM,$A,$E,$EAP,$SN,$TRCA)
            Return $This.NewWifiProfile($PT,$WFAN)
        }
        [Object] NewWifiProfileEap([String]$PN,[String]$CM,[String]$A,[String]$E,[String]$EAP,[String]$WFAN)
        {
            # // ------------------------------------------------------------------------------
            # // | PN: ProfileName | EAP: EapType | WFAN: WiFiAdapterName | CM: ConnectionMode  |
            # // | A: Authentication | E: Encryption | SN: ServerNames | TRCA: TrustedRootCA    |
            # // | PT: ProfileTemp                                                              |
            # // ------------------------------------------------------------------------------

            $SN   = ''
            $TRCA = $Null
```

```
        $PT    = $This.NewWifiProfileXmlEap($PN,$CM,$A,$E,$EAP,$SN,$TRCA)
        Return $This.NewWifiProfile($PT,$WFAN)
    }
    [Object] NewWifiProfileEap([String]$PN,[String]$CM,[String]$A,[String]$E,[String]$Eap,[String[]]$SN,
                               [String]$WFAN)
    {
        # // ------------------------------------------------------------------------------
        # // | PN: ProfileName | EAP: EapType | WFAN: WiFiAdapterName | CM: ConnectionMode  |
        # // | A: Authentication | E: Encryption | SN: ServerNames | TRCA: TrustedRootCA    |
        # // | PT: ProfileTemp                                                              |
        # // ------------------------------------------------------------------------------

        $TRCA     = $Null
        $PT       = $This.NewWifiProfileXmlEap($PN,$CM,$A,$E,$EAP,$SN,$TRCA)
        Return $This.NewWifiProfile($PT,$WFAN)
    }
    [Object] NewWifiProfileEap([String]$PN,[String]$CM,[String]$A,[String]$E,[String]$Eap,[String[]]$SN,
                               [String]$TRCA,[String]$WFAN)
    {
        # // ------------------------------------------------------------------------------
        # // | PN: ProfileName | EAP: EapType | WFAN: WiFiAdapterName | CM: ConnectionMode  |
        # // | A: Authentication | E: Encryption | SN: ServerNames | TRCA: TrustedRootCA    |
        # // | PT: ProfileTemp                                                              |
        # // ------------------------------------------------------------------------------

        $PT       = $This.NewWifiProfileXmlEap($PN,$CM,$A,$E,$EAP,$SN,$TRCA)
        Return $This.NewWifiProfile($PT,$WFAN)
    }
    [Object] NewWifiProfileXml([String]$PX,[String]$WFAN,[Bool]$O)
    {
        # // ------------------------------------------------------------
        # // | PX: ProfileXml | WFAN: WiFiAdapterName | O: Overwrite |
        # // ------------------------------------------------------------

        Return $This.NewWifiProfile($PX,$WFAN)
    }
    NewWifiProfile([String]$PX,[String]$WFAN,[Bool]$O)
    {
        # // ------------------------------------------------------------------------------
        # // | PX: ProfileXml | WFAN: WiFiAdapterName | O: Overwrite | IG: InterfaceGuid |
        # // | CH: ClientHandle | F: Flags | PP: ProfilePointer                           |
        # // | RSC: ReasonCode | RSCM: ReasonCodeMessage                                  |
        # // | RTC: ReturnCode | RTCM: ReturnCodeMessage                                  |
        # // ------------------------------------------------------------------------------

        Try
        {
            $IG   = $This.GetWiFiInterfaceGuid($WFAN)
            $CH   = $This.NewWiFiHandle()
            $F    = 0
            $RSC  = [IntPtr]::Zero
            $PP   = [System.Runtime.InteropServices.Marshal]::StringToHGlobalUni($PX)
            $RTC  = $This.WlanSetProfile($CH,[Ref]$IG,$F,$PP,[IntPtr]::Zero,$O,[IntPtr]::Zero,[Ref]$RSC)
            $RTCM = $This.Win32Exception($RTC)
            $RSCM = $This.WiFiReasonCode($RSC)

            If ($RTC -eq 0)
            {
                Write-Verbose -Message $RTCM
            }
            Else
            {
                Throw $RTCM
            }

            Write-Verbose -Message $RSCM
        }
        Catch
        {
            Throw "Failed to create the profile"
        }
        Finally
```

```
        {
            If ($CH)
            {
                $This.RemoveWiFiHandle($CH)
            }
        }
    }
    RemoveWifiProfile([String]$PN)
    {
        # // ______________________________________
        # // | PN: ProfileName | CH: ClientHandle |
        # // --------------------------------------

        $CH = $This.NewWiFiHandle()
        $This.WlanDeleteProfile($CH,[Ref]$This.Selected.Guid,$PN,[IntPtr]::Zero)
        $This.RemoveWifiHandle($CH)
    }
    Select([String]$D)
    {
        # // __________________
        # // | D: Description |
        # // ------------------

        # // ____________________________________________
        # // | Select the adapter from its description |
        # // --------------------------------------------

        $This.Selected = $This.GetWifiInterface() | ? Description -eq $D
        $This.Update()
    }
    Unselect()
    {
        $This.Selected = $Null
        $This.Update()
    }
    Disconnect()
    {
        If (!$This.Selected)
        {
            Write-Host "No network selected"
        }
        If ($This.Selected.State -eq "CONNECTED")
        {
            $CH                     = $This.NewWiFiHandle()
            $This.WlanDisconnect($CH,[Ref]$This.Selected.Guid,[IntPtr]::Zero)
            $This.RemoveWifiHandle($CH)

            $This.Connected                 = $Null

            $Splat                          = @{

                Type    = "Image"
                Mode    = 2
                Image   = $This.OEMLogo
                Message = "Disconnected: $($This.Selected.SSID)"
            }

            Show-ToastNotification @Splat

            $Link                           = $This.Selected.Description
            $This.Unselect()
            $This.Select($Link)
        }
    }
    Connect([String]$SSID)
    {
        If (!$This.Selected)
        {
            Write-Host "Must select an active interface"
        }

        If ($This.Selected)
```

```
        {
            $Link                               = $This.Selected.Description
            $This.Unselect()
            $This.Select($Link)

            If ($This.Selected.State -ne "CONNECTED")
            {
                $Result = $This.GetWifiProfileInfo($SSID,$This.Selected.Guid)
                If ($Result)
                {
                    $Param = $This.GetWiFiConnectionParameter($SSID)
                    $CH    = $This.NewWiFiHandle()
                    $This.WlanConnect($CH,[Ref]$This.Selected.Guid,[Ref]$Param,[IntPtr]::Zero)
                    $This.RemoveWifiHandle($CH)

                    $Link   = $This.Selected.Description
                    $This.Unselect()
                    $This.Select($Link)

                    $This.Update()

                    $Splat                              = @{

                        Type    = "Image"
                        Mode    = 2
                        Image   = $This.OEMLogo
                        Message = "Connected: $SSID"
                    }

                    Show-ToastNotification @Splat
                }
                If (!$Result)
                {
                    $Network = $This.Output.SelectedItem
                    If ($Network.Authentication -match "psk")
                    {
                        $This.Passphrase($Network)
                    }
                    Else
                    {
                        Write-Host "Eas/Peap not yet implemented"
                    }
                }
            }
        }
    }
    Passphrase([Object]$NW)
    {
        $PW    = Read-Host -AsSecureString -Prompt "Enter passphrase for Network: [$($NW.SSID)]"
        $A     = $Null
        $E     = $Null

        If ($NW.Authentication -match "RsnaPsk")
        {
            $A = "WPA2PSK"
        }
        If ($NW.Encryption -match "Ccmp")
        {
            $E = "AES"
        }

        $PX    = $This.NewWifiProfileXmlPsk($NW.Name,"Manual",$A,$E,$PW)
        $This.NewWifiProfile($PX,$This.Selected.Name,$True)

        $Param = $This.GetWiFiConnectionParameter($NW.Name)
        $CH    = $This.NewWiFiHandle()
        $This.WlanConnect($CH,[Ref]$This.Selected.Guid,[Ref]$Param,[IntPtr]::Zero)
        $This.RemoveWifiHandle($CH)

        Start-Sleep 3
        $Link  = $This.Selected.Description
        $This.Unselect()
```

```
            $This.Select($Link)

            $This.Update()
            If ($This.Connected)
            {
                $Splat                                = @{

                    Type    = "Image"
                    Mode    = 2
                    Image   = $This.OEMLogo
                    Message = "Connected: $($NW.Name)"
                }

                Show-ToastNotification @Splat
            }
            If (!$This.Connected)
            {
                $This.RemoveWifiProfile($NW.Name)

                $Splat                                = @{

                    Type    = "Image"
                    Mode    = 2
                    Image   = $This.OEMLogo
                    Message = "Unsuccessful: Passphrase failure"
                }
                Show-ToastNotification @Splat
            }
        }
        Update()
        {
            "Determine/Set connection state" | Write-Comment -I 12 | Set-Clipboard
            Switch -Regex ($This.Selected.Status)
            {
                Up
                {
                    $This.Connected = $This.NetshShowInterface($This.Selected.Name)
                }
                Default
                {
                    $This.Connected = $Null
                }
            }
        }
        Wireless()
        {
            # // ____________________________
            # // | Load the module location |
            # // ____________________________

            $This.Module  = Get-FEModule
            $This.OEMLogo = $This.Module.Graphics | ? Name -eq OEMLogo.bmp | % Fullname

            # // ___________________________
            # // | Load the runtime types |
            # // ___________________________

            ForEach ($X in "","AccessStatus","State")
            {
                $Item = "[Windows.Devices.Radios.Radio$X, Windows.System.Devices, ContentType=WindowsRuntime]"
                "$Item > `$Null" | Invoke-Expression
            }

            # // ______________________________________
            # // | Get access to any wireless adapters |
            # // ______________________________________

            $This.Adapters = $This.RefreshAdapterList()

            # // ________________________________________
            # // | Throw if no existing wireless adapters |
            # // ________________________________________
```

```
        If ($This.Adapters.Count -eq 0)
        {
            Throw "No existing wireless adapters on this system"
        }

        # // ____________________________
        # // | Requesting Radio Access |
        # // ----------------------------

        $This.Request = $This.RadioRequestAccess()
        $This.Request.Wait(-1) > $Null

        # // ______________________________________
        # // | Throw if unable to ascertain access |
        # // --------------------------------------

        If ($This.Request.Result -ne "Allowed")
        {
            Throw "Unable to request radio access"
        }

        # // ___________________________________
        # // | Establish radio synchronization |
        # // -----------------------------------

        $This.Radios = $This.RadioSynchronization()
        $This.Radios.Wait(-1) > $Null

        # // ____________________________________________
        # // | Throw if unable to synchronize radios |
        # // --------------------------------------------

        If (!($This.Radios.Result | ? Kind -eq WiFi))
        {
            Throw "Unable to synchronize wireless radio(s)"
        }

        $This.Refresh()
    }
    [Object[]] RefreshAdapterList()
    {
        Return Get-NetAdapter | ? PhysicalMediaType -match "(Native 802.11|Wireless (W|L)AN)"
    }
    Scan()
    {
        $This.List   = @( )
        $This.Output = @( )

        [Windows.Devices.WiFi.WiFiAdapter, Windows.System.Devices, ContentType=WindowsRuntime] > $Null
        $This.List   = $This.RadioFindAllAdaptersAsync()
        $This.List.Wait(-1) > $Null
        $This.List.Result

        $This.List.Result.NetworkReport.AvailableNetworks | % {

            $This.Output += [Ssid]::New($This.Output.Count,$_)
        }

        $This.Output = $This.Output | Sort-Object Strength -Descending
        Switch ($This.Output.Count)
        {
            {$_ -gt 1}
            {
                ForEach ($X in 0..($This.Output.Count-1))
                {
                    $This.Output[$X].Index = $X
                }
            }
            {$_ -eq 1}
            {
                $This.Output[0].Index = 0
```

```
            }
            {$_ -eq 0}
            {
                Throw "No networks detected"
            }
        }
    }
    Refresh()
    {
        Start-Sleep -Milliseconds 150
        $This.Scan()

        Write-Progress -Activity Scanning -Status Starting -PercentComplete 0

        $C = 0
        $This.Output | % {

            $Status  = "($C/$($This.Output.Count-1)"
            $Percent =  ([long]($C * 100 / $This.Output.Count))

            Write-Progress -Activity Scanning -Status $Status -PercentComplete $Percent

            $C ++
        }

        Write-Progress -Activity Scanning -Status Complete -Completed
        Start-Sleep -Milliseconds 50
    }
}
```

```
# // ------------------------------------------------------------------------------
# // | Alright, so now it is time to capture the class a process it into an object. |
# // | Here we go.                                                                  |
# // | Go ahead and capture the [Wireless] class to a variable named $Wifi          |
# // ------------------------------------------------------------------------------

$WiFi = [Wireless]::New()
```

```
PS Prompt:\> $WiFi = [Wireless]::New()
```

```
# // ------------------------------------------------------------------------------------------
# // | If the device you're using has wireless radios in it...? You probably just saw the      |
# // | Write-Progress function calculate all of the available wireless networks nearby.        |
# // | Don't take my word for it... let's test the output.                                     |
# // | What do we get back from this object if we start playing around with it in the console? |
# // ------------------------------------------------------------------------------------------
```

```
PS Prompt:\> $WiFi

Adapters  : {MSFT_NetAdapter (CreationClassName = "MSFT_NetAdapter", DeviceID = "{E3A47A46-9920-469E-....}
Request   : System.Threading.Tasks.Task`1[Windows.Devices.Radios.RadioAccessStatus]
Radios    : System.Threading.Tasks.Task`1[System.Collections.Generic.IReadOnlyList`1[Windows.Devices...
List      : System.Threading.Tasks.Task`1[System.Collections.Generic.IReadOnlyList`1[Windows.Devices...
Output    : {, , , ...}
Selected  :
Connected :
```

```
# // ------------------------------------------------------------------------------------------
# // | Well, that's a fair amount of information right there at our fingertips.               |
# // | To be blunt, the information we want to see is in the property, Output                 |
# // | If we access that property, we will see a list that has items that look like this... |
# // ------------------------------------------------------------------------------------------
```

```
PS Prompt:\> $WiFi.Output

Index            : 0
Name             :
Bssid            : 8A:15:04:A2:44:F4
Type             : 802.11n
Uptime           : 11h 22m 41s 027
NetworkType      : Infrastructure
Authentication   : Rsna
Encryption       : Ccmp
Strength         : 3
BeaconInterval   : 00:00:00.1024000
ChannelFrequency : 2437000
IsWifiDirect     : False
```

```
# // ______________________________________________________________________
# // | Now, there are many other entries in this particular list, but... |
# // | The LIST format doesn't show them all the best way.                |
# // | So let's try to use $WiFi.Output | Format-Table                    |
# // ----------------------------------------------------------------------
```

```
PS Prompt:\> $Wifi.Output | Format-Table

Index Name                 Bssid             Type     Uptime               Netw. Auth.     Enc. Str. Beacon Int
----- ----                 -----             ----     ------               ----- --------- ---- ---- ----------
    0                      8A:15:04:A2:44:F4 802.11n  11h 22m 41s 027      Infr. Rsna      Ccmp    3 00:00:00~
    1                      8A:15:04:A2:44:F7 802.11n  11h 22m 41s 028      Infr. Rsna      Ccmp    3 00:00:00~
    2                      8A:15:04:A2:44:F3 802.11n  11h 22m 41s 027      Infr. Rsna      Ccmp    3 00:00:00~
    3                      8A:15:04:A2:44:FF 802.11n  11h 22m 41s 024      Infr. Open80211 Wep     3 00:00:00~
    4 Market 32            8A:15:04:A2:44:F0 802.11n  11h 22m 40s 981      Infr. Open80211 None    3 00:00:00~
    5                      8A:15:04:A2:E3:44 802.11n  19h 38m 30s 275      Infr. Rsna      Ccmp    3 00:00:00~
    6                      8A:15:04:A2:96:5F 802.11n  2d 13h 57m 15s 904   Infr. Open80211 Wep     3 00:00:00~
    7                      8A:15:04:A2:F0:27 802.11n  4d 08h 11m 17s 380   Infr. Rsna      Ccmp    3 00:00:00~
    8                      8A:15:04:A2:CC:37 802.11n  1d 09h 05m 17s 111   Infr. Rsna      Ccmp    3 00:00:00~
    9                      8A:15:04:A2:44:F6 802.11n  11h 22m 41s 027      Infr. Rsna      Ccmp    3 00:00:00~
   10 Market 32            8A:15:04:A3:A8:10 802.11n  1d 07h 30m 06s 865   Infr. Open80211 None    3 00:00:00~
   11 HP-Print-71-Off...   80:CE:62:93:8B:71 Unknown  10d 04h 27m 11s 766  Infr. RsnaPsk   Ccmp    3 00:00:00~
   12                      8A:15:04:A3:B7:13 802.11n  2d 07h 17m 33s 111   Infr. Rsna      Ccmp    3 00:00:00~
   13                      8A:15:04:A3:B7:14 802.11n  2d 07h 17m 33s 111   Infr. Rsna      Ccmp    3 00:00:00~
   14                      8A:15:04:A3:B7:1F 802.11n  2d 07h 17m 33s 108   Infr. Open80211 Wep     3 00:00:00~
   15 Market 32            8A:15:04:A3:B7:10 802.11n  2d 07h 17m 33s 029   Infr. Open80211 None    3 00:00:00~
   16                      8A:15:04:A3:B7:11 802.11n  2d 07h 17m 33s 110   Infr. Rsna      Ccmp    3 00:00:00~
   17                      8A:15:04:A2:E3:4F 802.11n  19h 38m 30s 886      Infr. Open80211 Wep     3 00:00:00~
   18                      8A:15:04:A2:E3:46 802.11n  19h 38m 30s 275      Infr. Rsna      Ccmp    3 00:00:00~
   19                      8A:15:04:A2:44:F1 802.11n  11h 22m 41s 026      Infr. Rsna      Ccmp    3 00:00:00~
   20                      8A:15:04:A3:B7:16 802.11n  2d 07h 17m 33s 111   Infr. Rsna      Ccmp    3 00:00:00~
   21                      8A:15:04:A3:B7:17 802.11n  2d 07h 17m 33s 112   Infr. Rsna      Ccmp    3 00:00:00~
   22 Marks Car            00:6F:F2:31:0E:66 802.11n  00h 29m 23s 581      Infr. RsnaPsk   Ccmp    2 00:00:00~
   23 JOY                  E4:71:85:17:7C:80 802.11n  37d 05h 07m 17s 427  Infr. RsnaPsk   Ccmp    2 00:00:00~
   24 MarksAutomotive...   E6:F4:C6:08:DA:43 802.11n  6d 23h 33m 16s 214   Infr. RsnaPsk   Ccmp    2 00:00:00~
   25                      00:30:44:39:FC:01 802.11n  15h 29m 38s 201      Infr. RsnaPsk   Ccmp    2 00:00:00~
   26                      8A:15:04:A2:E3:40 802.11n  19h 38m 30s 274      Infr. Open80211 None    2 00:00:00~
   27 TheShop              C0:56:27:3D:6D:F4 802.11ac 37d 05h 07m 35s 966  Infr. RsnaPsk   Ccmp    2 00:00:00~
   28 DIRECT-epson-se...   E2:BB:9E:56:AC:F4 802.11n  1d 07h 08m 19s 032   Infr. RsnaPsk   Ccmp    2 00:00:00~
```

There's a lot of stuff that I haven't covered nor talked about in this lesson plan. I'm going to go on a couple of relevant tangents, to discuss a concept called PSYCHOLOGICAL MANIPULATION. Because, the REASON why I've WRITTEN this lesson plan, is this simple. In our society, American society, there are a lot of people that just casually trust one another, to the point where nearly every living person within our society develops what's called a “CALLOUS” to when people lie to one another, and it causes problems to develop ELSEWHERE.

It's like this, FOX NEWS/HANNITY/CARLSON say... “Climate change ain't real...” but then Hurricane Ian, 250mi F4 tornado tracks, droughts like Lake Mead being at a historic 1/3 of its normal capacity, wildfires that are able to burn millions of acres of (forest/homes) to the ground, massive flooding... Climate change is DEFINITELY REAL.

Commentary /----------\
/------------

I'm going to sidestep away from the comments I just made about CLIMATE CHANGE, as I talk about that a LOT in:

| Top Deck Awareness - Not News \| Used to be news...? Now it's Not News. Not News. Part of the Not News Network |
| https://github.com/mcc85s/FightingEntropy/blob/main/Docs/2022_1008_TDA_Not_News.pdf |

I want to focus on this particular application.

```
# // ------------------------------------------------------------------------------
# // |----------------------------------------------------------------------------|
# // | Sorta looks like I know what I'm doing, doesn't it...?                     |
# // | There's plenty more that can be done with this particular radio class.     |
# // |                                                                            |
# // | The point of this, is to exhibit the following statement:                  |
# // | I'm an actual expert who knows what he is doing/saying.                    |
# // |     ----------------     ----------------------------                      |
# // | Sometimes I even know how to EXPOSE people who either KNOW they're lying...|
# // | ...or DON'T KNOW they're lying. That's the quality that causes me to be... |
# // | ...a CUT ABOVE THE REST.                                                   |
# // |                                                                            |
# // | Before I came along...? You had people giving each other high fives, a real|
# // | comfortable pat on the back, or a thumbs up. George W. Bush inspired Facebook|
# // | and Google to adopt the "thumbs up" button, as a direct result of this:    |
# // |                                                                            |
# // | [Mission Accomplished]                                                     |
# // | 05/01/03 | https://drive.google.com/file/d/1EDBRJHxcPKOkbJ75hfPik7rxD2zERtt3 |
# // |                                                                            |
# // | I realize that some people will look at that picture or read my commentary,|
# // | and they'll just casually pretend as if they don't see how much sense I make,|
# // | in THAT PICTURE, as well as the rest of the lesson plan, and the videos that|
# // | some guys talk about in the following dialog...                            |
# // |____________________________________________________________________________|
# // |----------------------------------------------------------------------------|
# // | Guy[1]: Maybe if we just pretend like we can't HEAR him, he'll go away.    |
# // | Guy[2]: That is a splendid plan, Guy[1].                                   |
# // | Guy[1]: Heh heh heh...                                                     |
# // |         Indubitubly, my good friend...                                     |
# // |         Indubitubly indeed.                                                |
# // | Guy[2]: Yeah, well...                                                      |
# // |         This dude would have to have like THOUSANDS of videos to catch US. |
# // |         https://youtu.be/LfZW-s0BMow                                       |
# // | Guy[1]: Heh heh heh...                                                     |
# // |         You basically read my mind there, Guy[2].                          |
# // | Guy[3]: What the hell are you guys talking about...?                       |
# // | Guy[2]: Heh, this MICHAEL C. COOK SR. from CLIFTON PARK, NY...             |
# // |         Thinks he's hot s***.                                              |
# // | Guy[1]: Yeh, LOL.                                                          |
# // |         Dude doesn't even have his CISCO CERTIFICATIONS...                 |
# // | Guy[3]: What exactly is this video, anyway...?                             |
# // |         https://youtu.be/LfZW-s0BMow                                       |
# // | Guy[1]: Oh, it's this MICHAEL C. COOK SR. guy from CLIFTON PARK, NY...     |
# // |         Trying to go around teaching people how to reset a cable modem.    |
# // |         And like, managing the CLIFTON PARK COMPUTER ANSWERS shop.         |
# // | Guy[3]: He looks like he's quite the professional...                       |
# // |         He knows how to use IPCONFIG.                                      |
# // | Guy[1]: Yeah, but look at how messy the shop is in the background...       |
# // |         THAT AUTOMATICALLY MEANS, what he's sayin' is stupid.              |
# // |         And, so is HE.                                                     |
# // | Guy[2]: Yyyyyyyyyyup.                                                      |
# // |         Dude needs to go back in time to when he recorded this video...?   |
# // |         And then just clean the shop and make the video AGAIN.             |
# // |         THEN he would have our respect...                                  |
# // |         So...                                                              |
# // | Guy[3]: ...are you serious...?                                             |
# // | Guy[1]: Yep.                                                               |
# // | Guy[2]: Absolutely.                                                        |
# // |         Can't go around calling yourself an expert if stuff isn't organized.|
# // | Guy[3]: What about this video...?                                          |
# // |         https://youtu.be/0nEiGijjOEY                                       |
```

```
# // | Guy[1]: WOW.                                                                   |
# // |         Dude goes around wearing a vest and stuff...?                          |
# // |         That's pretty dumb.                                                    |
# // | Guy[2]: It is pretty dumb, isn't it...?                                        |
# // | Guy[3]: Did your like, mom and dad teach you that wearing a vest and managing |
# // |         network equipment is dumb...?                                          |
# // | Guy[1]: Uh, no.                                                                |
# // | Guy[2]: ...                                                                    |
# // | Guy[3]: Guy[2]...?                                                             |
# // | Guy[2]: Yeah man, my mom and dad DID teach me that stuff is dumb.              |
# // | Guy[3]: Wow.                                                                   |
# // |         You know...                                                            |
# // |         *wearing a vest* I'm wearin' a black vest that looks like-             |
# // | Guy[1]: ...the one he's wearing in that video...                               |
# // | Guy[2]: ...                                                                    |
# // | Guy[3]: So, you think I've ALWAYS looked pretty dumb in this vest that         |
# // |         I wear all the time, huh...?                                           |
# // | Guy[2]: ...                                                                    |
# // | Guy[3]: Wow, dude.                                                             |
# // |         *shaking head, walking away* Unbelievable...                           |
# // |________________________________________________________________________________|
# // |--------------------------------------------------------------------------------|
# // | Look, that's a fictional skit that I just made up on the fly.                  |
# // | I'm not sure if that's READILY APPARENT with some of my rhetoric...?           |
# // | But, I gotta TRY and make the MATERIAL interesting to read.                    |
# // |                                                                                |
# // | Believe it or not, but there ARE some experts in the field, that do this too. |
# // | For instance, Kevlin Henney, Robert Sopolsky, Jeremy Rifkin, Jordan Peterson, |
# // | they're not ALL professors, however, they all have INTERESTING CONJECTURE, or |
# // | RHETORIC. Perhaps they may not write up fictional stories like I do, but that |
# // | is just a technique that I like to use, as it is part of my STYLE.             |
# // |________________________________________________________________________________|
# // ----------------------------------------------------------------------------------
```

```
                                                                                                   ____________/
\_________________________________________________________________________________________________/ Commentary
  Examination /-------------------------------------------------------------------------------------------------\
/-------------
```

```
# // ____________________________________________________________________________________
# // | I just mentioned a bunch of really cool dudes who happen to be experts at:       |
# // | knowing what the hell they're (saying/doing).                                    |
# // |                                                                                  |
# // | Kevlin Henney has a series of videos that I've seen once upon a time...          |
# // | one of the most notable is this one in particular, about the ol' FizzBuzz game. |
# // | https://youtu.be/LueeMTTDePg?t=645                                               |
# // |                                                                                  |
# // | Here is the code in that specific video at that specific time...                 |
# // |                                                                                  |
# // | # // C Sharp | Verbatim                                                          |
# // | for (var i = 1; 1 <= 100; i++) {                                                 |
# // |     // For each iteration,                                                       |
# // |     // initialize an empty string                                                |
# // |     var string = '';                                                             |
# // |                                                                                  |
# // |     // If 'i' is divisible through 3                                             |
# // |     // without a rest, append 'Fizz'                                             |
# // |     if (i % 3 == 0) {                                                            |
# // |         string += 'Fizz';                                                        |
# // |     }                                                                            |
# // |                                                                                  |
# // |     // If 'i' is divisible through 5                                             |
# // |     // without a rest, append 'Buzz'                                             |
# // |     if (i % 5 == 0) {                                                            |
# // |         string += 'Buzz';                                                        |
# // |     }                                                                            |
# // |                                                                                  |
# // |     // If 'string is still empty,                                                |
# // |     // 'i' is not divisible by 3 or 5,                                           |
# // |     // so use the number instead                                                 |
```

```
# // |     if (string == '') {                                                  |
# // |         string += i;                                                     |
# // |     }                                                                    |
# // |                                                                          |
# // |     // At the end of this iteration, print the string                    |
# // |     console.log(string);                                                 |
# // | }                                                                        |
# // |                                                                          |
# // | Before I continue with the lesson plan, I'm going to CONVERT all of      |
# // | that C Sharp into PowerShell, since it's basically the same thing.       |
# // ---------------------------------------------------------------------------
```

_____/ Examination
FizzBuzz /_____\
/----------

```
# // ------------------------------------------------------------------------------------------
# // | Now I'm about to take things to the next level by making this thing far more complicated,   |
# // | but also...? More manageable, and able to be interacted with, as well as providing a        |
# // | Write-Progress indicator, so that people can throw whatever number they want at this thing. |
# // ------------------------------------------------------------------------------------------
```

```
# // ------------------------------------------------------------------------------------------------------
# // | This right here, is your regular, ordinary, every-day-at-the-park, standard-issue FizzBuzz object. |
# // | Basically, the console output is saved here, and each object will calculate whether it is:         |
# // | 1) a Fizz object,                                                                                  |
# // | 2) a Buzz object,                                                                                  |
# // | 3) a Fizz Buzz object,                                                                             |
# // | 4) or a number if it's none of those objects/items...                                              |
# // ------------------------------------------------------------------------------------------------------

Class FizzBuzzObject
{
    [UInt32] $Index
    [String] $String
    FizzBuzzObject([UInt32]$Index)
    {
        $This.Index   = $Index
        $This.String += @( Switch ($Index)
        {
            {$_ % 3 -eq 0} { "Fizz" } {$_ % 5 -eq 0} { "Buzz" } Default { $Index }
        })
    }
    [String] ToString()
    {
        Return $This.String
    }
}
```

```
# // ------------------------------------------------------------------------------------------------------
# // | This is basically a stacking of the chips, a container for each individual FizzBuzzObject.        |
# // | With it, you can totally mess up some bad guy's days... without putting a lot of thought into it. |
# // |                                                                                                    |
# // | In all seriousness, this keeps track of total, depth, and the output.                              |
# // | You can ALSO insert a number that is HIGHER or LOWER than 100 (but no less than 1)                 |
# // ------------------------------------------------------------------------------------------------------

Class FizzBuzzContainer
{
    [UInt32] $Total
    [UInt32] $Depth
    [Object] $Output
    FizzBuzzContainer([UInt32]$Total)
    {
        If ($Total -le 1)
```

```
            {
                Throw "Must provide a total higher than 1"
            }

            $This.Total    = $Total
            $This.Depth    = ([String]$Total).Length
            $Stage         = [Math]::Round($Total/20)
            $Slot          = 0..($Total) | ? { $_ % $Stage -eq 0 }
            $Slot[-1]      = $Total
            $Hash          = @{ }

            Write-Progress -Activity "Calc. [FizzBuzz]" -Status $This.Status(0) -PercentComplete 0
            ForEach ($X in 1..$Total)
            {
                $Hash.Add($Hash.Count,[FizzBuzzObject]::New($X))
                If ($X -in $Slot)
                {
                    Write-Progress -Activity "Calc." -Status $This.Status($X) -PercentComplete $This.Percent($X)
                }
            }
            Write-Progress -Activity "Calc. [FizzBuzz]" -Status $This.Status($This.Total) -Complete

            $This.Output = $Hash[0..($Hash.Count-1)]
        }
        [String] Status([UInt32]$Rank)
        {
            Return "({0:d$($This.Depth)}/$($This.Total))" -f $Rank
        }
        [Double] Percent([UInt32]$Rank)
        {
            Return ($Rank*100)/$This.Total
        }
        [Object[]] Factor([UInt32]$Mode)
        {
            If ($Mode -notin 0,1)
            {
                Throw "Invalid mode"
            }

            Return @( Switch ($Mode)
            {
                0 { $This.Output | ? String -notmatch \d+ } 1 { $This.Output | ? String -match \d+ }

            })
        }
    }
```

```
    # // ------------------------------------------------------------------------------------
    # // | Lets instantiate this PowerShell representation of Kevlin Henney's C# code.      |
    # // | Might not have been HIS, but it's whatever.                                      |
    # // |                                                                                  |
    # // | I'm pretty sure that the original author won't care if I EXAMINE it.             |
    # // | If they do...? Well, feel free to get a hold of me, and I will take this educational |
    # // | lesson on the sacred Fizz Buzz object, offline.                                  |
    # // ------------------------------------------------------------------------------------

    $List = [FizzBuzzContainer]::New(10000)
```

```
    PS Prompt:\> $List = [FizzBuzzContainer]::New(10000)
    PS Prompt:\> $List

    Total Depth Output
    ----- ----- ------
    10000     5 {1, 2, Fizz, 4...}

    PS Prompt:\>
```

Alright, what happens if we were to examine the OUTPUT class...?

Since there's 10000 entries based on the sample input, I'm going to LIMIT the actual results to about 30.

To select 30 elements from the Output property, we can do this a few ways.
My FAVORITE way to do just that, is to simply use a LEFT_SQ_BRACKET NUMBER DOT DOT HIGHER_NUMBER RIGHT_SQ_BRACKET.
To ILLUSTRATE that as characters... [0..29] ← That is an array selector. If there are indeed, at least 30 items, then this will select those items.

Note: If the numbers are OUTSIDE of the BOUNDS of the ARRAY...? The console will throw an error.
And that's “No bueno, Broseph McGoseph.”

```
/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\

Broseph McGoseph : Uh... it isn't...?
Me               : Nah, it's no bueno.
Broseph McGoseph : How'd you know I was reading your like, lesson plan...?
Me               : I didn't.
Broseph McGoseph : WELL, my name just so happens to be Broseph McGoseph.
Me               : Cool, I was just coming up with a name out of thin air, dudeface.
Dudeface         : What the heck...?
                   How'd you know I was reading your lesson plan TOO, dude...?
Me               : I didn't.
Dudeface         : WELL... just like *points at Broseph McGoseph* that dude right there...?
Me               : …?
Dudeface         : ...my name just so happens to be Dudeface...
Me               : So, I'm just pulling random people's names out of thin air, and accurately naming people...?
Broseph McGoseph : *crosses arms* Yeah, dude.
Dudeface         : *also crosses arms* What gives, pal...?
Me               : *hands up* Alright~!
                   Jeez~!
                   Sorry if I offended either one of you by randomly naming you...
Broseph McGoseph : You better be...
Dudeface         : Yeah man.
                   *shaking head* You got a lot of nerve to just say our names, randomly like that.
Me               : I just apologized for doin' that.
Dudeface         : Well, I'm satisfied by your apology...
                   *looks at Broseph McGoseph* What about *points at Broseph McGoseph* YOU, dude...?
Broseph McGoseph : I, am ALSO satisfied by this dude's apology...
Me               : Alright, can I continue with the lesson plan...?
Dudeface         : Yeh, that's fine.
Broseph McGoseph : *uncrosses arms, pauses slightly* Yeah man, I guess...
                   I have such a RARE NAME, that it seems more than just COINCIDENTAL...
                   ...that you'd be obtuse enough to CASUALLY throw my name into the mix on ACCIDENT...
Me               : I mean, it was accidental, buddy.
                   Sorry.
Broseph McGoseph : Fine.
                   *waves hands around* Go ahead and proceed with your lesson plan that I'm learning from...
Me               : Alright then.

\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/
```

```
PS Prompt:\> $List.Output[0..29]

Index String
----- ------
    1 1
    2 2
    3 Fizz
    4 4
    5 Buzz
    6 Fizz
    7 7
    8 8
    9 Fizz
   10 Buzz
   11 11
   12 Fizz
   13 13
   14 14
   15 Fizz Buzz
   16 16
   17 17
```

```
    18 Fizz
    19 19
    20 Buzz
    21 Fizz
    22 22
    23 23
    24 Fizz
    25 Buzz
    26 26
    27 Fizz
    28 28
    29 29
    30 Fizz Buzz

PS Prompt:\>
```

Now, with this particular output...? You can get a sense of how the remaining (10000 - 30) items will appear.
Now, the REASON why someone would want to do something like this, is to essentially capture the way the output appears, and can be reproduced.

With Kevlin Henney's example, it is literally just outputting stuff to the console object directly.
With the example I've just provided, I can select entries in the array and then access the OBJECT rather than the string output. This is a FEATURE that makes PowerShell SO POWERFUL.

```
                                                                                                              __________/
\____________________________________________________________________________________________________________/ FizzBuzz
  History /‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾\
/‾‾‾‾‾‾‾‾‾
```

PowerShell is INCREDIBLY FLEXIBLE.
It can encompass aspects of both FUNCTIONAL languages like HASKELL, or OBJECT-ORIENTED languages like C#, C++, Java, JavaScript, Python, etc. It can also encompass aspects of STRING-BASED languages such as tsch/T C Shell, Bash, or whatever on Linux/Unix/FreeBSD.

Yeah, I have a suggestion for the PowerShell team. At some point I saw the PowerShell development team say...

```
---------------------------------------------------------------------
| Q: Hey, community... Should we like, make a build for FreeBSD...? |
| A: Yeh. You totally should.                                       |
---------------------------------------------------------------------

/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\
```

MS Guy[1]: Alright, so we've got like AT LEAST (1) dude who thinks that would be pretty cool, to implement
           PowerShell on FreeBSD.
MS Guy[2]: *puts sunglasses on* Well, alright then...
           Looks like we've got some work to do.

```
\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/
```

Here's a video of me doing some real cool stuff with Remote Desktop, Hyper-V, PowerShell, and FreeBSD.

```
--------------------------------------------------------------------------------
| 06/27/21 | Advanced System Administration Lab | https://youtu.be/xgffIccX1eg |
--------------------------------------------------------------------------------
```

What is featured in THAT particular video, was eventually repurposed and streamlined into a demonstration with the GRAPHICAL USER INTERFACE, in this video...

```
------------------------------------------------------------------------------------
| 12/04/21 | [FightingEntropy(π)][FEInfrastructure] | https://youtu.be/6yQr06_rA4I |
------------------------------------------------------------------------------------
```

The point of all this, is to showcase examples as to (HOW/WHY), I might be worth making an INVESTMENT into.
The lesson plan is ALSO meant to showcase that I know how to DISSECT various components of NETWORKING, PSYCHOLOGY, INSTRUCTIONAL DESIGN, VIRTUALIZATION, and APPLICATION DEVELOPMENT.

I ALSO know many various components of GRAPHICAL USER INTERFACE design, and GRAPHIC DESIGN.

```
--------------------------------------------------------------------------------------------------------------
|----------|---------------------------------------------|---------------------------------------------------|
| 07/21/01 | <|3FG20K>'s Shopping Maul : 20KDM1 Complete!                                                    |
|            https://web.archive.org/web/20010801215620/http://www.planetquake.com/bfg20k/news.html          |
| 01/05/02 | [Lvlworld Review: 20KDM1/Tempered Graveyard] | https://lvlworld.com/review/Tempered%20Graveyard |
| 06/07/21 | [Quake III Arena: 20KDM1/Tempered Graveyard] | https://youtu.be/dyHwm9AdkQs                     |
| 03/09/21 | A Deep Dive Demonstration: XAML/PowerShell   | https://youtu.be/NK4NuQrraCI                     |
|__________|_____________________________________________|___________________________________________________|
--------------------------------------------------------------------------------------------------------------
```

Some people might say “Hey, there's a TYPO on the DATES of those first (2) entries.”

Nah, those aren't typos. Those are actual, real-life, examples, of ME... at some point 21 years ago... being really skilled at GRAPHIC DESIGN... from like 2001. I made many other maps for Quake III Arena when I was 15+.

In the LVL review of that specific map, 20KDM1/Tempered Graveyard, it got tanked a lot for the LIGHTING.
I was able to fix the lighting and then repurpose the map, and include it in the Rocket Arena 3 map that I made for a gaming clan I was in, many, many years ago, called HELL.

The name of that RA3 map was called "Dude, you can go to HELL"
I try to use a lot of cheesy puns, so calling the map "Dude, you can go to HELL"...?
It's just the type of thing where people are gonna look at each other with a question mark on their face...

/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯\

```
Guy[1]: *looks at Guy[2]* Hey...
Guy[2]: *eyebrows up* Sup...?
Guy[1]: So, like...
        ...this dude made a map for Rocket Arena 3 like, (21) years ago called 'Dude, you can go to HELL'.
Guy[2]: Wow... that's really INSULTING...
Guy[1]: *eyebrows inward* Now, who in their right mind, would just go around naming a level something like that...?
Guy[2]: *eyebrows inward* I dunno, dude.
        If this dude thinks we're gonna play this map...?
        Well, buddy...
        NOW we're NOT gonna play it, because the name of the map is rather INSULTING...
Guy[1]: Yeah, dude.
        You said it.
        *crosses arms* I'm not even gonna look at the review for it either.
Guy[2]: Yeah dude, the map probably sucks, because of the name.
Guy[1]: That's pretty lame, for the map to have that name.
```

__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/

Look, I'm pretty sure that people aren't going around having conversations like that.
However, if I INCLUDE something that shows how INSIGHTFUL I am, like... I have insight into how ridiculous any particular IDEA or STORYBOARD sounds...?

Then, I have to emphasize and capitalize on the notion, that maybe my ATTENTION TO DETAIL, like, (21) years ago, might allow some people to be like "Well, alright dude, you made those maps when you were 15 years old 21 years ago...?"

Yeah. I'm not going around saying that I deserve a Nobel peace prize for making those maps.
Nor a Pulitzer prize like this guy...

11/24/95	The Road Ahead	https://en.wikipedia.org/wiki/The_Road_Ahead_%28Gates_book%29

Dude knew what the hell he was (saying/doing) way back then. It's like the hallmark of real smart dudes... knowing what the hell you were (saying/doing) at some point in the distant past... and having been RECORDED, (saying/doing) something rather INSIGHTFUL and INTELLIGENT.

Cause, I gotta tell ya... I'm not sure if the reader has ever heard of this guy "WILLIAM GATES"...?
But- the man has literally showed the world how real he's kept it, all of these years.

Now he spends his days researching things that are seen in videos like this...

04/03/15	Bill Gates: The next outbreak...? We're not ready	https://youtu.be/6Af6b_wyiwI

And sure as s***...? He was correct about that (when is he not...?) because then this stuff started happening...

11/17/19	First Documented case of COVID-19	https://youtu.be/5DGwOJXSxqg

Uh-oh. People heard Bill Gates go on stage and talk about the next epidemic, saying "We're not ready"...?
But then, they were like... "Look ya friggen smart guy. We heard ya, so... simmer down, sparky."
The man wasn't even showing any signs of frustration or anything like that either, he just CASUALLY stated that the next pandemic would be pretty bad, and that the world wasn't ready for it.

Then, people all unwittingly stepped into Dr. Emmett Brown's flying Delorean time machine with Marty McFly...
...and then the world found out the hard way, 4.5 years later, that listening to smart dudes who know what the hell they're (saying/doing), is good for your health and stuff. Ya know...? Bill Gates didn't go around saying:

Gates : You better listen to what I say if you know what's good for ya~!

Nah. People just AUTOMATICALLY assumed "Alright... dude says some stuff about pandemics... maybe we should pay attention..." But, when people SAY STUFF like "OooOOhhHH, let's avoid the next pandemic."...?

Don't JUST give the guy a standing ovation.
Don't JUST clap in unison at how smart this dude is, and then walk out of the auditorium saying:

/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯\

Guy[1]: Man, that Bill Gates guy...?
        He sure knows how to dazzle the audience...
Guy[2]: Yeah man, he's a real charmer for sure.
Guy[1]: Just the WAY that the man spoke about avoiding the next pandemic...?
Guy[2]: ...It was such a delightful thing to experience, wasn't it...?
Guy[1]: Yeah, dude.
        I basically feel safer, already.
Guy[2]: I mean, we heard the man tell us how it really is.
        That's basically good enough to practically avoid the next epidemic, already.
Guy[1]: *shaking head* You said it, pal.
        That's exactly what I was thinking.

__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/

I'm not saying that people shouldn't clap for the man nor talk highly about him at all.
There's nothing wrong with being blown away by how smart this dude is.

He's got a tendency to be so on top of things...?
He can PREDICT if some dude is gonna come out of the woodwork, trying to SABOTAGE something, somewhere...
(Side point: I talk about his ability to PREDICT potential PROBLEMS, in a skit named "Always Watchin'")

The problem is, people will hear what this man says, and FAIL TO TAKE ACTION.
That is the part that our society has trouble with... FAILING TO TAKE ACTION.

Because I gotta tell ya...
People will actually argue with me about what I just said, and say:

/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯\

Guy[1]: We took action, dude...
        We actually CLAPPED for the man AND gave him a STANDING OVATION, at that TED talk in 2015.
        So...
        Don't go around saying we FAILED TO TAKE ACTION.
Guy[2]: Yeah, dude.
        I literally recorded it on video, we all STOOD...?
        And literally CLAPPED for the man and his bright minded rhetoric.
        Shame on you for saying we FAILED TO TAKE ACTION...

__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/

Yeah, STANDING OVATIONS and CLAPPING is like, not the ONLY ACTION people should've chosen to take.
Nah, OPERATION WARP SPEED...? That's like, the ACTION that people COULD'VE taken... years beforehand.

That's what the man was basically telling people to do.

/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯\

Gates   : Build something that can advance humanity's ability to handle the next pandemic.
            Because, I gotta say...?
            shaking head, slight frown The world just isn't ready for the next pandemic...
Everybody : Oh wow.
            stands up, starts clapping for 20 minutes
            I'm literally blown away, right now.

__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/

It was fine for people to give him a standing ovation. The problem is, what he MEANT, in that presentation, was that it SHOULD BE A PRIORITY, to: SET UP THE FOUNDATION for PROGRAMS LIKE OPERATION WARP SPEED. Ya know...?

That's what he was sayin'. That's why the dude will literally spend his days, going to dinner parties, talking about EPIDEMIC this, PANDEMIC that, and thinking to himself "Man, I must be a real bore at these friggen dinner parties, huh...?" And I ALSO realize that people must think I'm mad boring, going around trying to teach people about CLASSES this, PROGRAMS/PROGRAMS that, DESIGN and PSYCHOLOGICAL constructs are in need of advancement.

Yeah. That's the notion that I'm alluding to, is that the man probably realizes that even with his great success and PHILANTHROPIC gestures...? And, as smart as he is...? He probably gets pretty frustrated by how often people talk highly about him, but then they don't actually take action on what he goes around saying to people.

| 05/08/22 | Bill Gates/CNN, $1B/Y plan to avoid next pandemic | https://youtu.be/F_heKZUKnCU |

This entire document, is ME, choosing to TAKE ACTION, on some of the COOL STUFF that the man just went right ahead, and built... with his best friend Paul Allen. Starting out with a little Altair 8800, writing a version of BASIC for it, and then just crushing an entire industry of people that worked their entire lives, not doing that. I'm not going to talk about how the Gates+Allen combo-pack (company/legacy) remains alive and well to this very day, because that was just (1) chapter of this man Bill Gates' legacy.

He does a lot of work with his FOUNDATION, “The Bill and Melinda Gates Foundation”, which basically takes things to the next level with providing humanitarian aid all around the globe.

Correlations / History

The truth is, I am not at the particular Market 32 that I WAS at, to construct the beginning of the lesson plan. Nah. Still... If the people at the Golub Corporation wanna see something really intricate and detailed that showcases a living example of how I could orchestrate the administration and management of ALL of their stores...?

| 06/27/21 | Advanced System Administration Lab | https://youtu.be/xgffIccX1eg |
| 12/04/21 | [FightingEntropy(π)][FEInfrastructure] | https://youtu.be/6yQr06_rA4I |

Uh, I think that I can easily do that.
What I'm performing in the second video, is a role called “SOLUTIONS ARCHITECT”.

But- essentially it is the same role in the first video.
What I was attempting to do in the first video, was to develop a series of functions that allows the ROUTERS and GATEWAYS to be remotely controllable from the module, [FightingEntropy(π)], by using FREE SOFTWARE called: FreeBSD, OpnSense, pfSense, and various other offshoots of FreeBSD/HardenedBSD.

The first and second video showcase ALL of the FOLLOWING fields of EXPERTISE...

Application Development	Virtualization	Network Engineering
Network Security	Hardware Magistration	Cross-platform Development
Advanced Mathematics	Security Engineering	Graphic Design

BSD's tsch is quite a lot like the COMMAND LINE INTERFACE for Cisco equipment, it's essentially identical to UNICS/UNIX. Whether that Cisco equipment is an old Cisco Catalyst like what is seen in THIS particular video...

| 03/01/18 | Network Troubleshooting 101 | https://youtu.be/0nEiGijjOEY |

...or it's a Cisco Aironet 1142 like as seen in this picture...

| Cisco Aironet 1142 | https://drive.google.com/file/d/1J9l973CcyAvgMXz8trAjruAlgEMlvUcf |

Daniel Pickett from NFRASTRUCTURE (my former employer), did us a really big favor by SELLING COMPUTER ANSWERS this particular (make/model) access point that was used in the following pictures...

Note: I had to reflash the access point from LWAPP mode to standalone mode because we did not have the access point controller. We actually bought like, a LOT of these access points from ProTek Recycling when it USED to be in Troy, off of 1st Ave or something...

But yeh, the following pictures were taken with my Apple iPhone 8+ that had PEGASUS/PHANTOM deployed to it... when I took these pictures AND uploaded these pictures to my Google Drive account, from this particular access point that I configured back in 2018.

05/25/20 2329	IMG_0636	https://drive.google.com/file/d/1a-lb9MOUKi1wy9c4cEEyuclH_rQIMhNo
05/25/20 2329	IMG_0637	https://drive.google.com/file/d/1ZNmufDVX7Xkyf4pHqQfPk2Ww2tvkwGCL
05/25/20 2329	IMG_0638	https://drive.google.com/file/d/1uIxufETfzgpM1uLp9mclF4quMkWak4LY
05/25/20 2329	IMG_0639	https://drive.google.com/file/d/1EL_JllhbHWTkYTPAm595SxjhMyRF5vKP

Now, allow me to explain what these pictures are.
They fall under this term called “EVIDENCE”, and they are ALSO a part of these following pictures...

05/25/20 2343	IMG_0646	https://drive.google.com/file/d/1Lb8RLYUsJnnKnTOHbunlyBmidIXycjVD
05/25/20 2343	IMG_0647	(OBSTRUCTION OF JUSTICE -> 05/26/20 0005 (MISSING VIDEO)
05/26/20 0005	IMG_0648	https://drive.google.com/file/d/18xllhtJW6XZhxJOZXWtesywn-Ph37KK9
05/26/20 0011	IMG_0649	https://drive.google.com/file/d/1W0234ojNChSpwDZWnWPzjjZRBQ2CQm0L
05/26/20 0011	IMG_0650	https://drive.google.com/file/d/1vu2bhSSCv2HO-HCeCCh5-iqcYpiiqC2l
05/26/20 0011	IMG_0651	https://drive.google.com/file/d/1imYzaTA--eVDMeSM-dHfYBfC2tiAHsLV
05/26/20 0348	IMG_0652	https://drive.google.com/file/d/1w0Q6lhLYH9ACwQfUosucUE9x5-uAsNzI

So, what THOSE pictures are, is of (2) serial killers that were using a program called PEGASUS/PHANTOM, who I recorded a VIDEO of, and showed the video to NYSP Trooper Shaemus Leavey on 05/27/20. For whatever reason...? My device was (REMOTELY DISABLED/LOCKED) by the MANUFACTURER OF THE DEVICE, 5 minutes after I showed that police officer the video of the murder attempt.

As for SHAEMUS LEAVEY...? I'm not calling him a moron.
However, uh- the police in the town of Clifton Park, NY...? They're f****** morons.

They don't get it. I have experience that I've exhibited in all of these videos and exhibits, right...?
It's as if they're mentally challenged, and they can't make the CORRELATIONS...

I think it really comes down to the same exact concept of how people FAIL TO TAKE ACTION, ya know...?
Bill Gates went and told people for like, YEARS, that humanity was just NOT QUITE READY for the next pandemic.
I went and told people for like, YEARS, that (2) serial killers tried to kill me outside of my old job.

The COMPARISON I am making is this... Humanity has a SERIOUS PROBLEM WITH LAZINESS.

And you know what...?
Maybe it's the WAY that I told them that a couple dudes hacked my network at COMPUTER ANSWERS on 01/15/2019, and the WAY that I told them just wasn't all that INTERESTING or COOL. So, they paid NO ATTENTION and wrote off the HIGHLY SOPHISTICATED ATTACK that involved:

1) CVE-2019-8936
2) Distributed Denial of Service
3) Panther/Task Scheduler script based attack
4) Derivative of WannaCry
5) Pegasus/Phantom

I guess the REAL problem is this... why would anyone want to kill ME...?

Here's the answer.
I think that some people are blown away by how SKILLED and TALENTED I am, at what I do... right...?
It's because I'm a meticulous son of a bitch, and some people actually get INSANELY JEALOUS by my work ethic.
That's why.

The police are not very meticulous at all... they don't know what the term RUSSIAN CYBERCRIMINALS even means.
Neither do the judges that I've had to deal with...?
Neither do the social services workers...?
Neither do the doctors I've encountered...?

Except Dr. Samuel Bastian, he definitely was thorough. He might not know what RUSSIAN CYBERCRIMINALS are...?
But that's ok. He has literally read a Diagnostic and Statistical Manual version 5 before, and I'm damn certain of that because he took (3) hours to ask me a LOT of questions way back in September 2021, and then had to wait (3) total weeks or so, to get the results back from this “insanity” test requested by PAUL PELAGALLI.

Discounting Dr. Samuel Bastian, here's what ALL of those people's idea on what METICULOUS means:

have sex with a girl for about (2) seconds flat → ejaculate

Then, they're basically done. That's how “METICULOUS” the (POLICE/JUDGES/DOCTORS/LAWYERS/etc.) truly are, in (CLIFTON PARK, NY/SARATOGA COUNTY). And, that's ok.

It probably sounds like I'm INSULTING THEIR INTEGRITY or DUE DILIGENCE, right...? Well... think again.
I'm complimenting them on their extremely thorough nature... At no point whatsoever, have they bothered to collect or examine any of these exhibits. That's a pretty thorough job, right...?

Yeah, I believe that the Russian Mafia, who PAVEL ZAICHENKO is affiliated with, attempted to have me killed.
And that, the local police basically fail to realize what this particular record truly means...

05/26/20 0130	SCSO-2020-028501
	https://github.com/mcc85s/FightingEntropy/blob/main/Records/2020-028501%20Cook%20req.pdf

Return / Correlations

Since I have all the output I really need from the lesson plan, I can talk about how it can be used.
Most of the code in the class Wireless is meant for managing wireless profiles, interfaces, generating XML, accessing task objects, requesting access to the radios, syncing up the radios, and then scanning. It also has methods for connecting, disconnecting, effectively using it without the GUI is not exactly something

I've thought to exhibit, so this will be my attempt to do just that.

Wireless networking is pretty important to understand in today's world.
Mobile phones, tablets, laptops, and various other devices are used, and they typically take advantage of wireless access points. Sometimes adapters/radios perform multiple roles, so they'll be able to access BLUETOOTH, WIFI, and CELL RADIO. Those are all wireless technologies (though far from ALL of them).

The PITFALL with these wireless devices, is that they can certainly be hacked and attacked.
I have experienced first hand, when my Apple iPhone 8+ had ALL of it's radios turned off...
...but my device still received information from the network.

That's mainly because software controlled radios are a pretty scary concept, and they cannot be turned off.
The phone might SAY that the radio is off...? But there is no physical switch on the outside that allows that.
Unless you're a SMARTPHONE TECHNICIAN such as myself, who can open the device and unplug the wires that plug into the radios...? You have NO CONTROL over whether that device is SENDING/RECEIVING information.

Even with the power switch.
No control whatsoever.
The phones these days all have a LOW-POWER mode/state, where even if you power it off, it is still on.

In reality, a misconfigured access point can allow anyone who's walking down the road, to have access to that particular network, and then suddenly, that network is basically making itself available to everybody.

The truth is, the Cisco Meraki configuration at Market 32 has people click a SINGLE BUTTON on a website, in order to authenticate that device on the network. It doesn't AUTOMATICALLY allow devices onto the network without that push of a button.

Not all network engineers are created equal. Sometimes a business such as Dunkin Donuts, or Joe Shmoe's business at random address, Awesometown, NY...? They're too busy being awesome in that town, to realize that Joe Shmoe's access point was allowing everybody that drove by his business, to have access to his network. What is not told to people, is that Joe Shmoe doesn't know how to fix that. Nobody does.

It's like a black hole just sitting in the distance... nobody can do anything about the black hole.
If you go near it...? You're getting spaghettified, and you'll see the entire future of the universe unfold as you are pulled into the edge of infinity, where the rules of spacetime and the laws of physics are told to piss off. Something else governs that singularity, dudeface.

Let's not be wicked dramatic about what happens around Joe Shmoe's misconfigured access point.
The chances are, few people are going to camp out at Joe Shmoe's network waiting for innocent victims to meander by the black hole. However, uh- that can actually lead to situations where someone who DOES know what they're doing, can set up a (device/program) that just collects information.

That's not good, because if a device connects to a misconfigured access point that's just AUTOMATICALLY inviting every passerby to their birthday party...? They're gonna wind up having a lot of strangers at this “birthday party” that they all got rick-rolled into attending, whether that was WITTINGLY or UNWITTINGLY.

The truth is, sometimes I occasionally walk by a business and suddenly, I have internet.
I get messages that I didn't expect to get at that point in time.

Now, is it somebody from Anonymous that's basically keeping tabs on me, and just giving me updates...?
Or, is it just your standard, run-of-the-mill, underprovisioned access point that's telling you that you've got internet NOW, bud. Yeah. You didn't have internet BEFORE...? But, you've got internet NOW.

Now, you're ready to stream movies or YouTube videos.
Now what...?
I'll return to the variable $WiFi

```
PS Prompt:\> $WiFi

Adapters  : {MSFT_NetAdapter (CreationClassName = "MSFT_NetAdapter", DeviceID = "{E3A47A46-9920-469E-9...}
Request   : System.Threading.Tasks.Task`1[Windows.Devices.Radios.RadioAccessStatus]
Radios    : System.Threading.Tasks.Task`1[System.Collections.Generic.IReadOnlyList`1[Windows.Devices.R...
List      : System.Threading.Tasks.Task`1[System.Collections.Generic.IReadOnlyList`1[Windows.Devices...
Output    : {Subway, SubSecure, Uncommon Grounds, SKMW...}
Selected  :
Connected :
```

This is a different location, so obviously the information will not be the same.
Most of the information in the property $Wifi.Output from up above at Market 32, were all internal networks at Market 32, networks that control the point-of-sales, the credit card terminals, receipt printers, security

cameras, and perhaps various other desktop/office based equipment and stuff like that, and THOSE devices have to be on a DIFFERENT subnetwork. They MAY or MAY NOT be on a different SUBNET, or they may be on another virtual lan, or they may be on a totally separate network that is kept separate even though the wires and servers are all the same... when it comes to GUESSING how a businesses network is set up, I have to make SOME speculations in relation to the observations I've made about the business in particular.

This is actually referred to as a SITE SURVEY.
I provided these SITE SURVEYS when I managed the Computer Answers Business Solutions division.

What I commonly found on these SITE SURVEYS, is that most people don't know what the hell they're doing.
Further to that point, the internet service providers will not actually touch any internal networks whatsoever.
So in reference to SPECTRUM, the farthest they will go is this:

/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯\

```
Spectrum : Uh, is the light on, on the device...?
You      : I see a lot of lights, buddy.
Spectrum : There should be like (5) lights on that model of the cable modem.
You      : What's a cable modem...?
Spectrum : Ugh...
           We can send a service technician to your house in like (5) days, if you set up an appointment.
You      : I need internet like right now though...
Spectrum : *shakes head* Look, pal.
           Everybody needs internet right now.
           You're just not THAT special.
You      : I'm not...?
Spectrum : I mean, we get thousands of people calling in saying “I don't have internet...”
You      : It was workin' like, a half hour ago...?
Spectrum : Yeah, but it says right here that there's no service disruption in your area.
You      : So, if I gotta wait (5) days for a technician, can I get like (5) days off of my bill...?
Spectrum : Nah, I'm afraid NOT, sir.
           Gotta pay for the internet even if it's not working correctly or at all.
You      : What if I terminate my account, and then set up an appointment for it to be reinstalled...?
Spectrum : You can't do that.
You      : Why not...?
Spectrum : Cause, that's gotta be illegal or something.
You      : So, you charge me for (5) days that the internet doesn't work...?
Spectrum : Yeah man.
           I mean the ONLY OTHER ALTERNATIVE...
           ...is that you figure out where the cable modem is...
           ...and like, unplug the power, and then plug it back in.
You      : ...that's it...?
Spectrum : I mean, it MIGHT resolve the issue, that's why it's part of the automated voice system.
You      : Well, I guess...
           Let's give it a shot.
```

__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/

However, that's about as basic as their assistance goes. They're not there to troubleshoot a server that isn't printing invoices on a network printer. They're not there to touch your DHCP server or DNS configuration settings if you're using an enterprise routing solution like PFSense or OpnSense.

```
------------------------------------------------------------------------
| 07/31/17 | Spectrum Cable Modem Reset | https://youtu.be/LfZW-s0BMow |
------------------------------------------------------------------------
```

That's what is being used in this particular video, though the shop was SORT OF messy at the time, I was having an issue where employees would say “HEY I HAVE NO INTERNET AND I CAN'T SUBMIT CREDIT CARD PAYMENTS...”

So then, I had to show them how to put their big boy pants on...?
And like... reset the cable modem.

Wicked challenging stuff.
1) Unplug the power wire...?
2) Plug it back in...?
3) Wait like a couple minutes...
4) Go type in IPCONFIG /RELEASE in the command prompt
5) Then type IPCONFIG /RENEW
6) Check if the internet is like, back...
7) Have (1) less reason to not be doing work when I'm not around.

That was a thing sometimes. PAVEL ZAICHENKO, the OWNER of the company, never did this. Nah, the dude flat out, expected that problems would resolve themselves, and that the money the company made, made itself. No real effort or thought put INTO maintaining the daily operations at the company... So if somebody didn't have internet...?

Oh no. What can anybody do...? Can't just call Michael Cook anymore, since I left the company for a KEY REASON.
Someone called the OWNER OF THE COMPANY was committing LABOR FRAUD and TAX EVASION, and nobody cared.

I mean, even the NEW YORK STATE DEPARTMENT OF LABOR, like “INVESTIGATED” the LABOR FRAUD, like, WHEN I RECORDED THAT SPECIFIC VIDEO UP ABOVE... right...? But, he got away with it because all of the employees basically kept their mouths shut during the investigation.

I gotta tell ya... At some point in the future, I might physically meet someone that believes all of this.
MAYBE. POSSIBLY. It's possible that I'll never in MY lifetime, ever meet anybody that understands any of this...
And then I'll just go out into the cold, dark edge of infinity... where nobody knows what the hell is going on.

In reference to Market 32, uh- I know a guy who's basically been employed by the Golub Corporation for a large number of years. I know a few people who've been employed by the Golub Corporation... my cousin worked there, a few people I went to school with ALSO worked there... But the guy I happen to be referring to, when I say...

Me : Oh, I know a guy who's been employed by the Golub Corporation for a large number of years...

Dude used to have to answer an actual phone, from time to time.
Someone would be having problems at any branch in the area...
And they would say:

/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯\

Person : Hey, Mike...
My register isn't like, working correctly, I keep punching in keys-
Mike : Numlock key.
Person : ...on the keyboard-
Mike : Numlock key, try pushing that.
Person : Uh...
Ok.
pushes the Numlock key
Alright, I pushed the Numlock key...
Mike : Try it now.
Person : Alright...
tries it again
Oh, hey~!
Look at that, it's workin' now~!
Mike : That's just your standard-issue, Numlock key not being on.
Person : Wow dude...
How'd you know that's what it was...?
Mike : *puts his sunglasses on* Buddy, when is it NOT the numlock key...?
Person : Well, thanks Mike.
Mike : Not a problem buddy, take care.

__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/¯¯__/

I've written about this guy a number of times.
His name isn't Robert Paulsen or anything like that... So, you're not likely to meet a group of dudes who just start chanting his name, saying “His name, was Robert Paulsen”.

But, when you throw down the gauntlet like I do from time to time...?
Casually mentioning that you've known a guy named Michael Philipsak for like, 30+ years or so...?
That usually gets some interest from somebody somewhere...

Yeah, dude's been a loyal employee of the Golub Corporation, for a large number of years.
He does not have a bat cave, as far as I know, but if he does, he's done a fantastic job of keeping it a secret.

← Dude used to drive around in his white Price Chopper van that had the old logo on it.

Just a friggen axe chopping prices haphazardly, left and right...

Price on something way too high...?
“Yeah, I don't think so, pal...”
chopped that price

Something else cost way too much...?
“Yeah right, dude.”
chopped an additional price

You would think after like, decades of chopping prices left and right, that these people would have had enough of the chaos... but- not from what I can tell. The war against prices that are just WAY TOO HIGH, is seemingly, never ending.

Mike does still drive around in a white van, but it says Golub Corporation on it.
Mercedez Benz Sprinters that the Amazon Corporation ALSO drives around in.

Anyway, I'm tangenting.
Lets dive into the other properties of the variable $Wifi

```
PS Prompt:\> $WiFi.Adapters

Name      InterfaceDescription                    ifIndex Status       MacAddress             LinkSpeed
----      --------------------                    ------- ------       ----------             ---------
Wi-Fi     1x1 11bgn Wireless LAN PCI Express H...      22 Disconnected 9C-B7-0D-20-08-FE        72 Mbps

PS Prompt:\>
```

```
# // ____________________________________________________________________________________________________
# // |--------------------------------------------------------------------------------------------------|
# // | So, this is covering an adapter in my laptop which isn't integrated into the motherboard.       |
# // | That means at any time, I can swap it out with another wireless adapter and not have to worry. |
# // |                                                                                                  |
# // | Doesn't necessarily mean that I'm PARANOID and feel a need to swap out the WiFi adapter...       |
# // | ...I'm just saying, with a SMARTPHONE that MOST PEOPLE HAVE...                                   |
# // | Then, uh-oh. Chances are, if someone's got your MAC ADDRESS...?                                  |
# // | You'll need a brand new phone/adapter, and someone to service it, if someone has your device    |
# // | address pegged.                                                                                  |
# // |                                                                                                  |
# // | Uh-oh. What can anybody do if that's the case...?                                                |
# // | That's actually a real problem.                                                                  |
# // | Most people will think NOTHING OF IT, and be like "Whatever bro... Nobody's watchin' me."        |
# // |--------------------------------------------------------------------------------------------------|
# // ----------------------------------------------------------------------------------------------------
```

```
PS Prompt:\> $Wifi.Request

Result                 : Allowed
Id                     : 841
Exception              :
Status                 : RanToCompletion
IsCanceled             : False
IsCompleted            : True
CreationOptions        : None
AsyncState             :
IsFaulted              : False
AsyncWaitHandle        : System.Threading.ManualResetEvent
CompletedSynchronously : False

PS Prompt:\>
```

```
# // ____________________________________________________________________________________________________
# // |--------------------------------------------------------------------------------------------------|
# // | This is essentially a task object, it is not all that different from a PowerShell runspace...   |
# // | or, IDK how to explain the concept of MULTITHREADING, but when a processor is focused on         |
# // | a particular TASK...? That's the OBJECT that the system uses, in order to accomplish its'        |
# // | multiple goals, in tandem.                                                                       |
# // |                                                                                                  |
# // | Asynchronous operations are basically like this...                                               |
# // |                                                                                                  |
# // | Suppose you have (8) friends, right...?                                                          |
# // | They all live somewhere.                                                                         |
# // | If you tell them ALL, to:                                                                        |
# // | 1) Go home                                                                                       |
# // | 2) change their clothes                                                                          |
```

```
# // | 3) come right back...                                                                                    |
# // |                                                                                                          |
# // | Chances are that they will NOT all return at the same exact time.                                        |
# // | That's because they all have different routes to travel, and they may all                                |
# // | encounter different circumstances...                                                                     |
# // | Yeah, the ACTIVITY that they are all doing is the same...?                                               |
# // | But- the actual work they each individually have to do, is NOT the same.                                 |
# // |                                                                                                          |
# // | That's the idea behind asynchronous operations.                                                          |
# // |                                                                                                          |
# // | That's what this is, right here.                                                                         |
# // |__________________________________________________________________________________________________________|
# // ----------------------------------------------------------------------------------------------------------
```

```
PS Prompt:\> $Wifi.Radios

Result                 : System.__ComObject
Id                     : 668
Exception              :
Status                 : RanToCompletion
IsCanceled             : False
IsCompleted            : True
CreationOptions        : None
AsyncState             :
IsFaulted              : False
AsyncWaitHandle        : System.Threading.ManualResetEvent
CompletedSynchronously : False

PS Prompt:\>
```

```
# // ____________________________________________________________________________
# // | This is the same type of object as up above. However, the TASK is different. |
# // ----------------------------------------------------------------------------
```

```
PS Prompt:\> $Wifi.List

Result                 : System.__ComObject
Id                     : 669
Exception              :
Status                 : RanToCompletion
IsCanceled             : False
IsCompleted            : True
CreationOptions        : None
AsyncState             :
IsFaulted              : False
AsyncWaitHandle        : System.Threading.ManualResetEvent
CompletedSynchronously : False

PS Prompt:\>
```

```
# // ____________________________________________________________________________
# // | Again, this is the same type of object. However, the TASK is different. |
# // ----------------------------------------------------------------------------
```

```
PS Prompt:\> $WiFi.Output | Format-Table

Index Name                Bssid       Type     Uptime          Netw. Auth.     Enc. Str.
----- ----                -----       ----     ------          ----- ---------- ---- ----
    0 Subway          ... CE:2D:E0... 802.11n  7d 18h 57m 59s  Infr. Open80211 None    4
    1 SubSecure       ... CC:2D:E0... 802.11n  7d 18h 57m 59s  Infr. RsnaPsk   Ccmp    4
    2 Uncommon Grounds... F0:9F:C2... 802.11n  7d 19h 04m 12s  Infr. Open80211 None    4
    3 SKMW            ... E2:CB:FC... 802.11n  13h 23m 38s     Infr. RsnaPsk   Ccmp    3
    4                 ... E2:CB:FC... 802.11n  13h 23m 39s     Infr. RsnaPsk   Ccmp    3
    5 CCR's Wi-Fi Netw... C4:B3:01... 802.11n  7d 18h 57m 52s  Infr. RsnaPsk   Ccmp    3
```

```
    6 Continuum       ... 38:4C:90... 802.11n  7d 18h 54m 02s  Infr. RsnaPsk   Ccmp    3
    7 SK-Guest        ... E2:CB:FC... 802.11n  13h 23m 38s     Infr. Open80211 None    3
    8                 ... F8:ED:A5... 802.11n  7d 18h 57m 08s  Infr. RsnaPsk   Ccmp    3
    9                 ... 6C:B0:CE... 802.11ac 7d 18h 56m 53s  Infr. RsnaPsk   Ccmp    3
   10 Uncommon Grounds... D4:05:98... 802.11n  7d 18h 53m 25s  Infr. RsnaPsk   Ccmp    3
   11 YANE100         ... BE:75:36... 802.11n  00h 09m 51s     Infr. RsnaPsk   Ccmp    3
   12 iCryo           ... 14:59:C0... 802.11ac 26d 00h 11m 04s Infr. RsnaPsk   Ccmp    2
   13 hum07823        ... B8:9F:09... 802.11n  01h 38m 31s     Infr. RsnaPsk   Ccmp    2
```

```
# // ____________________________________________________________________
# // | Here, we've got the same information that I posted WAY up above. |
# // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

```
So, I'm about to go over the components of how versatile this friggen variable $WiFi truly is.
Some people might assume that all variables are created equal.

Yeah, I don't think so.
This variable right here does a lot of stuff.
It is effectively, able to control ALL of the stuff I just pasted, plus, do a hell of a lot more.

Methods, and properties differ in this following description.

Sometimes the hottest girls out there are good at one thing only. Being really hot... that's it.
So if you want a hot girlfriend that COOKS, CLEANS, and DOES USEFUL STUFF...?
You have to determine if they know more than how to just do their hair or nails.

It's merely COSMETIC in APPEARANCE when some hot girl comes along and is like:

/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\

Girl : OoooHHhhhH, look at me.
       Hot as hell.
You  : Yeah, I see that.
Girl : *smacks her own fanny*
       Look at all these goods I've got, bub.
       MAD hot.
You  : Alright...?
       Can you like, count to 10...?
Girl : Obviously, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
       Done.
You  : Alright.
       Do you know the alphabet...?
Girl : Obviously, A, B, C, D... X, Y, Z.
You  : Uh, you skipped a whole bunch.
Girl : Yeah, but that doesn't matter, obviously I know them all.
You  : ...what are the remaining letters...?
Girl : *gets frustrated* LOOK PAL, I KNOW MY ALPHABET, ALRIGHT...?
You  : So, what IS it...?
Girl : A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z.
You  : Alright, so...
       Can you COOK FOOD THAT I'LL BE WILLING TO EAT EVERY DAY...?
Girl : Uh, Idk, honestly.
You  : Alright...?
       So, you know your numbers, and letters, but, can't cook food...?
Girl : Whatever bro.
       I'm STILL mad hot...
You  : Yeah, but if you get PREGNANT with a CHILD, can you CARE for it...?

\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/

Believe it or not, this TANGENT is somewhat relevant.
Mainly because some variables, or classes/types, they only do (1) thing.
They hold a PROPERTY and a VALUE.
They're not very complicated, and they can be overwhelmed pretty easily.

Whereas, this thing that I just assembled, talked about, discussed, and examined...?
It does a lot more than just saying numbers, letters, and cooking food. Nah. This sorta teaches basic rudimentary
stuff as to HOW/WHY businesses: operate. Looking good is only a PORTION of what allows something to WORK/SURVIVE.
```

However, uh- I can tell you first hand, a lot of people in society don't really care to know about how things work. They just want things to work and for it to not be all that complicated. So if something IS complicated, then you need to be wicked cool in order to cause people to want to know something complicated.

So if you're NOT COOL...? Then, nobody will care about those complicated things, because...
That's pretty complicated stuff and you just OVERLOAD people with information.

Overloading people with information IS NOT NECESSARILY A BAD THING, but society teaches people that it IS.
Overloading people with information is NECESSARY to have a business that survives and thrives.
Overloading people with STRATEGIES, PARADIGMS, CONSTRUCTS, METHODS, PROPERTIES, and things of that nature allow something to be COMPLEX, while also allowing the USE of such information, clean and accessible.

It's like this.
If you want something to be done, you have to know what the hell you're doing, FIRST... and then you can do it.
People don't just start out by doing something they've never done before, and being very successful at it.

Not unless you're like a NATURAL or a GENIUS, and even then...?
The chances of FAILURE is pretty high.

Now, consider the following list of PROPERTIES and METHODS in this particular variable, $WiFi.

```
PS Prompt:\> $Wifi | Get-Member

   TypeName: Wireless

Name                        MemberType Definition
----                        ---------- ----------
Connect                     Method     void Connect(string SSID)
Disconnect                  Method     void Disconnect()
Equals                      Method     bool Equals(System.Object obj)
FormatXml                   Method     System.Object FormatXml(System.Object Co
GetHashCode                 Method     int GetHashCode()
GetType                     Method     type GetType()
GetWiFiConnectionParameter Method     System.Object GetWiFiConnectionParameter
GetWiFiInterface            Method     System.Object[] GetWiFiInterface()
GetWiFiInterfaceGuid        Method     System.Object GetWiFiInterfaceGuid(strin
GetWiFiProfileInfo          Method     System.Object GetWiFiProfileInfo(string
GetWiFiProfileList          Method     System.Object[] GetWiFiProfileList(strin
Hex                         Method     string Hex(string PN)
NetshShowInterface          Method     System.Object NetshShowInterface(string
NewWifiHandle               Method     System.IntPtr NewWifiHandle()
NewWifiProfile              Method     void NewWifiProfile(string PX, string WF
NewWifiProfileEap           Method     System.Object NewWifiProfileEap(string P
NewWiFiProfilePsk           Method     System.Object NewWiFiProfilePsk(string P
NewWifiProfileXml           Method     System.Object NewWifiProfileXml(string P
NewWifiProfileXmlEap        Method     string NewWifiProfileXmlEap(string PN, s
NewWiFiProfileXmlPsk        Method     string NewWiFiProfileXmlPsk(string PN, s
Passphrase                  Method     void Passphrase(System.Object NW)
RaAsync                     Method     System.Object[] RaAsync()
RadioFindAllAdaptersAsync   Method     System.Object RadioFindAllAdaptersAsync(
RadioRequestAccess          Method     System.Object RadioRequestAccess()
RadioSynchronization        Method     System.Object RadioSynchronization()
RaList                      Method     System.Object RaList()
Refresh                     Method     void Refresh()
RefreshAdapterList          Method     System.Object[] RefreshAdapterList()
RemoveWifiHandle            Method     void RemoveWifiHandle(System.IntPtr CH)
RemoveWifiProfile           Method     void RemoveWifiProfile(string PN)
RsAsync                     Method     System.Object[] RsAsync()
RsList                      Method     System.Object RsList()
RxAsync                     Method     System.Object[] RxAsync()
RxStatus                    Method     System.Object RxStatus()
Scan                        Method     void Scan()
Select                      Method     void Select(string D)
Task                        Method     System.Object Task()
ToString                    Method     string ToString()
Unselect                    Method     void Unselect()
Update                      Method     void Update()
WifiConnectionParameter     Method     System.Object WifiConnectionParameter(st
WiFiProfileInfo             Method     System.Object WiFiProfileInfo(string PN,
WiFiReasonCode              Method     string WiFiReasonCode(System.IntPtr RC)
Win32Exception              Method     string Win32Exception(uint32 RC)
```

```
   WlanCloseHandle            Method     System.Object WlanCloseHandle(System.Int
   WlanConnect                Method     void WlanConnect(System.IntPtr HCH, guid
   WlanConnectionFlag         Method     System.Object WlanConnectionFlag(string
   WlanConnectionMode         Method     System.Object WlanConnectionMode(string
   WlanConnectionParams       Method     System.Object WlanConnectionParams()
   WlanDeleteProfile          Method     void WlanDeleteProfile(System.IntPtr CH,
   WlanDisconnect             Method     void WlanDisconnect(System.IntPtr HCH, g
   WlanDot11BssType           Method     System.Object WlanDot11BssType(string D)
   WlanEnumInterfaces         Method     System.Object WlanEnumInterfaces(System.
   WlanFreeMemory             Method     void WlanFreeMemory(System.IntPtr P)
   WlanGetProfile             Method     System.Object WlanGetProfile(System.IntP
   WlanGetProfileList         Method     System.Object WlanGetProfileList(System.
   WlanGetProfileListFromPtr  Method     System.Object[] WlanGetProfileListFromPt
   WlanInterfaceInfo          Method     System.Object WlanInterfaceInfo(System.O
   WlanInterfaceList          Method     System.Object WlanInterfaceList(System.I
   WlanOpenHandle             Method     System.Object WlanOpenHandle(uint32 CV,
   WlanProfileInfoObject      Method     System.Object WlanProfileInfoObject()
   WlanReasonCodeToString     Method     System.Object WlanReasonCodeToString(uin
   WlanSetProfile             Method     System.Object WlanSetProfile(uint32 CH,
   XmlTemplate                Method     System.Object XmlTemplate(uint32 Type)
   Adapters                   Property   System.Object Adapters {get;set;}
   Connected                  Property   System.Object Connected {get;set;}
   List                       Property   System.Object List {get;set;}
   Output                     Property   System.Object Output {get;set;}
   Radios                     Property   System.Object Radios {get;set;}
   Request                    Property   System.Object Request {get;set;}
   Selected                   Property   System.Object Selected {get;set;}

   PS Prompt:\>
```

So, all of the above things that say METHOD...?
They DO stuff.

All of the things that say PROPERTY...?
They ARE stuff.

Thing	Role
Method	Do stuff
Property	Are stuff

That's not confusing or complicated, is it...?
I'm gonna go out on a limb here, and say anybody will probably say “Nah, that's not complicated.”
Might even go so far as to say most people will see that and say “That's pretty easy to understand, actually.”

Methods → Do stuff
Properties → Are stuff

The STUFF that they DO or ARE...? That can be pretty complex. The stuff they DO or ARE, can require a lot of complicated things called “RULES”, or like “LOGIC”, to be written, in order for the stuff they DO or ARE, to make any sense.

I'm gonna leave it at that for today.

Michael C. Cook Sr.
Security Engineer
Secure Digits Plus LLC