```
  ____    __________________________________________________________________________________________________
 //¯¯\\__//¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\\___
 \\__//¯¯¯ Get-ViperBomb [~] AKA System Extension Control Utility                                    ___//¯¯\\
  ¯¯¯\\__________________________________________________________________________________________________//¯¯\\__//
      ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯    ¯¯¯¯
```

Introduction

Greetings,

My name is Michael C. Cook Sr., and I am an:

| APPLICATION DEVELOPER | DESIGN ARTIST | SYSTEM + NETWORK + SECURITY ENGINEER | SOLUTIONS ARCHITECT |

Here's my resume (Note: I have ALSO written a PROGRAM that BUILDS/RENDERS this resume, in the console.)

| 10/10/22 | https://github.com/mcc85s/FightingEntropy/blob/main/Docs/2022_1010-(MCC%20Short%20Resume).pdf |

Over the course of the last (3) years or so, I have been developing a module using [PowerShell], but I don't use [PowerShell] the same way that everybody else tends to use it. Ohhhhhh no.

I use a combination of [C#/CSharp], [.NET Framework], [XAML/Extensible Application Markup Language], and various other techniques to (implement + use) unmanaged code within [PowerShell].

Of the people who use [PowerShell], most may be CASUAL USERS...
Some may be MASTERS.
Few may be EXPERTS.
A handful may be Aficianado's, and Architects...
And there are probably less than (10) people in the entire world... that can WRITE THEIR OWN CODE, who can either MATCH or EXCEED the capabilities of what I am able to do, with [PowerShell]... beyond the actual people who work at [Microsoft] who write a lot of code for [PowerShell].

Simply put, there are not many people in the world that are taking this thing to the absolute maximum level.
Most people will resort to using [C#/CSharp] and [.NET Framework], to accomplish what I do in [PowerShell].

```
 _________________________________________________________
|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|
| Person : *eyebrows up* ...really...?                    |
| Me     : Yyyyyyyup.                                     |
| Person : Wow.                                           |
|          Sounds impressive as heck, dude...             |
| Me     : *chuckles* Heh.                                |
|          Just *points* YOU wait, pal...                 |
| Person : *pauses* ...for what...?                       |
| Me     : ...for things to be taken to the next level.   |
|_________________________________________________________|
 ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

Probably sounds rather suspenseful... and intense.

Allow me to reel things back a fair amount.
There's a reason why I'm establishing these details in this introduction.
I've been working on a handful of utilities that can be seen in this video...

Date	Name	Url
10/28/22	[FightingEntropy(π)][2022.10.1]	[https://youtu.be/S7k4lZdPE-I]

Additional videos of this module and my (portfolio/capabilities) are available at the readme markdown file at the base of my GitHub project...

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

Overview

So, this particular utility is a microcosm of the entire [FightingEntropy(π)] module I have been developing, and it is the System Control Extension Utility, which is intended to be rather similar to DISM++. In greater detail, what this function CURRENTLY is, is a utility that is able to review:

[System details]

- [+] Bios information
- [+] Operating system
- [+] Computer system
- [+] Processors
- [+] Disks
- [+] Network adapters

[System services]

- [+] Running services
- [+] Black Viper's service configuration
- [+] MadBomb122's extension of his utility
- [+] Logging
- [+] Registry settings
- [+] Backup components

[System controls]

- [+] MadBomb122's Windows 10 script
- [+] Extended capabilities of that utility
- [+] Extended GUI (I've collaborated with MadBomb122, and wrote the XAML GUI for both of his tools)

[...with many more to come]

- [+] Event log utility (I have been keeping this on the backburner)
- [+] Multi-threading dispatcher that utilizes “autothreading”
- [+] Archiving and exporting various (system/logging) details
- [+] Importing system snapshots
- [+] Programs and settings transfer (similar to Laplink PCMover)

To be clear, I still have plenty of adjustments to make to many components of this module.
It's not something that I'm trying to cobble together with rushed results.

Simply put, this will be an expertly crafted utility from top to bottom, with (0) games being played.
It will perform [Windows Server/Client] migration type activities AND will be a key [FightingEntropy(π)] feature,
using the [PowerShell Deployment] modification for the [Microsoft Deployment Toolkit] demonstrated in this video:

Date	Name	Url
12/05/21	[FightingEntropy(π)][FEInfrastructure]	[https://youtu.be/6yQr06_rA4I]

Some people might read that and say:

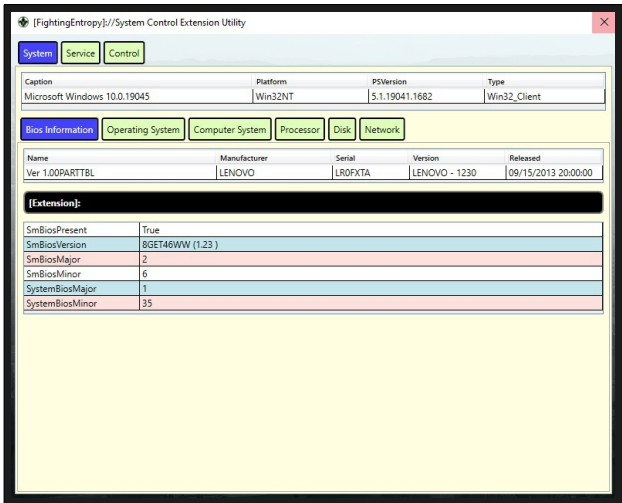
```
|-------------------------------------------------------------------------------|
| Person : (0) games being played...?                                           |
| Me     : That's right.                                                        |
|          (0).                                                                 |
| Person : ...but I like to play games, though...                               |
| Me     : That's fine.                                                         |
|          You can play games BEFORE and AFTER using the utility, when it's ready. |
| Person : *pauses* But, not WHILE using it...?                                 |
| Me     : *shakes head* I'm afraid not...                                      |
|          Gotta make sure the job is done correctly, otherwise...              |
|          ...you won't be able to play your games AFTER you use it.            |
| Person : Oh, bummer.                                                          |
| Me     : ...you'll be able to play your games AFTER you use it...             |
| Person : Oh, yeah, that's cool.                                               |
| Me     : Yeah, you'll be able to play your games.                             |
|          You wouldn't want your games to play you, right...?                  |
| Person : *pauses( ...nah.                                                     |
| Me     : *puts sunglasses on* Well, alright then, dudeface.                   |
|_______________________________________________________________________________|
```

In the next section, are some screenshots of the utility as it currently exists.
In the sections following that, I will cover the code within the function as it currently exists.

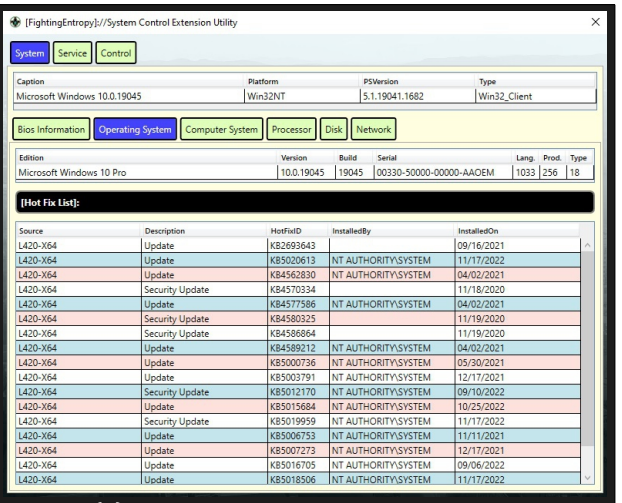
_____/ Overview

Screenshots

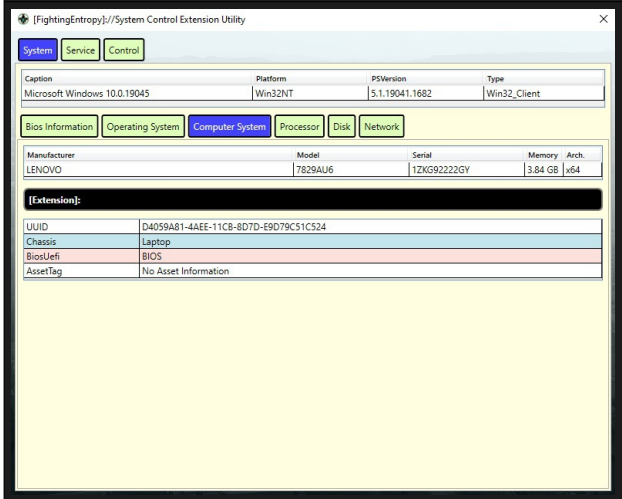
To view a larger version of the image, click on it. (Must download the PDF to traverse the hyperlinks)



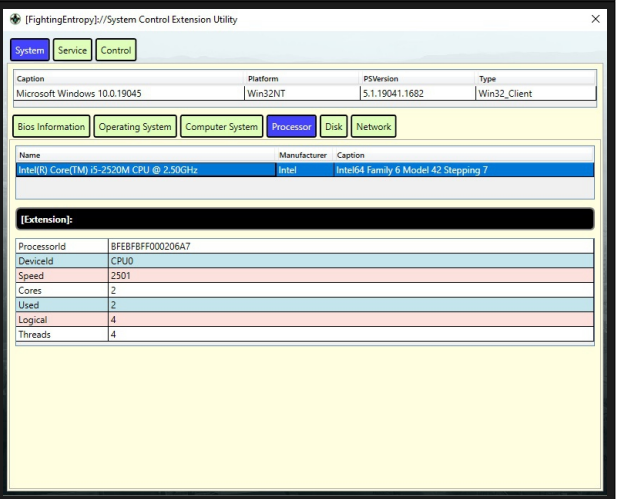
(1) System tab, OS/[PowerShell] Info + Bios Info subtab



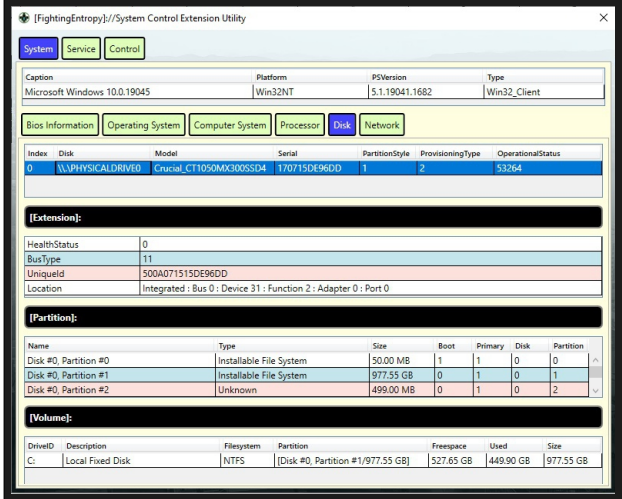
(2) System tab, Operating System subtab



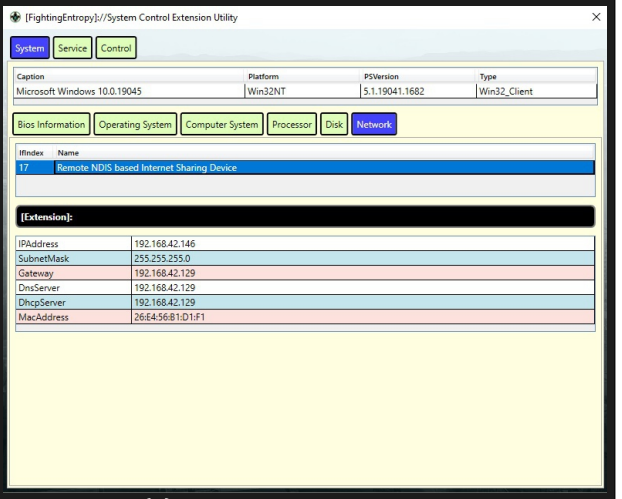
(3) System tab, Computer System subtab



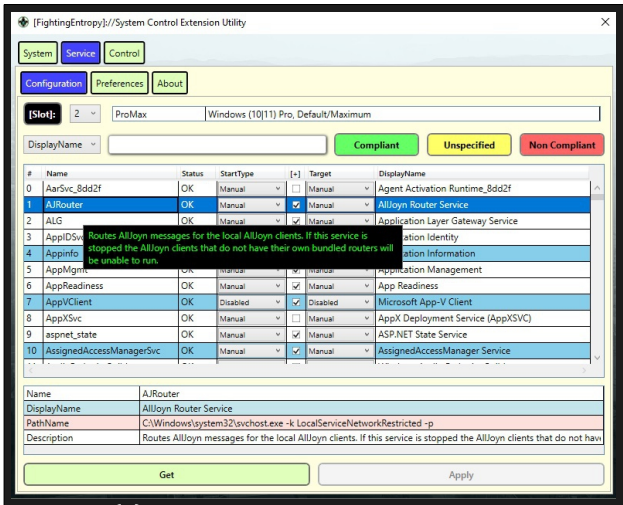
(4) System tab, Processor(s) subtab



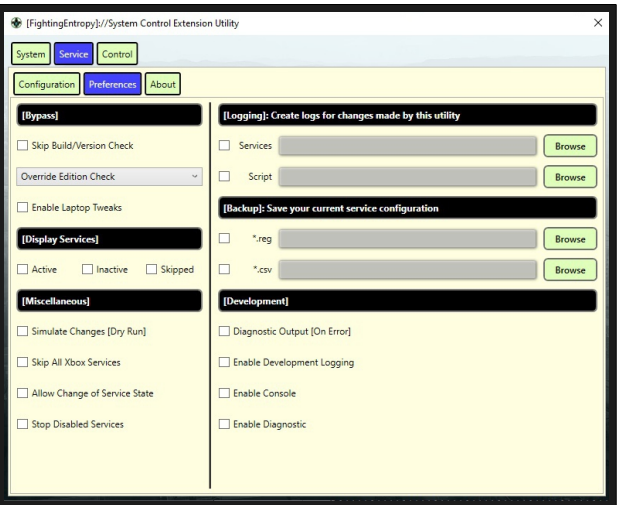
(5) System tab, Disks subtab



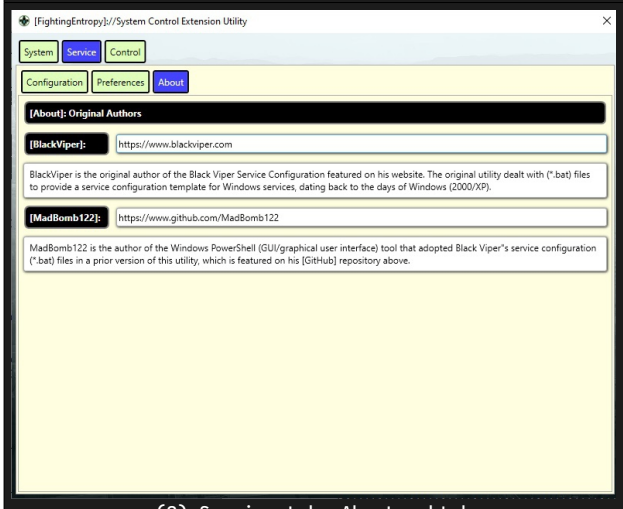
(6) System tab, Network subtab



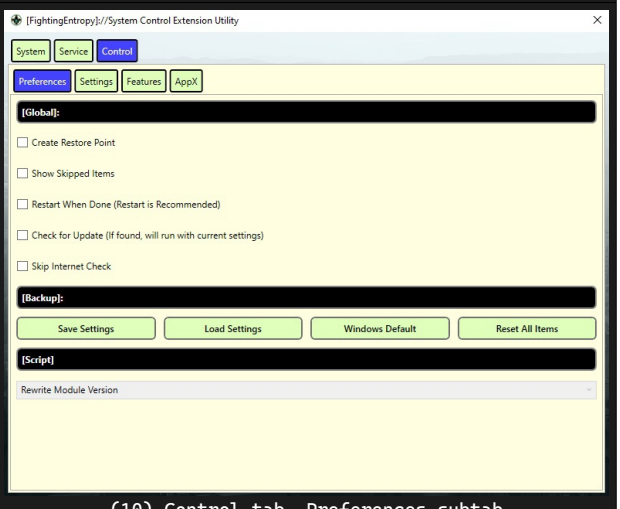
(7) Service tab, Configuration subtab



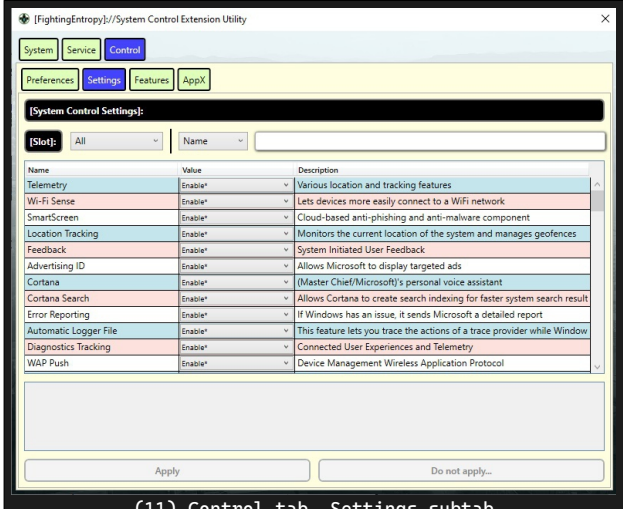
(8) Service tab, Preferences subtab



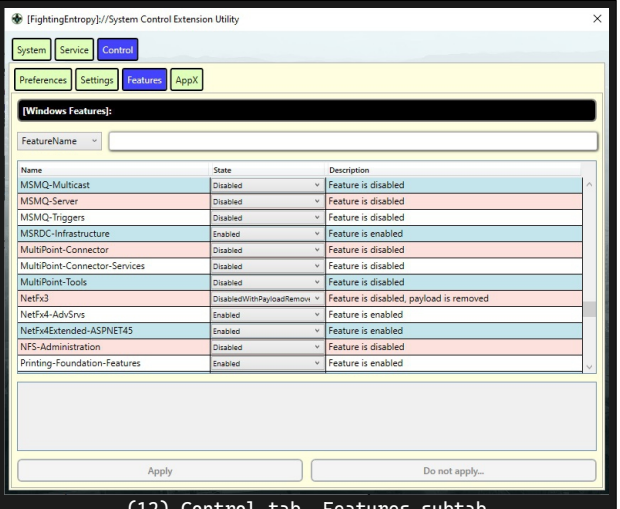
(9) Service tab, About subtab



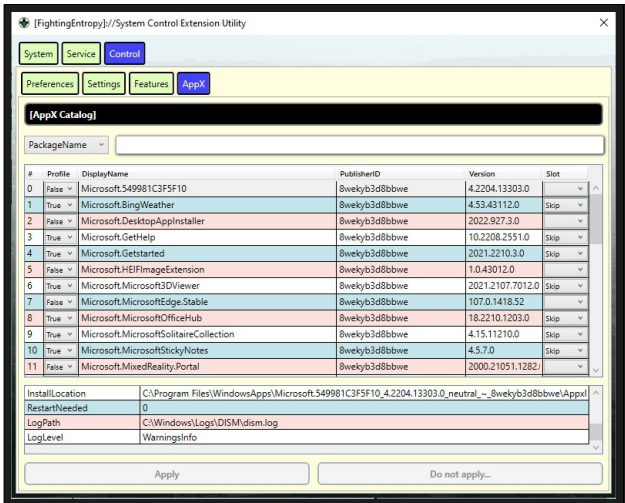
(10) Control tab, Preferences subtab



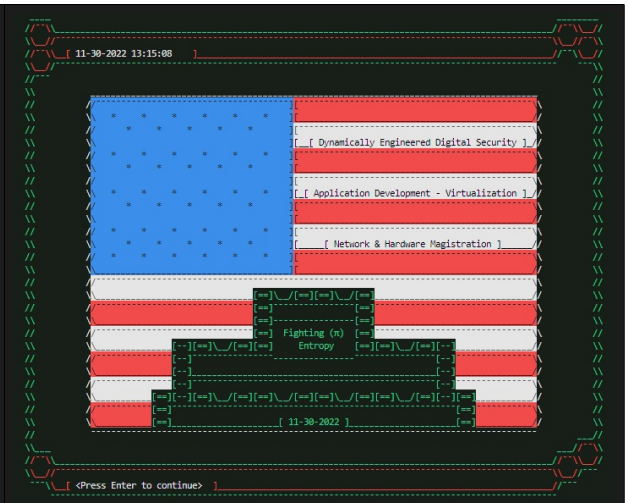
(11) Control tab, Settings subtab



(12) Control tab, Features subtab



(13) Control tab, AppX subtab



(14) [FightingEntropy(π)] Write-Theme -Flag

In the following sections, I will provide the code behind that produces the objects powering the GUI in the above screenshots [01] through [13].

After these sections, I will cover the console (objects/output) that the below code, produces.

Class [ViperBombXaml] Screenshots

This class is the Xaml object for the (GUI/graphical user interface).
It is strictly meant to take the Static [String] $Content, and hand it off to a class that creates the window.

Since the beginning of having to write this document, I've altered the format of the following XAML so that there is an absolute minimum amount of text-wrapping between lines.

```
Class ViperBombXaml
{
    Static [String] $Content = @(
    '<Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" ',
    '        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" ',
    '        Name="Window" ',
    '        Title="[FightingEntropy]://System Control Extension Utility" ',
    '        Height="640" ',
    '        Width="800" ',
    '        Topmost="True" ',
    '        BorderBrush="Black" ',
    '        Icon="C:\ProgramData\Secure Digits Plus LLC\FightingEntropy\2022.11.0\Graphics\icon.ico" ',
    '        ResizeMode="NoResize" ',
    '        HorizontalAlignment="Center" ',
    '        WindowStartupLocation="CenterScreen">',
    '    <Window.Resources>',
    '        <Style x:Key="DropShadow">',
    '            <Setter Property="TextBlock.Effect">',
    '                <Setter.Value>',
    '                    <DropShadowEffect ShadowDepth="1"/>',
    '                </Setter.Value>',
    '            </Setter>',
    '        </Style>',
    '        <Style TargetType="{x:Type TextBox}" BasedOn="{StaticResource DropShadow}">',
    '            <Setter Property="TextBlock.TextAlignment" Value="Left"/>',
    '            <Setter Property="VerticalContentAlignment" Value="Center"/>',
    '            <Setter Property="HorizontalContentAlignment" Value="Left"/>',
    '            <Setter Property="Height" Value="24"/>',
    '            <Setter Property="Margin" Value="4"/>',
    '            <Setter Property="FontSize" Value="12"/>',
    '            <Setter Property="Foreground" Value="#000000"/>',
    '            <Setter Property="TextWrapping" Value="Wrap"/>',
    '            <Style.Resources>',
```

```
'                <Style TargetType="Border">',
'                    <Setter Property="CornerRadius" Value="2"/>',
'                </Style>',
'            </Style.Resources>',
'        </Style>',
'        <Style TargetType="{x:Type PasswordBox}" BasedOn="{StaticResource DropShadow}">',
'            <Setter Property="TextBlock.TextAlignment" Value="Left"/>',
'            <Setter Property="VerticalContentAlignment" Value="Center"/>',
'            <Setter Property="HorizontalContentAlignment" Value="Left"/>',
'            <Setter Property="Margin" Value="4"/>',
'            <Setter Property="Height" Value="24"/>',
'        </Style>',
'        <Style TargetType="CheckBox">',
'            <Setter Property="VerticalContentAlignment" Value="Center"/>',
'            <Setter Property="Height" Value="24"/>',
'            <Setter Property="Margin" Value="5"/>',
'        </Style>',
'        <Style TargetType="ToolTip">',
'            <Setter Property="Background" Value="#000000"/>',
'            <Setter Property="Foreground" Value="#66D066"/>',
'        </Style>',
'        <Style TargetType="TabItem">',
'            <Setter Property="Template">',
'                <Setter.Value>',
'                    <ControlTemplate TargetType="TabItem">',
'                        <Border Name="Border" ',
'                                BorderThickness="2" ',
'                                BorderBrush="Black" ',
'                                CornerRadius="2" ',
'                                Margin="2">',
'                            <ContentPresenter x:Name="ContentSite" ',
'                                              VerticalAlignment="Center" ',
'                                              HorizontalAlignment="Right" ',
'                                              ContentSource="Header" ',
'                                              Margin="5"/>',
'                        </Border>',
'                        <ControlTemplate.Triggers>',
'                            <Trigger Property="IsSelected" Value="True">',
'                                <Setter TargetName="Border" Property="Background" Value="#4444FF"/>',
'                                <Setter Property="Foreground" Value="#FFFFFF"/>',
'                            </Trigger>',
'                            <Trigger Property="IsSelected" Value="False">',
'                                <Setter TargetName="Border" Property="Background" Value="#DFFFBA"/>',
'                                <Setter Property="Foreground" Value="#000000"/>',
'                            </Trigger>',
'                            <Trigger Property="IsEnabled" Value="False">',
'                                <Setter TargetName="Border" Property="Background" Value="#6F6F6F"/>',
'                                <Setter Property="Foreground" Value="#9F9F9F"/>',
'                            </Trigger>',
'                        </ControlTemplate.Triggers>',
'                    </ControlTemplate>',
'                </Setter.Value>',
'            </Setter>',
'        </Style>',
'        <Style TargetType="Button">',
'            <Setter Property="Margin" Value="5"/>',
'            <Setter Property="Padding" Value="5"/>',
'            <Setter Property="Height" Value="30"/>',
'            <Setter Property="FontWeight" Value="Semibold"/>',
'            <Setter Property="FontSize" Value="12"/>',
'            <Setter Property="Foreground" Value="Black"/>',
'            <Setter Property="Background" Value="#DFFFBA"/>',
'            <Setter Property="BorderThickness" Value="2"/>',
'            <Setter Property="VerticalContentAlignment" Value="Center"/>',
'            <Style.Resources>',
'                <Style TargetType="Border">',
'                    <Setter Property="CornerRadius" Value="5"/>',
'                </Style>',
'            </Style.Resources>',
'        </Style>',
'        <Style TargetType="ComboBox">',
'            <Setter Property="Height" Value="24"/>',
```

```
'            <Setter Property="Margin" Value="5"/>',
'            <Setter Property="FontSize" Value="12"/>',
'            <Setter Property="FontWeight" Value="Normal"/>',
'        </Style>',
'        <Style TargetType="TabControl">',
'            <Setter Property="TabStripPlacement" Value="Top"/>',
'            <Setter Property="HorizontalContentAlignment" Value="Center"/>',
'            <Setter Property="Background" Value="LightYellow"/>',
'        </Style>',
'        <Style TargetType="GroupBox">',
'            <Setter Property="Margin" Value="5"/>',
'            <Setter Property="Padding" Value="5"/>',
'            <Setter Property="BorderThickness" Value="2"/>',
'            <Setter Property="BorderBrush" Value="Black"/>',
'            <Setter Property="Foreground" Value="Black"/>',
'        </Style>',
'        <Style TargetType="TextBox" x:Key="Block">',
'            <Setter Property="Margin" Value="5"/>',
'            <Setter Property="Padding" Value="5"/>',
'            <Setter Property="FontFamily" Value="Consolas"/>',
'            <Setter Property="Height" Value="180"/>',
'            <Setter Property="FontSize" Value="10"/>',
'            <Setter Property="FontWeight" Value="Normal"/>',
'            <Setter Property="AcceptsReturn" Value="True"/>',
'            <Setter Property="VerticalAlignment" Value="Top"/>',
'            <Setter Property="TextAlignment" Value="Left"/>',
'            <Setter Property="VerticalContentAlignment" Value="Top"/>',
'            <Setter Property="VerticalScrollBarVisibility" Value="Visible"/>',
'            <Setter Property="TextBlock.Effect">',
'                <Setter.Value>',
'                    <DropShadowEffect ShadowDepth="1"/>',
'                </Setter.Value>',
'            </Setter>',
'        </Style>',
'        <Style TargetType="ComboBox" x:Key="DGCombo">',
'            <Setter Property="Margin" Value="0"/>',
'            <Setter Property="Padding" Value="2"/>',
'            <Setter Property="Height" Value="18"/>',
'            <Setter Property="FontSize" Value="10"/>',
'            <Setter Property="VerticalContentAlignment" Value="Center"/>',
'        </Style>',
'        <Style TargetType="DataGrid">',
'            <Setter Property="Margin" Value="5"/>',
'            <Setter Property="AutoGenerateColumns" Value="False"/>',
'            <Setter Property="AlternationCount" Value="3"/>',
'            <Setter Property="HeadersVisibility" Value="Column"/>',
'            <Setter Property="CanUserResizeRows" Value="False"/>',
'            <Setter Property="CanUserAddRows" Value="False"/>',
'            <Setter Property="IsReadOnly" Value="True"/>',
'            <Setter Property="IsTabStop" Value="True"/>',
'            <Setter Property="IsTextSearchEnabled" Value="True"/>',
'            <Setter Property="SelectionMode" Value="Extended"/>',
'            <Setter Property="ScrollViewer.CanContentScroll" Value="True"/>',
'            <Setter Property="ScrollViewer.VerticalScrollBarVisibility" Value="Auto"/>',
'            <Setter Property="ScrollViewer.HorizontalScrollBarVisibility" Value="Auto"/>',
'        </Style>',
'        <Style TargetType="DataGridRow">',
'            <Setter Property="BorderBrush" Value="Black"/>',
'            <Style.Triggers>',
'                <Trigger Property="AlternationIndex" Value="0">',
'                    <Setter Property="Background" Value="White"/>',
'                </Trigger>',
'                <Trigger Property="AlternationIndex" Value="1">',
'                    <Setter Property="Background" Value="#FFC5E5EC"/>',
'                </Trigger>',
'                <Trigger Property="AlternationIndex" Value="2">',
'                    <Setter Property="Background" Value="#FFFDE1DC"/>',
'                </Trigger>',
'            </Style.Triggers>',
'        </Style>',
'        <Style TargetType="DataGridColumnHeader">',
'            <Setter Property="FontSize"   Value="10"/>',
```

```
    '            <Setter Property="FontWeight" Value="Medium"/>',
    '            <Setter Property="Margin" Value="2"/>',
    '            <Setter Property="Padding" Value="2"/>',
    '        </Style>',
    '        <Style TargetType="Label">',
    '            <Setter Property="Margin" Value="5"/>',
    '            <Setter Property="FontWeight" Value="Bold"/>',
    '            <Setter Property="FontSize" Value="12"/>',
    '            <Setter Property="Background" Value="Black"/>',
    '            <Setter Property="Foreground" Value="White"/>',
    '            <Setter Property="BorderBrush" Value="Gray"/>',
    '            <Setter Property="BorderThickness" Value="2"/>',
    '            <Setter Property="VerticalContentAlignment" Value="Center"/>',
    '            <Style.Resources>',
    '                <Style TargetType="Border">',
    '                    <Setter Property="CornerRadius" Value="5"/>',
    '                </Style>',
    '            </Style.Resources>',
    '        </Style>',
    '    </Window.Resources>',
    '    <Grid>',
    '        <Grid.Background>',
    '            <ImageBrush Stretch="UniformToFill" ',
    '                        ImageSource="C:\ProgramData\Secure Digits Plus
LLC\FightingEntropy\2022.11.0\Graphics\background.jpg"/>',
    '        </Grid.Background>',
    '        <TabControl Margin="5">',
    '            <TabItem Header="System">',
    '                <Grid>',
    '                    <Grid.RowDefinitions>',
    '                        <RowDefinition Height="60"/>',
    '                        <RowDefinition Height="*"/>',
    '                    </Grid.RowDefinitions>',
    '                    <DataGrid Grid.Row="0" Name="OS">',
    '                        <DataGrid.Columns>',
    '                            <DataGridTextColumn Header="Caption"',
    '                                                Width="300"',
    '                                                Binding="{Binding Caption}"/>',
    '                            <DataGridTextColumn Header="Platform"',
    '                                                Width="150"',
    '                                                Binding="{Binding Platform}"/>',
    '                            <DataGridTextColumn Header="PSVersion" ',
    '                                                Width="150"',
    '                                                Binding="{Binding PSVersion}"/>',
    '                            <DataGridTextColumn Header="Type"',
    '                                                Width="*"',
    '                                                Binding="{Binding Type}"/>',
    '                        </DataGrid.Columns>',
    '                    </DataGrid>',
    '                    <TabControl Grid.Row="1">',
    '                        <TabItem Header="Bios Information">',
    '                            <Grid>',
    '                                <Grid.RowDefinitions>',
    '                                    <RowDefinition Height="55"/>',
    '                                    <RowDefinition Height="40"/>',
    '                                    <RowDefinition Height="130"/>',
    '                                </Grid.RowDefinitions>',
    '                                <DataGrid Grid.Row="0" Name="BiosInformation">',
    '                                    <DataGrid.Columns>',
    '                                        <DataGridTextColumn Header="Name"',
    '                                                            Width="*"',
    '                                                            Binding="{Binding Name}"/>',
    '                                        <DataGridTextColumn Header="Manufacturer"',
    '                                                            Width="150"',
    '                                                            Binding="{Binding Manufacturer}"/>',
    '                                        <DataGridTextColumn Header="Serial"',
    '                                                            Width="100"',
    '                                                            Binding="{Binding SerialNumber}"/>',
    '                                        <DataGridTextColumn Header="Version"',
    '                                                            Width="125"',
    '                                                            Binding="{Binding Version}"/>',
    '                                        <DataGridTextColumn Header="Released"',
```

```
'                                                   Width="125"',
'                                                   Binding="{Binding ReleaseDate}"/>',
'                                </DataGrid.Columns>',
'                            </DataGrid>',
'                            <Label Grid.Row="1" Content="[Extension]:"/>',
'                            <DataGrid Grid.Row="2"',
'                                      Name="BiosInformationExtension"',
'                                      HeadersVisibility="None">',
'                                <DataGrid.Columns>',
'                                    <DataGridTextColumn Header="Name"',
'                                                        Width="150"',
'                                                        Binding="{Binding Name}"/>',
'                                    <DataGridTextColumn Header="Value"',
'                                                        Width="*"',
'                                                        Binding="{Binding Value}"/>',
'                                </DataGrid.Columns>',
'                            </DataGrid>',
'                        </Grid>',
'                    </TabItem>',
'                    <TabItem Header="Operating System">',
'                        <Grid>',
'                            <Grid.RowDefinitions>',
'                                <RowDefinition Height="55"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="*"/>',
'                            </Grid.RowDefinitions>',
'                            <DataGrid Grid.Row="0" Name="OperatingSystem">',
'                                <DataGrid.Columns>',
'                                    <DataGridTextColumn Header="Edition"',
'                                                        Width="*"',
'                                                        Binding="{Binding Caption}"/>',
'                                    <DataGridTextColumn Header="Version"',
'                                                        Width="75"',
'                                                        Binding="{Binding Version}"/>',
'                                    <DataGridTextColumn Header="Build"',
'                                                        Width="50"',
'                                                        Binding="{Binding Build}"/>',
'                                    <DataGridTextColumn Header="Serial"',
'                                                        Width="180"',
'                                                        Binding="{Binding Serial}"/>',
'                                    <DataGridTextColumn Header="Lang."',
'                                                        Width="35"',
'                                                        Binding="{Binding Language}"/>',
'                                    <DataGridTextColumn Header="Prod."',
'                                                        Width="35"',
'                                                        Binding="{Binding Product}"/>',
'                                    <DataGridTextColumn Header="Type"',
'                                                        Width="35"',
'                                                        Binding="{Binding Type}"/>',
'                                </DataGrid.Columns>',
'                            </DataGrid>',
'                            <Label Grid.Row="1" Content="[Hot Fix List]:"/>',
'                            <DataGrid Grid.Row="2" Name="HotFix">',
'                                <DataGrid.Columns>',
'                                    <DataGridTextColumn Header="Source"',
'                                                        Binding="{Binding Source}"',
'                                                        Width="*"/>',
'                                    <DataGridTextColumn Header="Description"',
'                                                        Binding="{Binding Description}"',
'                                                        Width="*"/>',
'                                    <DataGridTextColumn Header="HotFixID"',
'                                                        Binding="{Binding HotFixID}"',
'                                                        Width="80"/>',
'                                    <DataGridTextColumn Header="InstalledBy"',
'                                                        Binding="{Binding InstalledBy}"',
'                                                        Width="*"/>',
'                                    <DataGridTextColumn Header="InstalledOn"',
'                                                        Binding="{Binding InstalledOn}"',
'                                                        Width="*"/>',
'                                </DataGrid.Columns>',
'                            </DataGrid>',
'                        </Grid>',
```

```
    '                    </TabItem>',
    '                    <TabItem Header="Computer System">',
    '                        <Grid>',
    '                            <Grid.RowDefinitions>',
    '                                <RowDefinition Height="55"/>',
    '                                <RowDefinition Height="40"/>',
    '                                <RowDefinition Height="90"/>',
    '                            </Grid.RowDefinitions>',
    '                            <DataGrid Grid.Row="0" Name="ComputerSystem">',
    '                                <DataGrid.Columns>',
    '                                    <DataGridTextColumn Header="Manufacturer"',
    '                                                        Width="*"',
    '                                                        Binding="{Binding Manufacturer}"/>',
    '                                    <DataGridTextColumn Header="Model"',
    '                                                        Width="150"',
    '                                                        Binding="{Binding Model}"/>',
    '                                    <DataGridTextColumn Header="Serial"',
    '                                                        Width="150"',
    '                                                        Binding="{Binding Serial}"/>',
    '                                    <DataGridTextColumn Header="Memory"',
    '                                                        Width="50"',
    '                                                        Binding="{Binding Memory}"/>',
    '                                    <DataGridTextColumn Header="Arch."',
    '                                                        Width="50"',
    '                                                        Binding="{Binding Architecture}"/>',
    '                                </DataGrid.Columns>',
    '                            </DataGrid>',
    '                            <Label Grid.Row="1" Content="[Extension]:"/>',
    '                            <DataGrid Grid.Row="2"',
    '                                      Name="ComputerSystemExtension"',
    '                                      HeadersVisibility="None">',
    '                                <DataGrid.Columns>',
    '                                    <DataGridTextColumn Header="Name"',
    '                                                        Width="150"',
    '                                                        Binding="{Binding Name}"/>',
    '                                    <DataGridTextColumn Header="Value"',
    '                                                        Width="*"',
    '                                                        Binding="{Binding Value}"/>',
    '                                </DataGrid.Columns>',
    '                            </DataGrid>',
    '                        </Grid>',
    '                    </TabItem>',
    '                    <TabItem Header="Processor">',
    '                        <Grid>',
    '                            <Grid.RowDefinitions>',
    '                                <RowDefinition Height="80"/>',
    '                                <RowDefinition Height="40"/>',
    '                                <RowDefinition Height="150"/>',
    '                            </Grid.RowDefinitions>',
    '                            <DataGrid Grid.Row="0" Name="Processor">',
    '                                <DataGrid.Columns>',
    '                                    <DataGridTextColumn Header="Name"',
    '                                                        Width="*"',
    '                                                        Binding="{Binding Name}"/>',
    '                                    <DataGridTextColumn Header="Manufacturer"',
    '                                                        Width="75"',
    '                                                        Binding="{Binding Manufacturer}"/>',
    '                                    <DataGridTextColumn Header="Caption"',
    '                                                        Width="*"',
    '                                                        Binding="{Binding Caption}"/>',
    '                                </DataGrid.Columns>',
    '                            </DataGrid>',
    '                            <Label Grid.Row="1" Content="[Extension]:"/>',
    '                            <DataGrid Grid.Row="2"',
    '                                      Name="ProcessorExtension"',
    '                                      HeadersVisibility="None">',
    '                                <DataGrid.Columns>',
    '                                    <DataGridTextColumn Header="Name"',
    '                                                        Width="120"',
    '                                                        Binding="{Binding Name}"/>',
    '                                    <DataGridTextColumn Header="Value"',
    '                                                        Width= "*"',
```

```
'                                                   Binding="{Binding Value}"/>',
'                            </DataGrid.Columns>',
'                        </DataGrid>',
'                    </Grid>',
'                </TabItem>',
'                <TabItem Header="Disk">',
'                    <Grid>',
'                        <Grid.RowDefinitions>',
'                            <RowDefinition Height="80"/>',
'                            <RowDefinition Height="40"/>',
'                            <RowDefinition Height="90"/>',
'                            <RowDefinition Height="40"/>',
'                            <RowDefinition Height="90"/>',
'                            <RowDefinition Height="40"/>',
'                            <RowDefinition Height="80"/>',
'                        </Grid.RowDefinitions>',
'                        <DataGrid Grid.Row="0" RowHeaderWidth="0" Name="Disk">',
'                            <DataGrid.Columns>',
'                                <DataGridTextColumn Header="Index"',
'                                                    Width= "40"',
'                                                    Binding="{Binding Index}"/>',
'                                <DataGridTextColumn Header="Disk"',
'                                                    Width="125"',
'                                                    Binding="{Binding Disk}"/>',
'                                <DataGridTextColumn Header="Model"',
'                                                    Width="160"',
'                                                    Binding="{Binding Model}"/>',
'                                <DataGridTextColumn Header="Serial"',
'                                                    Width="110"',
'                                                    Binding="{Binding Serial}"/>',
'                                <DataGridTextColumn Header="PartitionStyle"',
'                                                    Width="75"',
'                                                    Binding="{Binding PartitionStyle}"/>',
'                                <DataGridTextColumn Header="ProvisioningType"',
'                                                    Width="100"',
'                                                    Binding="{Binding ProvisioningType}"/>',
'                                <DataGridTextColumn Header="OperationalStatus"',
'                                                    Width="140"',
'                                                    Binding="{Binding OperationalStatus}"/>',
'                            </DataGrid.Columns>',
'                        </DataGrid>',
'                        <Label Grid.Row="1" Content="[Extension]:"/>',
'                        <DataGrid Grid.Row="2" Name="DiskExtension" HeadersVisibility="None">',
'                            <DataGrid.Columns>',
'                                <DataGridTextColumn Header="Name"',
'                                                    Width="150"',
'                                                    Binding="{Binding Name}"/>',
'                                <DataGridTextColumn Header="Value"',
'                                                    Width="*"',
'                                                    Binding="{Binding Value}"/>',
'                            </DataGrid.Columns>',
'                        </DataGrid>',
'                        <Label Grid.Row="3" Content="[Partition]:"/>',
'                        <DataGrid Grid.Row="4" Name="DiskPartition">',
'                            <DataGrid.Columns>',
'                                <DataGridTextColumn Header="Name"',
'                                                    Width="*"',
'                                                    Binding="{Binding Name}"/>',
'                                <DataGridTextColumn Header="Type"',
'                                                    Width="200"',
'                                                    Binding="{Binding Type}"/>',
'                                <DataGridTextColumn Header="Size"',
'                                                    Width="85"',
'                                                    Binding="{Binding Size}"/>',
'                                <DataGridTextColumn Header="Boot"',
'                                                    Width="50"',
'                                                    Binding="{Binding Boot}"/>',
'                                <DataGridTextColumn Header="Primary"',
'                                                    Width="50"',
'                                                    Binding="{Binding Primary}"/>',
'                                <DataGridTextColumn Header="Disk"',
'                                                    Width="50"',
```

```
'                                                   Binding="{Binding Disk}"/>',
'                                <DataGridTextColumn Header="Partition"',
'                                                   Width="50"',
'                                                   Binding="{Binding Partition}"/>',
'                            </DataGrid.Columns>',
'                        </DataGrid>',
'                        <Label Grid.Row="5" Content="[Volume]:"/>',
'                        <DataGrid Grid.Row="6" Name="DiskVolume">',
'                            <DataGrid.Columns>',
'                                <DataGridTextColumn Header="DriveID"',
'                                                   Width="50"',
'                                                   Binding="{Binding DriveID}"/>',
'                                <DataGridTextColumn Header="Description"',
'                                                   Width="*"',
'                                                   Binding="{Binding Description}"/>',
'                                <DataGridTextColumn Header="Filesystem"',
'                                                   Width="70"',
'                                                   Binding="{Binding Filesystem}"/>',
'                                <DataGridTextColumn Header="Partition"',
'                                                   Width="200"',
'                                                   Binding="{Binding Partition}"/>',
'                                <DataGridTextColumn Header="Freespace"',
'                                                   Width= "75"',
'                                                   Binding="{Binding Freespace}"/>',
'                                <DataGridTextColumn Header="Used"',
'                                                   Width= "75"',
'                                                   Binding="{Binding Used}"/>',
'                                <DataGridTextColumn Header="Size"',
'                                                   Width= "75"',
'                                                   Binding="{Binding Size}"/>',
'                            </DataGrid.Columns>',
'                        </DataGrid>',
'                    </Grid>',
'                </TabItem>',
'                <TabItem Header="Network">',
'                    <Grid>',
'                        <Grid.RowDefinitions>',
'                            <RowDefinition Height="80"/>',
'                            <RowDefinition Height="40"/>',
'                            <RowDefinition Height="135"/>',
'                        </Grid.RowDefinitions>',
'                        <DataGrid Grid.Row="0" Name="Network">',
'                            <DataGrid.Columns>',
'                                <DataGridTextColumn Header="IfIndex"',
'                                                   Width="50"',
'                                                   Binding="{Binding Index}"/>',
'                                <DataGridTextColumn Header="Name"',
'                                                   Width="*"',
'                                                   Binding="{Binding Name}"/>',
'                            </DataGrid.Columns>',
'                        </DataGrid>',
'                        <Label Grid.Row="1" Content="[Extension]:"/>',
'                        <DataGrid Grid.Row="2" ',
'                                  Name="NetworkExtension"',
'                                  RowHeaderWidth="0"',
'                                  HeadersVisibility="None">',
'                            <DataGrid.Columns>',
'                                <DataGridTextColumn Header="Name"',
'                                                   Width="150"',
'                                                   Binding="{Binding Name}"/>',
'                                <DataGridTextColumn Header="Value"',
'                                                   Width="*"',
'                                                   Binding="{Binding Value}"/>',
'                            </DataGrid.Columns>',
'                        </DataGrid>',
'                    </Grid>',
'                </TabItem>',
'            </TabControl>',
'        </Grid>',
'    </TabItem>',
'    <TabItem Header="Service">',
'        <TabControl>',
```

```
'                    <TabItem Header="Configuration">',
'                        <Grid>',
'                            <Grid.RowDefinitions>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="*"/>',
'                                <RowDefinition Height="100"/>',
'                                <RowDefinition Height="40"/>',
'                            </Grid.RowDefinitions>',
'                            <Grid Grid.Row="0">',
'                                <Grid.ColumnDefinitions>',
'                                    <ColumnDefinition Width="60"/>',
'                                    <ColumnDefinition Width="50"/>',
'                                    <ColumnDefinition Width="*"/>',
'                                </Grid.ColumnDefinitions>',
'                                <Label Content="[Slot]:"/>',
'                                <ComboBox Grid.Column="1" Name="ServiceSlot"/>',
'                                <DataGrid Grid.Column="2" ',
'                                          Name="ServiceDisplay" ',
'                                          HeadersVisibility="None"',
'                                          Margin="10">',
'                                    <DataGrid.Columns>',
'                                        <DataGridTextColumn Header="Type"',
'                                                            Binding="{Binding Type}"',
'                                                            Width="120"/>',
'                                        <DataGridTextColumn Header="Description"',
'                                                            Binding="{Binding Description}"',
'                                                            Width="*"/>',
'                                    </DataGrid.Columns>',
'                                </DataGrid>',
'                            </Grid>',
'                            <Grid Grid.Row="1">',
'                                <Grid.ColumnDefinitions>',
'                                    <ColumnDefinition Width="110"/>',
'                                    <ColumnDefinition Width="*"/>',
'                                    <ColumnDefinition Width="120"/>',
'                                    <ColumnDefinition Width="120"/>',
'                                    <ColumnDefinition Width="120"/>',
'                                </Grid.ColumnDefinitions>',
'                                <ComboBox Grid.Column="0"',
'                                          Margin="5"',
'                                          Name="ServiceProperty"',
'                                          VerticalAlignment="Center"',
'                                          SelectedIndex="1">',
'                                    <ComboBoxItem Content="Name"/>',
'                                    <ComboBoxItem Content="DisplayName"/>',
'                                </ComboBox>',
'                                <TextBox Grid.Column="1" Margin="5" Name="ServiceFilter"/>',
'                                <Label Grid.Column="2" ',
'                                       Background="#66FF66"',
'                                       Foreground="Black"',
'                                       HorizontalContentAlignment="Center"',
'                                       Content="Compliant"/>',
'                                <Label Grid.Column="3"',
'                                       Background="#FFFF66"',
'                                       Foreground="Black"',
'                                       HorizontalContentAlignment="Center"',
'                                       Content="Unspecified"/>',
'                                <Label Grid.Column="4"',
'                                       Background="#FF6666"',
'                                       Foreground="Black"',
'                                       HorizontalContentAlignment="Center"',
'                                       Content="Non Compliant"/>',
'                            </Grid>',
'                            <DataGrid Grid.Row="2" ',
'                                      Grid.Column="0" ',
'                                      Name="Service" ',
'                                      RowHeaderWidth="0"',
'                                      ScrollViewer.CanContentScroll="True"',
'                                      ScrollViewer.IsDeferredScrollingEnabled="True"',
'                                      ScrollViewer.HorizontalScrollBarVisibility="Visible">',
'                                <DataGrid.RowStyle>',
```

```
'                                <Style TargetType="{x:Type DataGridRow}">',
'                                    <Style.Triggers>',
'                                        <Trigger Property="AlternationIndex" Value="0">',
'                                            <Setter Property="Background" Value="White"/>',
'                                        </Trigger>',
'                                        <Trigger Property="AlternationIndex" Value="1">',
'                                            <Setter Property="Background" Value="SkyBlue"/>',
'                                        </Trigger>',
'                                        <Trigger Property="IsMouseOver" Value="True">',
'                                            <Setter Property="ToolTip">',
'                                                <Setter.Value>',
'                                                    <TextBlock Text="{Binding Description}" ',
'                                                               TextWrapping="Wrap" ',
'                                                               Width="400" ',
'                                                               Background="#000000" ',
'                                                               Foreground="#00FF00"/>',
'                                                </Setter.Value>',
'                                            </Setter>',
'                                            <Setter Property="ToolTipService.ShowDuration" ',
'                                                    Value="360000000"/>',
'                                        </Trigger>',
'                                        <MultiDataTrigger>',
'                                            <MultiDataTrigger.Conditions>',
'                                                <Condition Binding="{Binding Scope}" Value="1"/>',
'                                                <Condition Binding="{Binding Match}" Value="0"/>',
'                                            </MultiDataTrigger.Conditions>',
'                                            <Setter Property="Background" Value="#F08080"/>',
'                                        </MultiDataTrigger>',
'                                        <MultiDataTrigger>',
'                                            <MultiDataTrigger.Conditions>',
'                                                <Condition Binding="{Binding Scope}" Value="0"/>',
'                                                <Condition Binding="{Binding Match}" Value="0"/>',
'                                            </MultiDataTrigger.Conditions>',
'                                            <Setter Property="Background" Value="#FFFFFF64"/>',
'                                        </MultiDataTrigger>',
'                                        <MultiDataTrigger>',
'                                            <MultiDataTrigger.Conditions>',
'                                                <Condition Binding="{Binding Scope}" Value="1"/>',
'                                                <Condition Binding="{Binding Match}" Value="1"/>',
'                                            </MultiDataTrigger.Conditions>',
'                                            <Setter Property="Background" Value="LightGreen"/>',
'                                        </MultiDataTrigger>',
'                                    </Style.Triggers>',
'                                </Style>',
'                            </DataGrid.RowStyle>',
'                            <DataGrid.Columns>',
'                                <DataGridTextColumn Header="#"',
'                                                    Width="25"',
'                                                    Binding="{Binding Index}"/>',
'                                <DataGridTextColumn Header="Name"',
'                                                    Width="175"',
'                                                    Binding="{Binding Name}"/>',
'                                <DataGridTextColumn Header="Status"',
'                                                    Width="50"',
'                                                    Binding="{Binding Status}"/>',
'                                <DataGridTemplateColumn Header="StartType" Width="90">',
'                                    <DataGridTemplateColumn.CellTemplate>',
'                                        <DataTemplate>',
'                                            <ComboBox SelectedIndex="{Binding StartMode.Index}"',
'                                                      Margin="0"',
'                                                      Padding="2"',
'                                                      Height="18"',
'                                                      FontSize="10"',
'                                                      VerticalContentAlignment="Center">',
'                                                <ComboBoxItem Content="Skip"/>',
'                                                <ComboBoxItem Content="Disabled"/>',
'                                                <ComboBoxItem Content="Manual"/>',
'                                                <ComboBoxItem Content="Auto"/>',
'                                                <ComboBoxItem Content="Auto Delayed"/>',
'                                            </ComboBox>',
'                                        </DataTemplate>',
'                                    </DataGridTemplateColumn.CellTemplate>',
```

```
    '                                </DataGridTemplateColumn>',
    '                                <DataGridTemplateColumn Header="[+]"     Width="25">',
    '                                    <DataGridTemplateColumn.CellTemplate>',
    '                                        <DataTemplate>',
    '                                            <CheckBox IsChecked="{Binding Scope}"',
    '                                                      Margin="0"',
    '                                                      HorizontalAlignment="Center">',
    '                                                <CheckBox.LayoutTransform>',
    '                                                    <ScaleTransform ScaleX="0.75" ScaleY="0.75"/>',
    '                                                </CheckBox.LayoutTransform>',
    '                                            </CheckBox>',
    '                                        </DataTemplate>',
    '                                    </DataGridTemplateColumn.CellTemplate>',
    '                                </DataGridTemplateColumn>',
    '                                <DataGridTemplateColumn Header="Target" Width="90">',
    '                                    <DataGridTemplateColumn.CellTemplate>',
    '                                        <DataTemplate>',
    '                                            <ComboBox SelectedIndex="{Binding Target.Index}"',
    '                                                      Margin="0"',
    '                                                      Padding="2"',
    '                                                      Height="18"',
    '                                                      FontSize="10"',
    '                                                      VerticalContentAlignment="Center">',
    '                                                <ComboBoxItem Content="Skip"/>',
    '                                                <ComboBoxItem Content="Disabled"/>',
    '                                                <ComboBoxItem Content="Manual"/>',
    '                                                <ComboBoxItem Content="Auto"/>',
    '                                                <ComboBoxItem Content="Auto Delayed"/>',
    '                                            </ComboBox>',
    '                                        </DataTemplate>',
    '                                    </DataGridTemplateColumn.CellTemplate>',
    '                                </DataGridTemplateColumn>',
    '                                <DataGridTextColumn Header="DisplayName"',
    '                                                    Width="*"',
    '                                                    Binding="{Binding DisplayName}"/>',
    '                            </DataGrid.Columns>',
    '                        </DataGrid>',
    '                        <DataGrid Grid.Row="3" Name="ServiceExtension" HeadersVisibility="None">',
    '                            <DataGrid.Columns>',
    '                                <DataGridTextColumn Header="Name"',
    '                                                    Width="150"',
    '                                                    Binding="{Binding Name}"/>',
    '                                <DataGridTextColumn Header="Value"',
    '                                                    Width="*"',
    '                                                    Binding="{Binding Value}"/>',
    '                            </DataGrid.Columns>',
    '                        </DataGrid>',
    '                        <Grid Grid.Row="4">',
    '                            <Grid.ColumnDefinitions>',
    '                                <ColumnDefinition Width="*"/>',
    '                                <ColumnDefinition Width="*"/>',
    '                            </Grid.ColumnDefinitions>',
    '                            <Button Grid.Column="0" ',
    '                                    Name="ServiceGet" ',
    '                                    Content="Get"/>',
    '                            <Button Grid.Column="1" ',
    '                                    Name="ServiceSet" ',
    '                                    Content="Apply" ',
    '                                    IsEnabled="False"/>',
    '                        </Grid>',
    '                    </Grid>',
    '                </TabItem>',
    '                <TabItem Header="Preferences">',
    '                    <Grid>',
    '                        <Grid.ColumnDefinitions>',
    '                            <ColumnDefinition Width="*"/>',
    '                            <ColumnDefinition Width="10"/>',
    '                            <ColumnDefinition Width="2*"/>',
    '                        </Grid.ColumnDefinitions>',
    '                        <Grid Grid.Row="0">',
    '                            <Grid.RowDefinitions>',
    '                                <RowDefinition Height="40"/>',
```

```
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                            </Grid.RowDefinitions>',
'                            <Label Grid.Row="0" Content="[Bypass]"/>',
'                            <CheckBox Grid.Row="1" ',
'                                      Name="ServiceBypassBuild"',
'                                      Content="Skip Build/Version Check"/>',
'                            <ComboBox Grid.Row="2"',
'                                      Name="ServiceBypassEdition"',
'                                      VerticalAlignment="Center">',
'                                <ComboBoxItem Content="Override Edition Check" IsSelected="True"/>',
'                                <ComboBoxItem Content="Windows 10 Home"/>',
'                                <ComboBoxItem Content="Windows 10 Pro"/>',
'                            </ComboBox>',
'                            <CheckBox Grid.Row="3" ',
'                                      Name="ServiceBypassLaptop" ',
'                                      Content="Enable Laptop Tweaks"/>',
'                            <Label Grid.Row="4" Content="[Display Services]"/>',
'                            <Grid Grid.Row="5">',
'                                <Grid.ColumnDefinitions>',
'                                    <ColumnDefinition Width="*"/>',
'                                    <ColumnDefinition Width="*"/>',
'                                    <ColumnDefinition Width="*"/>',
'                                </Grid.ColumnDefinitions>',
'                                <CheckBox Grid.Column="0"',
'                                          Name="ServiceDisplayActive"',
'                                          Content="Active"/>',
'                                <CheckBox Grid.Column="1"',
'                                          Name="ServiceDisplayInactive"',
'                                          Content="Inactive"/>',
'                                <CheckBox Grid.Column="2"',
'                                          Name="ServiceDisplaySkipped"',
'                                          Content="Skipped"/>',
'                            </Grid>',
'                            <Label    Grid.Row="6"',
'                                      Content="[Miscellaneous]"',
'                                      Margin="5"/>',
'                            <CheckBox Grid.Row="7"',
'                                      Name="ServiceMiscSimulate"',
'                                      Content="Simulate Changes [Dry Run]"/>',
'                            <CheckBox Grid.Row="8"',
'                                      Name="ServiceMiscXbox"',
'                                      Content="Skip All Xbox Services"/>',
'                            <CheckBox Grid.Row="9"',
'                                      Name="ServiceMiscChange"',
'                                      Content="Allow Change of Service State"/>',
'                            <CheckBox Grid.Row="10"',
'                                      Name="ServiceMiscStopDisabled"',
'                                      Content="Stop Disabled Services"/>',
'                        </Grid>',
'                        <Border Grid.Column="1" Background="Black" BorderThickness="0" Margin="4"/>',
'                        <Grid Grid.Column="2">',
'                            <Grid.RowDefinitions>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
```

```
    '                                <RowDefinition Height="40"/>',
    '                                <RowDefinition Height="40"/>',
    '                                <RowDefinition Height="40"/>',
    '                                <RowDefinition Height="40"/>',
    '                                <RowDefinition Height="40"/>',
    '                            </Grid.RowDefinitions>',
    '                            <Label Grid.Row="0"',
    '                                   Content="[Logging]: Log all changes made by this utility"/>',
    '                            <Grid Grid.Row="1">',
    '                                <Grid.ColumnDefinitions>',
    '                                    <ColumnDefinition Width="80"/>',
    '                                    <ColumnDefinition Width="*"/>',
    '                                    <ColumnDefinition Width="80"/>',
    '                                </Grid.ColumnDefinitions>',
    '                                <CheckBox Grid.Column="0"',
    '                                          Name="ServiceLogServiceSwitch"',
    '                                          Content="Services"',
    '                                          HorizontalContentAlignment="Right"/>',
    '                                <TextBox  Grid.Column="1"',
    '                                          Name="ServiceLogServiceFile"',
    '                                          IsEnabled="False"/>',
    '                                <Button   Grid.Column="2"',
    '                                          Name="ServiceLogServiceBrowse"',
    '                                          Content="Browse"/>',
    '                            </Grid>',
    '                            <Grid Grid.Row="2">',
    '                                <Grid.ColumnDefinitions>',
    '                                    <ColumnDefinition Width="80"/>',
    '                                    <ColumnDefinition Width="*"/>',
    '                                    <ColumnDefinition Width="80"/>',
    '                                </Grid.ColumnDefinitions>',
    '                                <CheckBox Grid.Column="0"',
    '                                          Name="ServiceLogScriptSwitch"',
    '                                          Content="Script"',
    '                                          HorizontalContentAlignment="Right"/>',
    '                                <TextBox  Grid.Column="1"',
    '                                          Name="ServiceLogScriptFile"',
    '                                          IsEnabled="False"/>',
    '                                <Button   Grid.Column="2"',
    '                                          Name="ServiceLogScriptBrowse"',
    '                                          Content="Browse"/>',
    '                            </Grid>',
    '                            <Label Grid.Row="3" ',
    '                                   Content="[Backup]: Save your current service configuration"/>',
    '                            <Grid Grid.Row="4">',
    '                                <Grid.ColumnDefinitions>',
    '                                    <ColumnDefinition Width="80"/>',
    '                                    <ColumnDefinition Width="*"/>',
    '                                    <ColumnDefinition Width="80"/>',
    '                                </Grid.ColumnDefinitions>',
    '                                <CheckBox Grid.Column="0"',
    '                                          Name="ServiceRegSwitch"',
    '                                          Content="*.reg"',
    '                                          HorizontalContentAlignment="Right"/>',
    '                                <TextBox  Grid.Column="1"',
    '                                          Name="ServiceRegFile"',
    '                                          IsEnabled="False"/>',
    '                                <Button   Grid.Column="2"',
    '                                          Name="ServiceRegBrowse"',
    '                                          Content="Browse"/>',
    '                            </Grid>',
    '                            <Grid Grid.Row="5">',
    '                                <Grid.ColumnDefinitions>',
    '                                    <ColumnDefinition Width="80"/>',
    '                                    <ColumnDefinition Width="*"/>',
    '                                    <ColumnDefinition Width="80"/>',
    '                                </Grid.ColumnDefinitions>',
    '                                <CheckBox Grid.Column="0"',
    '                                          Name="ServiceCsvSwitch"',
    '                                          Content="*.csv"',
    '                                          HorizontalContentAlignment="Right"/>',
    '                                <TextBox  Grid.Column="1"',
```

```
'                                        Name="ServiceCsvFile"',
'                                        IsEnabled="False"/>',
'                                <Button   Grid.Column="2"',
'                                        Name="ServiceCsvBrowse"',
'                                        Content="Browse"/>',
'                            </Grid>',
'                            <Label Grid.Row="6"',
'                                   Content="[Development]"/>',
'                            <CheckBox Grid.Row="7"',
'                                      Name="ServiceDevErrors"',
'                                      Content="Diagnostic Output [On Error]"/>',
'                            <CheckBox Grid.Row="8"',
'                                      Name="ServiceDevLog"',
'                                      Content="Enable Development Logging"/>',
'                            <CheckBox Grid.Row="9"',
'                                      Name="ServiceDevConsole"',
'                                      Content="Enable Console"/>',
'                            <CheckBox Grid.Row="10"',
'                                      Name="ServiceDevReport"',
'                                      Content="Enable Diagnostic"/>',
'                        </Grid>',
'                    </Grid>',
'                </TabItem>',
'                <TabItem Header="About">',
'                    <Grid>',
'                        <Grid.RowDefinitions>',
'                            <RowDefinition Height="40"/>',
'                            <RowDefinition Height="40"/>',
'                            <RowDefinition Height="55"/>',
'                            <RowDefinition Height="40"/>',
'                            <RowDefinition Height="55"/>',
'                        </Grid.RowDefinitions>',
'                        <Label Grid.Row="0" Content="[About]: Original Authors"/>',
'                        <Grid Grid.Row="1">',
'                            <Grid.ColumnDefinitions>',
'                                <ColumnDefinition Width="120"/>',
'                                <ColumnDefinition Width="*"/>',
'                            </Grid.ColumnDefinitions>',
'                            <Label Grid.Column="0" Content="[BlackViper]:"/>',
'                            <TextBox Grid.Column="1" Text="https://www.blackviper.com"/>',
'                        </Grid>',
'                        <TextBox Grid.Row="2" ',
'                                 Height="45" ',
'                                 Padding="5" ',
'                                 VerticalContentAlignment="Top" >',
'                            BlackViper is the original author of the Black Viper ',
'                                 Service Configuration featured on his website. The ',
'                                 original utility dealt with (*.bat) files to provide ',
'                                 a service configuration template for Windows services, ',
'                                 dating back to the days of Windows (2000/XP).',
'                        </TextBox>',
'                        <Grid Grid.Row="3">',
'                            <Grid.ColumnDefinitions>',
'                                <ColumnDefinition Width="120"/>',
'                                <ColumnDefinition Width="*"/>',
'                            </Grid.ColumnDefinitions>',
'                            <Label Grid.Column="0" Content="[MadBomb122]:"/>',
'                            <TextBox Grid.Column="1" Text="https://www.github.com/MadBomb122"/>',
'                        </Grid>',
'                        <TextBox Grid.Row="4"',
'                                 Height="45"',
'                                 Padding="5"',
'                                 VerticalContentAlignment="Top">',
'                            MadBomb122 is the author of the Windows PowerShell ',
'                                 (GUI/graphical user interface) tool that adopted Black ',
'                                 Viper&amp;apos;s service configuration (*.bat) files in a prior ',
'                                 version of this utility, which is featured on his [GitHub] ',
'                                 repository above.',
'                        </TextBox>',
'                    </Grid>',
'                </TabItem>',
'            </TabControl>',
```

```
'            </TabItem>',
'            <TabItem Header="Control">',
'                <TabControl>',
'                    <TabItem Header="Preferences">',
'                        <Grid>',
'                            <Grid.RowDefinitions>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                                <RowDefinition Height="40"/>',
'                            </Grid.RowDefinitions>',
'                            <Label Grid.Row="0" Content="[Global]:"/>',
'                            <Grid Grid.Row="1">',
'                                <Grid.ColumnDefinitions>',
'                                    <ColumnDefinition Width="150"/>',
'                                    <ColumnDefinition Width="10"/>',
'                                    <ColumnDefinition Width="150"/>',
'                                    <ColumnDefinition Width="250"/>',
'                                </Grid.ColumnDefinitions>',
'                                <CheckBox Grid.Column="0"',
'                                      Name="ControlGlobalRestorePoint"',
'                                      Content="Create Restore Point"/>',
'                                <Border Grid.Column="1" Margin="4" Background="Black"/>',
'                                <CheckBox Grid.Column="2"',
'                                      Name="ControlGlobalRestart"',
'                                      Content="Restart When Done"/>',
'                                <Label Grid.Column="3" Content="Restart recommended"/>',
'                            </Grid>',
'                            <Grid Grid.Row="2">',
'                                <Grid.ColumnDefinitions>',
'                                    <ColumnDefinition Width="150"/>',
'                                    <ColumnDefinition Width="10"/>',
'                                    <ColumnDefinition Width="150"/>',
'                                    <ColumnDefinition Width="250"/>',
'                                </Grid.ColumnDefinitions>',
'                                <CheckBox Grid.Column="0"',
'                                          Name="ControlGlobalShowSkipped"',
'                                          Content="Show Skipped Items"/>',
'                                <Border Grid.Column="1" Margin="4" Background="Black"/>',
'                                <CheckBox Grid.Column="2"',
'                                          Name="ControlGlobalVersionCheck"',
'                                          Content="Check for Update"/>',
'                                <Label Grid.Column="3" ',
'                                       Content="If found, will run with current settings"/>',
'                            </Grid>',
'                            <CheckBox Grid.Row="3"',
'                                      Name="ControlGlobalInternetCheck"',
'                                      Content="Skip Internet Check"/>',
'                            <Label Grid.Row="4" Content="[Backup]:" Margin="5"/>',
'                            <Grid Grid.Row="5">',
'                                <Grid.ColumnDefinitions>',
'                                    <ColumnDefinition Width="*"/>',
'                                    <ColumnDefinition Width="*"/>',
'                                    <ColumnDefinition Width="*"/>',
'                                    <ColumnDefinition Width="*"/>',
'                                </Grid.ColumnDefinitions>',
'                                <Button Grid.Column="0"',
'                                        Name="ControlBackupSave"',
'                                        Content="Save Settings"/>',
'                                <Button Grid.Column="1"',
'                                        Name="ControlBackupLoad"',
'                                        Content="Load Settings"/>',
'                                <Button Grid.Column="2"',
'                                        Name="ControlBackupWinDefault"',
'                                        Content="Windows Default"/>',
'                                <Button Grid.Column="3"',
'                                        Name="ControlBackupResetDefault"',
'                                        Content="Reset All Items"/>',
```

```
    '                        </Grid>',
    '                        <Label Grid.Row="6" Content="[Script]"/>',
    '                        <ComboBox Grid.Row="7" Margin="5" Height="24" IsEnabled="False">',
    '                            <ComboBoxItem Content="Rewrite Module Version" IsSelected="True"/>',
    '                        </ComboBox>',
    '                    </Grid>',
    '                </TabItem>',
    '                <TabItem Header="Settings">',
    '                    <Grid>',
    '                        <Grid.RowDefinitions>',
    '                            <RowDefinition Height="40"/>',
    '                            <RowDefinition Height="40"/>',
    '                            <RowDefinition Height="*"/>',
    '                            <RowDefinition Height="100"/>',
    '                            <RowDefinition Height="40"/>',
    '                        </Grid.RowDefinitions>',
    '                        <Label Content="[System Control Settings]:"/>',
    '                        <Grid Grid.Row="1">',
    '                            <Grid.ColumnDefinitions>',
    '                                <ColumnDefinition Width="60"/>',
    '                                <ColumnDefinition Width="130"/>',
    '                                <ColumnDefinition Width="10"/>',
    '                                <ColumnDefinition Width="100"/>',
    '                                <ColumnDefinition Width="*"/>',
    '                            </Grid.ColumnDefinitions>',
    '                            <Label Grid.Column="0" Content="[Slot]:"/>',
    '                            <ComboBox Grid.Column="1" Name="ControlSlot" SelectedIndex="0">',
    '                                <ComboBoxItem Content="All"/>',
    '                                <ComboBoxItem Content="Privacy"/>',
    '                                <ComboBoxItem Content="WindowsUpdate"/>',
    '                                <ComboBoxItem Content="Service"/>',
    '                                <ComboBoxItem Content="Context"/>',
    '                                <ComboBoxItem Content="Taskbar"/>',
    '                                <ComboBoxItem Content="StartMenu"/>',
    '                                <ComboBoxItem Content="Explorer"/>',
    '                                <ComboBoxItem Content="ThisPC"/>',
    '                                <ComboBoxItem Content="Desktop"/>',
    '                                <ComboBoxItem Content="LockScreen"/>',
    '                                <ComboBoxItem Content="Miscellaneous"/>',
    '                                <ComboBoxItem Content="PhotoViewer"/>',
    '                                <ComboBoxItem Content="WindowsApps"/>',
    '                            </ComboBox>',
    '                            <Border Grid.Column="2" Margin="4" Background="Black"/>',
    '                            <ComboBox Grid.Column="3" Name="ControlProperty" SelectedIndex="0">',
    '                                <ComboBoxItem Content="Name"/>',
    '                                <ComboBoxItem Content="Description"/>',
    '                            </ComboBox>',
    '                            <TextBox Grid.Column="4" Name="ControlFilter"/>',
    '                        </Grid>',
    '                        <DataGrid Grid.Row="2" Name="ControlOutput">',
    '                            <DataGrid.Columns>',
    '                                <DataGridTextColumn Header="Name"',
    '                                                    Width="200"',
    '                                                    Binding="{Binding DisplayName}"',
    '                                                    IsReadOnly="True"/>',
    '                                <DataGridTemplateColumn Header="Value" Width="150">',
    '                                    <DataGridTemplateColumn.CellTemplate>',
    '                                        <DataTemplate>',
    '                                            <ComboBox SelectedIndex="{Binding Value}" ',
    '                                                      ItemsSource="{Binding Options}" ',
    '                                                      Style="{StaticResource DGCombo}"/>',
    '                                        </DataTemplate>',
    '                                    </DataGridTemplateColumn.CellTemplate>',
    '                                </DataGridTemplateColumn>',
    '                                <DataGridTextColumn Header="Description"',
    '                                                    Width="*"',
    '                                                    Binding="{Binding Description}"',
    '                                                    IsReadOnly="True"/>',
    '                            </DataGrid.Columns>',
    '                        </DataGrid>',
    '                        <DataGrid Grid.Row="3" ',
    '                                  Name="ControlOutputExtension"',
```

```
'                                  HeadersVisibility="None"',
'                                  RowHeaderWidth="0">',
'                              <DataGrid.Columns>',
'                                  <DataGridTextColumn Header="Name"',
'                                                      Binding="{Binding Name}"',
'                                                      Width="150"/>',
'                                  <DataGridTextColumn Header="Value"',
'                                                      Binding="{Binding Value}"',
'                                                      Width="*"/>',
'                              </DataGrid.Columns>',
'                          </DataGrid>',
'                          <Grid Grid.Row="4">',
'                              <Grid.ColumnDefinitions>',
'                                  <ColumnDefinition Width="*"/>',
'                                  <ColumnDefinition Width="*"/>',
'                              </Grid.ColumnDefinitions>',
'                              <Button Grid.Column="0"',
'                                      Name="ControlOutputApply"',
'                                      Content="Apply"',
'                                      IsEnabled="False"/>',
'                              <Button Grid.Column="1"',
'                                      Name="ControlOutputDontApply"',
'                                      Content="Do not apply..."',
'                                      IsEnabled="False"/>',
'                          </Grid>',
'                      </Grid>',
'                  </TabItem>',
'                  <TabItem Header="Features">',
'                      <Grid>',
'                          <Grid.RowDefinitions>',
'                              <RowDefinition Height="40"/>',
'                              <RowDefinition Height="40"/>',
'                              <RowDefinition Height="*"/>',
'                              <RowDefinition Height="100"/>',
'                              <RowDefinition Height="40"/>',
'                          </Grid.RowDefinitions>',
'                          <Label Content="[Windows Features]:"/>',
'                          <Grid Grid.Row="1">',
'                              <Grid.ColumnDefinitions>',
'                                  <ColumnDefinition Width="120"/>',
'                                  <ColumnDefinition Width="*"/>',
'                              </Grid.ColumnDefinitions>',
'                              <ComboBox Grid.Column="0"',
'                                        Name="ControlFeatureProperty"',
'                                        SelectedIndex="0">',
'                                  <ComboBoxItem Content="FeatureName"/>',
'                                  <ComboBoxItem Content="State"/>',
'                              </ComboBox>',
'                              <TextBox Grid.Column="1"',
'                                       Name="ControlFeatureFilter"/>',
'                          </Grid>',
'                          <DataGrid Name="ControlFeature" Grid.Row="2">',
'                              <DataGrid.Columns>',
'                                  <DataGridTextColumn Header="Name"',
'                                                      Width="250"',
'                                                      Binding="{Binding FeatureName}"',
'                                                      IsReadOnly="True"/>',
'                                  <DataGridTemplateColumn Header="State" Width="150">',
'                                      <DataGridTemplateColumn.CellTemplate>',
'                                          <DataTemplate>',
'                                              <ComboBox SelectedIndex="{Binding State.Index}"',
'                                                        Style="{StaticResource DGCombo}">',
'                                                  <ComboBoxItem Content="Disabled"/>',
'                                                  <ComboBoxItem Content="DisabledWithPayloadRemoved"/>',
'                                                  <ComboBoxItem Content="Enabled"/>',
'                                              </ComboBox>',
'                                          </DataTemplate>',
'                                      </DataGridTemplateColumn.CellTemplate>',
'                                  </DataGridTemplateColumn>',
'                                  <DataGridTextColumn Header="Description"',
'                                                      Width="*"',
'                                                      Binding="{Binding State.Description}"',
```

```
'                                                   IsReadOnly="True"/>',
'                            </DataGrid.Columns>',
'                        </DataGrid>',
'                        <DataGrid Grid.Row="3"',
'                                  Name="ControlFeatureExtension"',
'                                  HeadersVisibility="None"',
'                                  RowHeaderWidth="0">',
'                            <DataGrid.Columns>',
'                                <DataGridTextColumn Header="Name"',
'                                                    Binding="{Binding Name}"',
'                                                    Width="150"/>',
'                                <DataGridTextColumn Header="Value"',
'                                                    Binding="{Binding Value}"',
'                                                    Width="*"/>',
'                            </DataGrid.Columns>',
'                        </DataGrid>',
'                        <Grid Grid.Row="4">',
'                            <Grid.ColumnDefinitions>',
'                                <ColumnDefinition Width="*"/>',
'                                <ColumnDefinition Width="*"/>',
'                            </Grid.ColumnDefinitions>',
'                            <Button Grid.Column="0"',
'                                    Name="ControlFeatureApply"',
'                                    Content="Apply"',
'                                    IsEnabled="False"/>',
'                            <Button Grid.Column="1"',
'                                    Name="ControlFeatureDontApply"',
'                                    Content="Do not apply..."',
'                                    IsEnabled="False"/>',
'                        </Grid>',
'                    </Grid>',
'                </TabItem>',
'                <TabItem Header="AppX">',
'                    <Grid>',
'                        <Grid.RowDefinitions>',
'                            <RowDefinition Height="40"/>',
'                            <RowDefinition Height="40"/>',
'                            <RowDefinition Height="*"/>',
'                            <RowDefinition Height="100"/>',
'                            <RowDefinition Height="40"/>',
'                        </Grid.RowDefinitions>',
'                        <Label Grid.Row="0" Content="[AppX Catalog]"/>',
'                        <Grid Grid.Row="1">',
'                            <Grid.ColumnDefinitions>',
'                                <ColumnDefinition Width="120"/>',
'                                <ColumnDefinition Width="*"/>',
'                            </Grid.ColumnDefinitions>',
'                            <ComboBox Grid.Column="0"',
'                                      Name="ControlAppXProperty"',
'                                      SelectedIndex="0">',
'                                <ComboBoxItem Content="PackageName"/>',
'                                <ComboBoxItem Content="DisplayName"/>',
'                                <ComboBoxItem Content="PublisherID"/>',
'                                <ComboBoxItem Content="InstallLocation"/>',
'                            </ComboBox>',
'                            <TextBox Grid.Column="1" Name="ControlAppXFilter"/>',
'                        </Grid>',
'                        <DataGrid Name="ControlAppX" Grid.Row="2">',
'                            <DataGrid.Columns>',
'                                <DataGridTextColumn Header="#"',
'                                                    Binding="{Binding Index}"',
'                                                    Width="25"/>',
'                                <DataGridTemplateColumn Header="Profile" Width="45">',
'                                    <DataGridTemplateColumn.CellTemplate>',
'                                        <DataTemplate>',
'                                            <ComboBox SelectedIndex="{Binding Profile}"',
'                                                      Margin="0"',
'                                                      Padding="2"',
'                                                      Height="18"',
'                                                      FontSize="10"',
'                                                      VerticalContentAlignment="Center">',
'                                                <ComboBoxItem Content="False"/>',
```

```
    '                                            <ComboBoxItem Content="True"/>',
    '                                        </ComboBox>',
    '                                    </DataTemplate>',
    '                                </DataGridTemplateColumn.CellTemplate>',
    '                            </DataGridTemplateColumn>',
    '                            <DataGridTextColumn Header="DisplayName"',
    '                                                Binding="{Binding DisplayName}"',
    '                                                Width="2*"/>',
    '                            <DataGridTextColumn Header="PublisherID"',
    '                                                Binding="{Binding PublisherID}"',
    '                                                Width="*"/>',
    '                            <DataGridTextColumn Header="Version"',
    '                                                Binding="{Binding Version}"',
    '                                                Width="100"/>',
    '                            <DataGridTemplateColumn Header="Slot" Width="60">',
    '                                <DataGridTemplateColumn.CellTemplate>',
    '                                    <DataTemplate>',
    '                                        <ComboBox SelectedIndex="{Binding Slot}"',
    '                                                  Margin="0"',
    '                                                  Padding="2"',
    '                                                  Height="18"',
    '                                                  FontSize="10"',
    '                                                  VerticalContentAlignment="Center">',
    '                                            <ComboBoxItem Content="Skip"/>',
    '                                            <ComboBoxItem Content="Unhide"/>',
    '                                            <ComboBoxItem Content="Hide"/>',
    '                                            <ComboBoxItem Content="Uninstall"/>',
    '                                        </ComboBox>',
    '                                    </DataTemplate>',
    '                                </DataGridTemplateColumn.CellTemplate>',
    '                            </DataGridTemplateColumn>',
    '                        </DataGrid.Columns>',
    '                    </DataGrid>',
    '                    <DataGrid Grid.Row="3"',
    '                              Name="ControlAppXExtension"',
    '                              HeadersVisibility="None"',
    '                              RowHeaderWidth="0">',
    '                        <DataGrid.Columns>',
    '                            <DataGridTextColumn Header="Name"',
    '                                                Binding="{Binding Name}"',
    '                                                Width="150"/>',
    '                            <DataGridTextColumn Header="Value"',
    '                                                Binding="{Binding Value}"',
    '                                                Width="*"/>',
    '                        </DataGrid.Columns>',
    '                    </DataGrid>',
    '                    <Grid Grid.Row="4">',
    '                        <Grid.ColumnDefinitions>',
    '                            <ColumnDefinition Width="*"/>',
    '                            <ColumnDefinition Width="*"/>',
    '                        </Grid.ColumnDefinitions>',
    '                        <Button Grid.Column="0"',
    '                                Name="ControlAppXApply"',
    '                                Content="Apply"',
    '                                IsEnabled="False"/>',
    '                        <Button Grid.Column="1"',
    '                                Name="ControlAppXDontApply"',
    '                                Content="Do not apply..."',
    '                                IsEnabled="False"/>',
    '                    </Grid>',
    '                </Grid>',
    '            </TabItem>',
    '        </TabControl>',
    '    </Grid>',
    '</Window>' -join "`n")
}
```

_____/ Class [ViperBombXaml]

Class [XamlProperty] /----------\

This is meant to capture each individual property that [PowerShell] can control in the (Xaml/GUI).

```
Class XamlProperty
{
    [UInt32] $Index
    [String] $Name
    [Object] $Type
    [Object] $Control
    XamlProperty([UInt32]$Index,[String]$Name,[Object]$Object)
    {
        $This.Index   = $Index
        $This.Name    = $Name
        $This.Type    = $Object.GetType().Name
        $This.Control = $Object
    }
    [String] ToString()
    {
        Return $This.Name
    }
}
```

\----------/ Class [XamlProperty]

Class [XamlWindow] /----------\

This is the class that converts the chunk of Xaml into the GUI object for [Windows Presentation Framework].

```
Class XamlWindow
{
    Hidden [Object]        $XAML
    Hidden [Object]         $XML
    [String[]]            $Names
    [Object]              $Types
    [Object]               $Node
    [Object]                 $IO
    [String]          $Exception
    XamlWindow([String]$Xaml)
    {
        If (!$Xaml)
        {
            Throw "Invalid XAML Input"
        }

        [System.Reflection.Assembly]::LoadWithPartialName('presentationframework')

        $This.Xaml              = $Xaml
        $This.Xml               = [XML]$Xaml
        $This.Names             = $This.FindNames()
        $This.Types             = @( )
        $This.Node              = [System.Xml.XmlNodeReader]::New($This.Xml)
        $This.IO                = [System.Windows.Markup.XamlReader]::Load($This.Node)

        ForEach ($X in 0..($This.Names.Count-1))
        {
            $Name               = $This.Names[$X]
            $Object             = $This.IO.FindName($Name)
            $This.IO            | Add-Member -MemberType NoteProperty -Name $Name -Value $Object -Force
            If (!!$Object)
            {
                $This.Types     += $This.XamlProperty($This.Types.Count,$Name,$Object)
            }
        }
    }
    [String[]] FindNames()
    {
        Return [Regex]::Matches($This.Xaml,"( Name\=\`"\w+`")").Value -Replace "( Name=|`")",""
    }
    [Object] XamlProperty([UInt32]$Index,[String]$Name,[Object]$Object)
```

```
    {
        Return [XamlProperty]::New($Index,$Name,$Object)
    }
    Invoke()
    {
        Try
        {
            $This.IO.Dispatcher.InvokeAsync({ $This.IO.ShowDialog() }).Wait()
        }
        Catch
        {
            $This.Exception = $PSItem
        }
    }
}
```

_____/ Class [XamlWindow] /_____\
/ Class [HotFixItem] /

Individual object returned from the command `Get-HotFix`, or even a portion of `Get-ComputerInfo`

```
Class HotFixItem
{
    [String] $Source
    [String] $Description
    [String] $HotFixID
    [String] $InstalledBy
    [String] $InstalledOn
    HotFixItem([Object]$HotFix)
    {
        $This.Source      = $HotFix.PSComputerName
        $This.Description = $HotFix.Description
        $This.HotFixID    = $HotFix.HotFixID
        $This.InstalledBy = $HotFix.InstalledBy
        $This.InstalledOn = ([DateTime]$HotFix.InstalledOn).ToString("MM/dd/yyyy")
    }
}
```

_____/ Class [HotFixItem] /_____\
/ Class [HotFixList] /

List of objects meant to comparmentalize each individual object returned from the command `Get-HotFix`

```
Class HotFixList
{
    [Object] $Output
    HotFixList()
    {
        $This.Output = @( )
        ForEach ($HotFix in Get-HotFix)
        {
            $This.Output += [HotFixItem]::New($HotFix)
        }
    }
}
```

_____/ Class [HotFixList] /_____\
/ Enum [ServiceStartModeType] /

Meant to enumerate the `[Object] $StartMode` values returned from the command `Get-Service`

```
Enum ServiceStartModeType
{
    Skip
```

```
    Disabled
    Manual
    Auto
    AutoDelayed
}
```

_____/ Enum [ServiceStartModeType]
Class [ServiceStartModeSlot] /_____\
/_____

This object is meant to index the values of the enumeration type above, as well as to provide a description, and be usable from the Xaml ComboBox. Effectively, this allows the default values returned from `Get-Service` to be able to BECOME an `[Object]`. THEN, you're REALLY gettin' some stuff done...

```
Class ServiceStartModeSlot
{
    [UInt32] $Index
    [String] $Type
    [String] $Description
    ServiceStartModeSlot([String]$Type)
    {
        $This.Type        = [ServiceStartModeType]::$Type
        $This.Index       = [UInt32][ServiceStartModeType]::$Type
    }
    [String] ToString()
    {
        Return $This.Type
    }
}
```

_____/ Class [ServiceStartModeSlot]
Class [ServiceStartModeList] /_____\
/_____

This is specifically meant to contain ALL of the (enumeration types/slots) specified above, in relation to the `[Object] $StartMode` values returned from the command `Get-Service`.

```
Class ServiceStartModeList
{
    [Object] $Output
    ServiceStartModeList()
    {
        $This.Output = @( )
        [System.Enum]::GetNames([ServiceStartModeType]) | % { $This.Add($_) }
    }
    Add([String]$Name)
    {
        $Item             = [ServiceStartModeSlot]::New($Name)
        $Item.Description = Switch ($Name)
        {
            Skip        { "The service is skipped"                          }
            Disabled    { "The service is totally disabled"                 }
            Manual      { "The service requires a manual start"             }
            Auto        { "The service automatically starts"                }
            AutoDelayed { "The service automatically starts, but is delayed" }
        }
        $This.Output += $Item
    }
    [Object] Get([String]$Type)
    {
        Return $This.Output[[UInt32][ServiceStartModeType]::$Type]
    }
}
```

_____/ Class [ServiceStartModeList]

Enum [ServiceStateType]

Same as the above Enum type, but specifically meant for the [Object] $State property.

```
Enum ServiceStateType
{
    Running
    Stopped
}
```

Class [ServiceStateSlot]

Same idea as the above slot object.

```
Class ServiceStateSlot
{
    [UInt32] $Index
    [String] $Type
    [String] $Description
    ServiceStateSlot([String]$Type)
    {
        $This.Type        = [ServiceStateType]::$Type
        $This.Index       = [UInt32][ServiceStateType]::$Type
    }
    [String] ToString()
    {
        Return $This.Type
    }
}
```

Class [ServiceStateList]

Same idea as the above list object.

```
Class ServiceStateList
{
    [Object] $Output
    ServiceStateList()
    {
        $This.Output = @( )
        [System.Enum]::GetNames([ServiceStateType]) | % { $This.Add($_) }
    }
    Add([String]$Name)
    {
        $Item             = [ServiceStateSlot]::New($Name)
        $Item.Description = Switch ($Name)
        {
            Running  { "The service is currently running"     }
            Stopped  { "The service is NOT currently running" }
        }
        $This.Output += $Item
    }
    [Object] Get([String]$Type)
    {
        Return $This.Output[[UInt32][ServiceStateType]::$Type]
    }
}
```

Class [ServiceSubcontroller]

The idea for this particular object, is to be able to process each and every individual object returned from the command `Get-Service`. In this instance, the value returned from the command is converted into an Enum type, and then the Enum type is converted into an Integer, and then, the ComboBox for both StartMode and State can be scaled by using the property SelectedIndex.

Almost as if someone who was extremely skilled at using XAML/WPF came along, and showed me how to do it.
And then I went ahead and decided to implement it all within [PowerShell].

```
Class ServiceSubcontroller
{
    [Object] $StartMode
    [Object] $State
    ServiceSubcontroller()
    {
        $This.StartMode = [ServiceStartModeList]::New()
        $This.State     = [ServiceStateList]::New()
    }
    Load([Object]$Service)
    {
        $Service.StartMode   = $This.StartMode.Get($Service.Wmi.StartMode)
        $Service.State       = $This.State.Get($Service.Wmi.State)
    }
}
```

Class [ServiceProfile]

This is a profile object meant specifically for being able to control the service configuration made by: Charles "Black Viper" Sparks. In this instance, the service NAME will be a property that is searchable.

If it finds the name in the collection, then it will apply the provided slots from the orinal (*.csv).

If it does NOT find the name in the collection, then what it will do, is select the current value for that service, and then apply it to the remaining profiles, effectively nullifying whatever this particular configuration is not scoped for.

```
Class ServiceProfile
{
    [String]      $Name
    [UInt32[]] $Profile
    ServiceProfile([String]$Name,[String]$Value)
    {
        $This.Name    = $Name
        $This.Profile = $Value -Split ","
    }
    ServiceProfile([Switch]$Flag,[String]$Name,[UInt32]$Value)
    {
        $This.Name    = $Name
        $This.Profile = @($Value)*10
    }
}
```

Class [ServiceTemplate]

So, this particular class is what is seen in the GUI, as well as the [Windows Management Instrumentation] object returned by the command `Get-WMIObject -Class Win32_Service`. It is actually an extension of the default object, as it is able to capture the WMI object, and then return various desired properties for the DataGrid object.

```
Class ServiceTemplate
{
    [UInt32]              $Index
    Hidden [Object]         $Wmi
    [String]               $Name
```

```
    [UInt32]              $Scope
    [UInt32[]]          $Profile
    [Object]          $StartMode
    [Object]             $Target
    [Object]              $State
    [UInt32]   $DelayedAutoStart
    [String]              $Status
    [String]        $DisplayName
    [String]           $PathName
    [String]        $Description
    Hidden [UInt32]       $Match
    ServiceTemplate([Int32]$Index,[Object]$Wmi)
    {
        $This.Index              = $Index
        $This.Wmi                = $Wmi
        $This.Name               = $Wmi.Name
        $This.DelayedAutoStart   = $Wmi.DelayedAutoStart
        $This.Status             = $Wmi.Status
        $This.DisplayName        = $Wmi.DisplayName
        $This.PathName           = $Wmi.PathName
        $This.Description        = $Wmi.Description
    }
}
```

_____/
_____/ Class [ServiceTemplate]
Class [ServiceControl] /_____\\
/_____

This particular class is meant to orchestrate the above classes in relation to services, not the XAML components. Effectively, the [Object]$Output property will forward itself to the GUI/DataGrid via [DataBinding].

```
Class ServiceControl
{
    [UInt32]      $Slot
    [Object]       $Sub
    [Object]    $Config
    [Object]    $Output
    ServiceControl()
    {
        $This.Sub                = $This.GetServiceSubcontroller()
        $This.Config             = $This.GetServiceConfig()
        $This.Output             = @( )

        ForEach ($Object in Get-WMIObject -Class Win32_Service | Sort-Object Name)
        {
            $Item                = $This.GetServiceTemplate($This.Output.Count,$Object)
            $Item.Scope          = $Item.Name -in $This.Config.Name
            $This.Load($Item)
            $Item.Profile        = Switch ($Item.Scope)
            {
                0
                {
                    @($Item.StartMode.Index) * 10
                }
                1
                {
                    $This.Config | ? Name -eq $Item.Name | % Profile
                }
            }
            $Item.Target         = $Item.StartMode

            $This.Output        += $Item
        }
    }
    [String[]] ConfigNames()
    {
        $Out = "AJRouter;ALG;AppHostSvc;AppIDSvc;Appinfo;AppMgmt;AppReadiness;AppVClient;aspnet"+
        "_state;AssignedAccessManagerSvc;AudioEndpointBuilder;AudioSrv;AxInstSV;BcastDVRUserSer"+
        "vice_{0};BDESVC;BFE;BITS;BluetoothUserService_{0};Browser;BTAGService;BthAvctpSvc;BthH"+
        "FSrv;bthserv;c2wts;camsvc;CaptureService_{0};CDPSvc;CDPUserSvc_{0};CertPropSvc;COMSysA"+
        "pp;CryptSvc;CscService;defragsvc;DeviceAssociationService;DeviceInstall;DevicePickerUs"+
```

```
        "erSvc_{0};DevQueryBroker;Dhcp;diagnosticshub.standardcollector.service;diagsvc;DiagTra"+
        "ck;DmEnrollmentSvc;dmwappushsvc;Dnscache;DoSvc;dot3svc;DPS;DsmSVC;DsRoleSvc;DsSvc;Dusm"+
        "Svc;EapHost;EFS;embeddedmode;EventLog;EventSystem;Fax;fdPHost;FDResPub;fhsvc;FontCache"+
        ";FontCache3.0.0.0;FrameServer;ftpsvc;GraphicsPerfSvc;hidserv;hns;HomeGroupListener;Hom"+
        "eGroupProvider;HvHost;icssvc;IKEEXT;InstallService;iphlpsvc;IpxlatCfgSvc;irmon;KeyIso;"+
        "KtmRm;LanmanServer;LanmanWorkstation;lfsvc;LicenseManager;lltdsvc;lmhosts;LPDSVC;LxssM"+
        "anager;MapsBroker;MessagingService_{0};MSDTC;MSiSCSI;MsKeyboardFilter;MSMQ;MSMQTrigger"+
        "s;NaturalAuthentication;NcaSVC;NcbService;NcdAutoSetup;Netlogon;Netman;NetMsmqActivato"+
        "r;NetPipeActivator;netprofm;NetSetupSvc;NetTcpActivator;NetTcpPortSharing;NlaSvc;nsi;O"+
        "neSyncSvc_{0};p2pimsvc;p2psvc;PcaSvc;PeerDistSvc;PerfHost;PhoneSvc;pla;PlugPlay;PNRPAu"+
        "toReg;PNRPsvc;PolicyAgent;Power;PrintNotify;PrintWorkflowUserSvc_{0};ProfSvc;PushToIns"+
        "tall;QWAVE;RasAuto;RasMan;RemoteAccess;RemoteRegistry;RetailDemo;RmSvc;RpcLocator;SamS"+
        "s;SCardSvr;ScDeviceEnum;SCPolicySvc;SDRSVC;seclogon;SEMgrSvc;SENS;Sense;SensorDataServ"+
        "ice;SensorService;SensrSvc;SessionEnv;SgrmBroker;SharedAccess;SharedRealitySvc;ShellHW"+
        "Detection;shpamsvc;smphost;SmsRouter;SNMPTRAP;spectrum;Spooler;SSDPSRV;ssh-agent;SstpS"+
        "vc;StiSvc;StorSvc;svsvc;swprv;SysMain;TabletInputService;TapiSrv;TermService;Themes;Ti"+
        "eringEngineService;TimeBroker;TokenBroker;TrkWks;TrustedInstaller;tzautoupdate;UevAgen"+
        "tService;UI0Detect;UmRdpService;upnphost;UserManager;UsoSvc;VaultSvc;vds;vmcompute;vmi"+
        "cguestinterface;vmicheartbeat;vmickvpexchange;vmicrdv;vmicshutdown;vmictimesync;vmicvm"+
        "session;vmicvss;vmms;VSS;W32Time;W3LOGSVC;W3SVC;WaaSMedicSvc;WalletService;WarpJITSvc;"+
        "WAS;wbengine;WbioSrvc;Wcmsvc;wcncsvc;WdiServiceHost;WdiSystemHost;WebClient;Wecsvc;WEP"+
        "HOSTSVC;wercplsupport;WerSvc;WFDSConSvc;WiaRpc;WinHttpAutoProxySvc;Winmgmt;WinRM;wisvc"+
        ";WlanSvc;wlidsvc;wlpasvc;wmiApSrv;WMPNetworkSvc;WMSVC;workfolderssvc;WpcMonSvc;WPDBusE"+
        "num;WpnService;WpnUserService_{0};wscsvc;WSearch;wuauserv;wudfsvc;WwanSvc;xbgm;XblAuth"+
        "Manager;XblGameSave;XboxGipSvc;XboxNetApiSvc"

        Return $Out -f (Get-Service | ? ServiceType -eq 224)[0].Name.Split('_')[-1] -Split ";"
    }
    [UInt32[]] ConfigMasks()
    {
        Return "0;1;2;3;3;4;3;5;3;6;2;2;3;3;3;2;7;3;3;0;0;0;3;3;4;7;2;0;3;2;8;3;3;3;3;2;3;3"+
        ";2;3;1;2;7;3;2;3;3;3;2;3;3;3;2;2;1;3;3;3;2;3;1;2;3;3;6;3;3;1;1;3;3;9;0;1;3;3;2;2;1;3;3"+
        ";3;2;3;1;0;3;3;1;11;2;2;0;3;3;0;0;3;2;2;3;3;2;1;2;2;7;3;3;2;8;3;1;3;3;3;3;2;3;3;2;3;"+
        "3;3;3;12;12;1;3;1;2;12;1;1;3;3;1;2;6;13;13;13;0;7;1;3;2;12;3;1;1;3;2;3;3;3;3;3;3;3;2;1"+
        "3;3;0;2;3;3;3;2;3;12;5;3;0;3;2;3;3;3;6;1;1;1;1;1;1;1;14;3;3;3;2;3;3;3;3;3;3;2;0;3;3;"+
        "0;3;3;3;3;13;3;3;2;1;1;15;3;3;3;1;3;1;1;3;2;2;7;7;3;3;1;3;1;1;3;1" -Split ";"
    }
    [String[]] ConfigValues()
    {
        Return "2,2,2,2,2,2,1,1,2,2;2,2,2,2,1,1,1,1,1;3,0,3,0,3,0,3,0,3,0;2,0,2,0,2,0,2,0,2,0"+
        ";0,0,2,2,2,2,1,1,2,2;0,0,1,0,1,0,1,0,1,0;0,0,2,0,2,0,2,0,2,0;4,0,4,0,4,0,4,0,4,0;0,0,2"+
        ",2,1,1,1,1,1,1;3,3,3,3,3,3,1,1,3,3;4,4,4,4,1,1,1,1,1,1;0,0,0,0,0,0,0,0,0,0;1,0,1,0,1,0"+
        ",1,0,1,0;2,2,2,2,1,1,1,1,2,2;0,0,3,0,3,0,3,0,3,0;3,3,3,3,2,2,2,2,3,3" -Split ";"
    }
    SetSlot([UInt32]$Slot)
    {
        ForEach ($X in 0..($This.Output.Count-1))
        {
            $Item           = $This.Output[$X]
            $Item.Target    = $This.Sub.StartMode.Output[$Item.Profile[$Slot]]
        }
    }
    [Object] GetServiceTemplate([UInt32]$Index,[Object]$Object)
    {
        Return [ServiceTemplate]::New($Index,$Object)
    }
    [Object] GetServiceSubcontroller()
    {
        Return [ServiceSubcontroller]::New()
    }
    [Object] GetServiceProfile([String]$Name,[String]$Values)
    {
        Return [ServiceProfile]::New($Name,$Values)
    }
    [Object] GetServiceConfig()
    {
        $Hash                       = @{ }
        $Names                      = $This.ConfigNames()
        $Masks                      = $This.ConfigMasks()
        $Values                     = $This.ConfigValues()

        ForEach ($X in 0..($Names.Count-1))
```

```
        {
            $Hash.Add($Hash.Count,$This.GetServiceProfile($Names[$X],$Values[$Masks[$X]]))
        }

        Return @($Hash[0..($Hash.Count-1)])
    }
    Load([Object]$Service)
    {
        $This.Sub.Load($Service)
    }
}
```

Class [Registry] / Class [ServiceControl]

This is specifically meant for the following class, to allow for management of registry paths and values, in a similar manner to how [Group Policy Object Editor] works.

Note: [Group Policy Object Editor] is accessed by gpedit.msc, which I learned how to use way back in 2001 when I attended Capital Region Career and Technical School under instructor David Patzarian, when he taught my class the [Microsoft System Administration] track... I would be willing to put money in it, that the mod that keeps banning me from the [PowerShell] subreddit on Reddit.com, isn't as experienced as I am. https://github.com/mcc85s/FightingEntropy/blob/main/Docs/2022_1201-(PowerShell%20Subreddit).pdf

```
Class Registry
{
    [String] $Path
    [String] $Name
    Registry([String]$Path)
    {
        $This.Path  = $Path
    }
    Registry([String]$Path,[String]$Name)
    {
        $This.Path  = $Path
        $This.Name  = $Name
    }
    [Object] Get()
    {
        $This.Test()
        If ($This.Name)
        {
            Return Get-ItemProperty -LiteralPath $This.Path -Name $This.Name
        }
        Else
        {
            Return Get-ItemProperty -LiteralPath $This.Path
        }
    }
    [Void] Test()
    {
        $Split = $This.Path.Split("\")
        $Path_ = $Split[0]
        ForEach ($Item in $Split[1..($Split.Count-1)])
        {
            $Path_ = $Path_, $Item -join '\'
            If (!(Test-Path $Path_))
            {
                New-Item -Path $Path_ -Verbose
            }
        }
    }
    [Void] Clear()
    {
        $This.Name  = $Null
        $This.Type  = $Null
        $This.Value = $Null
    }
    [Void] Set([Object]$Value)
```

```
    {
        $This.Test()
        Set-ItemProperty -LiteralPath $This.Path -Name $This.Name -Type "DWord" -Value $Value -Verbose
    }
    [Void] Set([String]$Type,[Object]$Value)
    {
        $This.Test()
        Set-ItemProperty -LiteralPath $This.Path -Name $This.Name -Type $Type -Value $Value -Verbose
    }
    [Void] Remove()
    {
        $This.Test()
        If ($This.Name)
        {
            Remove-ItemProperty -LiteralPath $This.Path -Name $This.Name -Verbose
        }
        Else
        {
            Remove-Item -LiteralPath $This.Path -Verbose
        }
    }
}
```

Class [Registry]

Class [ControlTemplate]

This is a class specifically meant for being a base class object, as it will be repeated for many of the classes below. If you see a class that says `Class <Name> : ControlTemplate`, then it is an EXTENSION of this class.

When that is the case, the class will be rather self-explanatory, and I will not make any comments about it. Some of the following classes WILL have SOME text-wrapping...

```
Class ControlTemplate
{
    [String]      $Source
    [String]        $Name
    [String] $DisplayName
    [UInt32]       $Value
    [String] $Description
    [String[]]   $Options
    [Object]      $Output
    ControlTemplate()
    {
        $This.Output  = @( )
    }
    Registry([String]$Path,[String]$Name)
    {
        $This.Output += [Registry]::New($Path,$Name)
    }
    [UInt32] GetWinVersion()
    {
        Return Get-ItemProperty 'HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion' | % ReleaseID
    }
}
```

Class [Telemetry : ControlTemplate]

Class [ControlTemplate]

```
Class Telemetry : ControlTemplate
{
    Telemetry() : base()
    {
        $This.Name        = "Telemetry"
        $This.DisplayName = "Telemetry"
        $This.Value       = 1
        $This.Description = "Various location and tracking features"
        $This.Options     = "Skip", "Enable*", "Disable"
```

```
        ('HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\DataCollection',
         'AllowTelemetry'),
        ('HKLM:\SOFTWARE\Policies\Microsoft\Windows\DataCollection',
         'AllowTelemetry'),
        ('HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Policies\DataCollection',
         'AllowTelemetry'),
        ('HKLM:\SOFTWARE\Policies\Microsoft\Windows\PreviewBuilds',
         'AllowBuildPreview'),
        ('HKLM:\SOFTWARE\Policies\Microsoft\Windows NT\CurrentVersion\Software Protection Platform',
         'NoGenTicket'),
        ('HKLM:\SOFTWARE\Policies\Microsoft\SQMClient\Windows',
         'CEIPEnable'),
        ('HKLM:\SOFTWARE\Policies\Microsoft\Windows\AppCompat',
         'AITEnable'),
        ('HKLM:\SOFTWARE\Policies\Microsoft\Windows\AppCompat',
         'DisableInventory'),
        ('HKLM:\SOFTWARE\Policies\Microsoft\AppV\CEIP',
         'CEIPEnable'),
        ('HKLM:\SOFTWARE\Policies\Microsoft\Windows\TabletPC',
         'PreventHandwritingDataSharing'),
        ('HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\TextInput',
         'AllowLinguisticDataCollection') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Telemetry"
                }
            }
            1
            {
                Write-Host "Enabling [~] Telemetry"
                $This.Output[0].Set(0)
                $This.Output[1].Set(0)
                If ([Environment]::Is64BitProcess)
                {
                    $This.Output[2].Set(0)
                }
                3..10 | % { $This.Output[$_].Remove() }
                $This.TelemetryTask() | % { Enable-ScheduledTask -TaskName $_ }
            }
            2
            {
                Write-Host "Disabling [~] Telemetry"
                $This.Output[0].Set(0)
                $This.Output[1].Set(0)
                If ([Environment]::Is64BitProcess)
                {
                    $This.Output[2].Set(0)
                }
                $This.Ouptut[ 3].Set(0)
                $This.Ouptut[ 4].Set(1)
                $This.Ouptut[ 5].Set(0)
                $This.Ouptut[ 6].Set(0)
                $This.Ouptut[ 7].Set(1)
                $This.Ouptut[ 8].Set(0)
                $This.Ouptut[ 9].Set(1)
                $This.Ouptut[10].Set(0)
                $This.TelemetryTask() | % { Disable-ScheduledTask -TaskName $_ }
            }
        }
```

```
    }
    [String[]] TelemetryTask()
    {
        Return "Microsoft\Windows\Application Experience\Microsoft Compatibility Appraiser",
               "Microsoft\Windows\Application Experience\ProgramDataUpdater",
               "Microsoft\Windows\Autochk\Proxy",
               "Microsoft\Windows\Customer Experience Improvement Program\Consolidator",
               "Microsoft\Windows\Customer Experience Improvement Program\UsbCeip",
               "Microsoft\Windows\DiskDiagnostic\Microsoft-Windows-DiskDiagnosticDataCollector",
               "Microsoft\Office\Office ClickToRun Service Monitor",
               "Microsoft\Office\OfficeTelemetryAgentFallBack2016",
               "Microsoft\Office\OfficeTelemetryAgentLogOn2016"
    }
}
```

Class [WiFiSense : ControlTemplate] Class [Telemetry : ControlTemplate]

```
Class WiFiSense : ControlTemplate
{
    WiFiSense()
    {
        $This.Name        = "WifiSense"
        $This.DisplayName = "Wi-Fi Sense"
        $This.Value       = 1
        $This.Description = "Lets devices more easily connect to a WiFi network"
        $This.Options     = "Skip", "Enable*", "Disable"

        ('HKLM:\SOFTWARE\Microsoft\PolicyManager\default\WiFi\AllowWiFiHotSpotReporting','Value'),
        ('HKLM:\SOFTWARE\Microsoft\PolicyManager\default\WiFi\AllowConnectToWiFiSenseHotspots','Value'),
        ('HKLM:\SOFTWARE\Microsoft\WcmSvc\wifinetworkmanager\config','AutoConnectAllowedOEM'),
        ('HKLM:\SOFTWARE\Microsoft\WcmSvc\wifinetworkmanager\config','WiFiSenseAllowed') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [-] Wi-Fi Sense"
                }
            }
            1
            {
                Write-Host "Enabling [~] Wi-Fi Sense"
                $This.Output[0].Set(1)
                $This.Output[1].Set(1)
                $This.Output[2].Set(0)
                $This.Output[3].Set(0)
            }
            2
            {
                Write-Host "Disabling [~] Wi-Fi Sense"
                $This.Output[0].Set(0)
                $This.Output[1].Set(0)
                $This.Output[2].Remove()
                $This.Output[3].Remove()
            }
        }
    }
}
```

_____/ Class [WiFiSense : ControlTemplate]
Class [SmartScreen : ControlTemplate] /-----\
/-----

```
Class SmartScreen : ControlTemplate
{
    SmartScreen()
    {
        $This.Name        = "SmartScreen"
        $This.DisplayName = "SmartScreen"
        $This.Value       = 1
        $This.Description = "Cloud-based anti-phishing and anti-malware component"
        $This.Options     = "Skip","Enable*","Disable"

        $Path             = Switch ([UInt32]($This.GetWinVersion() -ge 1703))
        {
            0 { $Null } 1 { Get-AppxPackage | ? Name -eq Microsoft.MicrosoftEdge | % PackageFamilyName }
        }

        $Phishing = "HKCU:","SOFTWARE","Classes","Local Settings","Software",
                    "Microsoft","Windows","CurrentVersion","AppContainer",
                    "Storage",$Path,"MicrosoftEdge","PhishingFilter" -join "\"

        ("HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer",
        "SmartScreenEnabled"),
        ("HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\AppHost",
        "EnableWebContentEvaluation"),
        ($Phishing,
        "EnabledV9"),
        ($Phishing,
        "PreventOverride") | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [-] SmartScreen Filter"
                }
            }
            1
            {
                Write-Host "Enabling [~] SmartScreen Filter"
                $This.Output[0].Set("String","RequireAdmin")
                1..3 | % { $This.Output[$_].Remove() }
            }
            2
            {
                Write-Host "Disabling [~] SmartScreen Filter"
                $This.Output[0].Set("String","Off")
                1..3 | % { $This.Output[$_].Set(0) }
            }
        }
    }
}
```

_____/
_____/ Class [SmartScreen : ControlTemplate]
Class [LocationTracking : ControlTemplate] /-----\
/-----

```
Class LocationTracking : ControlTemplate
{
    LocationTracking()
    {
        $This.Name        = "LocationTracking"
```

```
        $This.DisplayName = "Location Tracking"
        $This.Value       = 1
        $This.Description = "Monitors the current location of the system and manages geofences"
        $This.Options     = "Skip", "Enable*", "Disable"

        ('HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Sensor\Overrides\{BFA794E4-F964-4FDB-90F6-
51056BFE4B44}','SensorPermissionState'),
        ('HKLM:\SYSTEM\CurrentControlSet\Services\lfsvc\Service\Configuration','Status') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Location Tracking"
                }
            }
            1
            {
                Write-Host "Enabling [~] Location Tracking"
                $This.Output[0].Set(1)
                $This.Output[1].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Location Tracking"
                $This.Output[0].Set(0)
                $This.Output[1].Set(0)
            }
        }
    }
}
```

_____/
_____/ Class [LocationTracking : ControlTemplate]
Class [Feedback : ControlTemplate] /_____\
/_____

```
Class Feedback : ControlTemplate
{
    Feedback()
    {
        $This.Name        = "Feedback"
        $This.DisplayName = "Feedback"
        $This.Value       = 1
        $This.Description = "System Initiated User Feedback"
        $This.Options     = "Skip", "Enable*", "Disable"

        ('HKCU:\SOFTWARE\Microsoft\Siuf\Rules','NumberOfSIUFInPeriod'),
        ('HKLM:\SOFTWARE\Policies\Microsoft\Windows\DataCollection','DoNotShowFeedbackNotifications') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Feedback"
                }
            }
            1
```

```
            {
                Write-Host "Enabling [~] Feedback"
                $This.Output[0].Remove()
                $This.Output[1].Remove()
                ForEach ($Item in "Microsoft\Windows\Feedback\Siuf\DmClient",
                                  "Microsoft\Windows\Feedback\Siuf\DmClientOnScenarioDownload")
                {
                    Enable-ScheduledTask -TaskName $Item | Out-Null
                }
            }
            2
            {
                Write-Host "Disabling [~] Feedback"
                $This.Output[0].Set(0)
                $This.Output[1].Set(1)
                ForEach ($Item in "Microsoft\Windows\Feedback\Siuf\DmClient",
                                  "Microsoft\Windows\Feedback\Siuf\DmClientOnScenarioDownload")
                {
                    Disable-ScheduledTask -TaskName $Item | Out-Null
                }
            }
        }
    }
}
```

_____/ Class [AdvertisingID : ControlTemplate] /_____/ Class [Feedback : ControlTemplate] /_____\

```
Class AdvertisingID : ControlTemplate
{
    AdvertisingID()
    {
        $This.Name        = "AdvertisingID"
        $This.DisplayName = "Advertising ID"
        $This.Value       = 1
        $This.Description = "Allows Microsoft to display targeted ads"
        $This.Options     = "Skip", "Enable*", "Disable"

        ('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\AdvertisingInfo',
         'Enabled'),
        ('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Privacy',
         'TailoredExperiencesWithDiagnosticDataEnabled') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Advertising ID"
                }
            }
            1
            {
                Write-Host "Enabling [~] Advertising ID"
                $This.Output[0].Remove()
                $This.Output[1].Set(2)
            }
            2
            {
                Write-Host "Disabling [~] Advertising ID"
                $This.Output[0].Set(0)
                $This.Output[1].Set(0)
            }
        }
```

```
    }
}
```

_____/ Class [AdvertisingID : ControlTemplate] /_____\
Class [Cortana : ControlTemplate] /_____/

```
Class Cortana : ControlTemplate
{
    Cortana()
    {
        $This.Name        = "Cortana"
        $This.DisplayName = "Cortana"
        $This.Value       = 1
        $This.Description = "(Master Chief/Microsoft)'s personal voice assistant"
        $This.Options     = "Skip", "Enable*", "Disable"

        ("HKCU:\SOFTWARE\Microsoft\Personalization\Settings","AcceptedPrivacyPolicy"),
        ("HKCU:\SOFTWARE\Microsoft\InputPersonalization\TrainedDataStore","HarvestContacts"),
        ("HKCU:\SOFTWARE\Microsoft\InputPersonalization","RestrictImplicitTextCollection"),
        ("HKCU:\SOFTWARE\Microsoft\InputPersonalization","RestrictImplicitInkCollection"),
        ("HKLM:\SOFTWARE\Policies\Microsoft\Windows\Windows Search","AllowCortanaAboveLock"),
        ("HKLM:\SOFTWARE\Policies\Microsoft\Windows\Windows Search","ConnectedSearchUseWeb"),
        ("HKLM:\SOFTWARE\Policies\Microsoft\Windows\Windows Search","ConnectedSearchPrivacy"),
        ("HKLM:\SOFTWARE\Policies\Microsoft\Windows\Windows Search","DisableWebSearch"),
        ("HKCU:\SOFTWARE\Microsoft\Speech_OneCore\Preferences","VoiceActivationEnableAboveLockscreen"),
        ("HKLM:\SOFTWARE\Policies\Microsoft\InputPersonalization","AllowInputPersonalization"),
        ("HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced","ShowCortanaButton") | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Cortana"
                }
            }
            1
            {
                Write-Host "Enabling [~] Cortana"
                $This.Output[0].Remove()
                $This.Output[1].Remove()
                $This.Output[2].Set(0)
                $This.Output[3].Set(0)
                $This.Output[4].Remove()
                $This.Output[5].Remove()
                $This.Output[6].Remove()
                $This.Output[7].Set(1)
                $This.Output[8].Remove()
                $This.Output[9].Set(0)
            }
            2
            {
                Write-Host "Disabling [~] Cortana"
                $This.Output[0].Set(0)
                $This.Output[1].Set(1)
                $This.Output[2].Set(1)
                $This.Output[3].Set(0)
                $This.Output[4].Set(0)
                $This.Output[5].Set(1)
                $This.Output[6].Set(3)
                $This.Output[7].Set(0)
                $This.Output[8].Set(0)
                $This.Output[9].Set(1)
            }
```

```
        }
    }
}
```

Class [Cortana : ControlTemplate]

Class [CortanaSearch : ControlTemplate]

```
Class CortanaSearch : ControlTemplate
{
    CortanaSearch()
    {
        $This.Name        = "CortanaSearch"
        $This.DisplayName = "Cortana Search"
        $This.Value       = 1
        $This.Description = "Allows Cortana to create search indexing for faster system search results"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry("HKLM:\SOFTWARE\Policies\Microsoft\Windows\Windows Search","AllowCortana")
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Cortana Search"
                }
            }
            1
            {
                Write-Host "Enabling [~] Cortana Search"
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Cortana Search"
                $This.Output[0].Set(0)
            }
        }
    }
}
```

Class [CortanaSearch : ControlTemplate]

Class [ErrorReporting : ControlTemplate]

```
Class ErrorReporting : ControlTemplate
{
    ErrorReporting()
    {
        $This.Name        = "ErrorReporting"
        $This.DisplayName = "Error Reporting"
        $This.Value       = 1
        $This.Description = "If Windows has an issue, it sends Microsoft a detailed report"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry("HKLM:\SOFTWARE\Microsoft\Windows\Windows Error Reporting","Disabled")
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
```

```
                    Write-Host "Skipping [!] Error Reporting"
                }
            }
            1
            {
                Write-Host "Enabling [~] Error Reporting"
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Error Reporting"
                $This.Output[0].Set(0)
            }
        }
    }
}
```

_____/ Class [ErrorReporting : ControlTemplate] /_____\

_____/ Class [AutoLoggerFile : ControlTemplate] /_____\

```
Class AutoLoggerFile : ControlTemplate
{
    AutoLoggerFile()
    {
        $This.Name        = "AutoLoggerFile"
        $This.DisplayName = "Automatic Logger File"
        $This.Value       = 1
        $This.Description = "Lets you track trace provider actions while Windows is booting"
        $This.Options     = "Skip", "Enable*", "Disable"

        ($This.WmiRegistry(),
        "Start"),
        ("$($This.WmiRegistry())\{DD17FA14-CDA6-7191-9B61-37A28F7A10DA}",
        "Start") | % {

            $This.Registry($_[0],$_[1])
        }
    }
    [String] WmiRegistry()
    {
        Return "HKLM:\SYSTEM\ControlSet001\Control\WMI\AutoLogger\AutoLogger-Diagtrack-Listener"
    }
    [String] AutoLogger()
    {
        Return "$Env:PROGRAMDATA\Microsoft\Diagnosis\ETLLogs\AutoLogger"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] AutoLogger"
                }
            }
            1
            {
                Write-Host "Unrestricting [~] AutoLogger"
                icacls $This.AutoLogger() /grant:r SYSTEM:`(OI`)`(CI`)F
                $This.Output[0].Set(1)
                $This.Output[1].Set(1)
            }
            2
            {
                Write-Host "Removing [~] AutoLogger, and restricting directory"
                icacls $This.AutoLogger() /deny SYSTEM:`(OI`)`(CI`)F
                Remove-Item "$($This.AutoLogger())\AutoLogger-Diagtrack-Listener.etl" -EA 0 -Verbose
                $This.Output[0].Set(0)
```

```
                $This.Output[1].Set(0)
            }
        }
    }
}
```

_____/ Class [AutoLoggerFile : ControlTemplate]
Class [DiagTrack : ControlTemplate] /_____\
/_____

```
Class DiagTrack : ControlTemplate
{
    DiagTrack()
    {
        $This.Name        = "DiagTracking"
        $This.DisplayName = "Diagnostics Tracking"
        $This.Value       = 1
        $This.Description = "Connected User Experiences and Telemetry"
        $This.Options     = "Skip", "Enable*", "Disable"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Diagnostics Tracking"
                }
            }
            1
            {
                Write-Host "Enabling [~] Diagnostics Tracking"
                Get-Service -Name DiagTrack
                Set-Service -Name DiagTrack -StartupType Automatic
                Start-Service -Name DiagTrack
            }
            2
            {
                Write-Host "Disabling [~] Diagnostics Tracking"
                Stop-Service -Name DiagTrack
                Set-Service -Name DiagTrack -StartupType Disabled
                Get-Service -Name DiagTrack
            }
        }
    }
}
```

_____/ Class [DiagTrack : ControlTemplate]
Class [WAPPush : ControlTemplate] /_____\
/_____

```
Class WAPPush : ControlTemplate
{
    WAPPush()
    {
        $This.Name        = "WAPPush"
        $This.DisplayName = "WAP Push"
        $This.Value       = 1
        $This.Description = "Device Management Wireless Application Protocol"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry("HKLM:\SYSTEM\CurrentControlSet\Services\dmwappushservice","DelayedAutoStart")
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
```

```
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] WAP Push"
                }
            }
            1
            {
                Write-Host "Enabling [~] WAP Push Service"
                Set-Service -Name dmwappushservice -StartupType Automatic
                Start-Service -Name dmwappushservice
                $This.Output[0].Set(1)
                Get-Service -Name dmwappushservice
            }
            2
            {
                Write-Host "Disabling [~] WAP Push Service"
                Stop-Service -Name dmwappushservice
                Set-Service -Name dmwappushservice -StartupType Disabled
                Get-Service -Name dmwappushservice
            }
        }
    }
}
```

Enum [PrivacyType]

This is an Enum type that is specifically meant to access ALL of the above classes that are based on the ControlTemplate, and they all relate to some aspect of “Privacy”. There will be several other categories that enumerate additional types below.

```
Enum PrivacyType
{
    Telemetry
    WifiSense
    SmartScreen
    LocationTracking
    Feedback
    AdvertisingID
    Cortana
    CortanaSearch
    ErrorReporting
    AutologgerFile
    DiagTrack
    WAPPush
}
```

Class [PrivacyList]

This is specifically meant for turning all of the [PrivacyType]'s, into a list object.

```
Class PrivacyList
{
    [Object]  $Output
    PrivacyList()
    {
        $This.Output      = @( )
        ForEach ($Name in [System.Enum]::GetNames([PrivacyType]))
        {
            $Item         = Switch ($Name)
            {
                Telemetry        { [Telemetry]::New()        }
                WiFiSense        { [WiFiSense]::New()        }
                SmartScreen      { [SmartScreen]::New()      }
```

```
                LocationTracking { [LocationTracking]::New() }
                Feedback         { [Feedback]::New()         }
                AdvertisingID    { [AdvertisingID]::New()    }
                Cortana          { [Cortana]::New()          }
                CortanaSearch    { [CortanaSearch]::New()    }
                ErrorReporting   { [ErrorReporting]::New()   }
                AutologgerFile   { [AutologgerFile]::New()   }
                DiagTrack        { [DiagTrack]::New()        }
                WAPPush          { [WAPPush]::New()          }
            }
            $Item.Source  = "Privacy"
            $This.Output += $Item
        }
    }
}
```

Class [UpdateMSProducts : ControlTemplate]

```
Class UpdateMSProducts : ControlTemplate
{
    UpdateMSProducts()
    {
        $This.Name        = "UpdateMSProducts"
        $This.DisplayName = "Update MS Products"
        $This.Value       = 2
        $This.Description = "Searches Windows Update for Microsoft Products"
        $This.Options     = "Skip", "Enable", "Disable*"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Update Microsoft Products"
                }
            }
            1
            {
                Write-Host "Enabling [~] Update Microsoft Products"
                $This.ComMusm().AddService2("7971f918-a847-4430-9279-4a52d1efe18d", 7, "")
            }
            2
            {
                Write-Host "Disabling [~] Update Microsoft Products"
                $This.ComMusm().RemoveService("7971f918-a847-4430-9279-4a52d1efe18d")
            }
        }
    }
    [Object] ComMusm()
    {
        Return New-Object -ComObject Microsoft.Update.ServiceManager
    }
}
```

Class [CheckForWindowsUpdate : ControlTemplate]

```
Class CheckForWindowsUpdate : ControlTemplate
{
    CheckForWindowsUpdate()
    {
        $This.Name        = "CheckForWindowsUpdate"
        $This.DisplayName = "Check for Windows Updates"
```

```
        $This.Value       = 1
        $This.Description = "Allows Windows Update to work automatically"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry("HKLM:\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate","SetDisableUXWUAccess")
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Check for Windows Updates"
                }
            }
            1
            {
                Write-Host "Enabling [~] Check for Windows Updates"
                $This.Output[0].Set(0)
            }
            2
            {
                Write-Host "Disabling [~] Check for Windows Updates"
                $This.Output[0].Set(1)
            }
        }
    }
}
```

_____/ Class [CheckForWindowsUpdate : ControlTemplate]
Class [WinUpdateType : ControlTemplate] /_____\

```
Class WinUpdateType : ControlTemplate
{
    WinUpdateType()
    {
        $This.Name        = "WinUpdateType"
        $This.DisplayName = "Windows Update Type"
        $This.Value       = 3
        $This.Description = "Allows Windows Update to work automatically"
        $This.Options     = "Skip", "Notify", "Auto DL", "Auto DL+Install*", "Manual"

        $This.Registry("HKLM:\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\AU","AUOptions")
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Windows Update Check Type"
                }
            }
            1
            {
                Write-Host "Enabling [~] Notify for Windows Update downloads, notify to install"
                $This.Output[0].Set(2)
            }
            2
            {
                Write-Host "Enabling [~] Automatically download Windows Updates, notify to install"
                $This.Output[0].Set(3)
            }
            3
            {
                Write-Host "Enabling [~] Automatically download Windows Updates, schedule to install"
```

```
                $This.Output[0].Set(4)
            }
            4
            {
                Write-Host "Enabling [~] Allow local administrator to choose automatic updates"
                $This.Output[0].Set(5)
            }
        }
    }
}
```

Class [WinUpdateDownload : ControlTemplate]

Class [WinUpdateType : ControlTemplate]

```
Class WinUpdateDownload : ControlTemplate
{
    WinUpdateDownload()
    {
        $This.Name        = "WinUpdateDownload"
        $This.DisplayName = "Windows Update Download"
        $This.Value       = 1
        $This.Description = "Selects a source from which to pull Windows Updates"
        $This.Options     = "Skip", "P2P*", "Local Only", "Disable"

        ("HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\DeliveryOptimization\Config",
         "DODownloadMode"),
        ("HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\DeliveryOptimization",
         "SystemSettingsDownloadMode"),
        ("HKLM:\SOFTWARE\Policies\Microsoft\Windows\DeliveryOptimization",
         "SystemSettingsDownloadMode"),
        ("HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\DeliveryOptimization",
         "DODownloadMode") | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] "
                }
            }
            1
            {
                Write-Host "Enabling [~] Unrestricting Windows Update P2P to Internet"
                $This.Output[0].Remove()
                $This.Output[1].Remove()
            }
            2
            {
                Write-Host "Enabling [~] Restricting Windows Update P2P only to local network"
                $This.Output[1].Set(3)
                Switch($This.GetWinVersion())
                {
                    1507
                    {
                        $This.Output[0]
                    }
                    {$_ -gt 1507 -and $_ -le 1607}
                    {
                        $This.Output[0].Set(1)
                    }
                    Default
                    {
                        $This.Output[0].Remove()
```

```
                    }
                }
            }
            3
            {
                Write-Host "Disabling [~] Windows Update P2P"
                $This.Output[1].Set(3)
                Switch ($This.GetWinVersion())
                {
                    1507
                    {
                        $This.Output[0].Set(0)
                    }
                    Default
                    {
                        $This.Output[3].Set(100)
                    }
                }
            }
        }
    }
}
```

_____/
_____/ Class [WinUpdateDownload : ControlTemplate]
Class [UpdateMSRT : ControlTemplate] /_____\
/_____

```
Class UpdateMSRT : ControlTemplate
{
    UpdateMSRT()
    {
        $This.Name        = "UpdateMSRT"
        $This.DisplayName = "Update MSRT"
        $This.Value       = 1
        $This.Description = "Allows updates for the Malware Software Removal Tool"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry("HKLM:\SOFTWARE\Policies\Microsoft\MRT","DontOfferThroughWUAU")
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Malicious Software Removal Tool Update"
                }
            }
            1
            {
                Write-Host "Enabling [~] Malicious Software Removal Tool Update"
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Malicious Software Removal Tool Update"
                $This.Output[0].Set(1)
            }
        }
    }
}
```

_____/
_____/ Class [UpdateMSRT : ControlTemplate]
Class [UpdateDriver : ControlTemplate] /_____\
/_____

```
Class UpdateDriver : ControlTemplate
```

```
{
    UpdateDriver()
    {
        $This.Name        = "UpdateDriver"
        $This.DisplayName = "Update Driver"
        $This.Value       = 1
        $This.Description = "Allows drivers to be downloaded from Windows Update"
        $This.Options     = "Skip", "Enable*", "Disable"

        ("HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\DriverSearching",
         "SearchOrderConfig"),
        ("HKLM:\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate",
         "ExcludeWUDriversInQualityUpdate"),
        ("HKLM:\SOFTWARE\Policies\Microsoft\Windows\Device Metadata",
         "PreventDeviceMetadataFromNetwork") | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Driver update through Windows Update"
                }
            }
            1
            {
                Write-Host "Enabling [~] Driver update through Windows Update"
                $This.Output[0].Remove()
                $This.Output[1].Remove()
                $This.Output[2].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Driver update through Windows Update"
                $This.Output[0].Set(0)
                $This.Output[1].Set(1)
                $This.Output[2].Set(1)
            }
        }
    }
}
```

Class [RestartOnUpdate : ControlTemplate]

Class [UpdateDriver : ControlTemplate]

```
Class RestartOnUpdate : ControlTemplate
{
    RestartOnUpdate()
    {
        $This.Name        = "RestartOnUpdate"
        $This.DisplayName = "Restart on Update"
        $This.Value       = 1
        $This.Description = "Reboots the machine when an update is installed and requires it"
        $This.Options     = "Skip", "Enable*", "Disable"

        ("HKLM:\SOFTWARE\Microsoft\WindowsUpdate\UX\Settings",
         "UxOption"),
        ("HKLM:\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\AU",
         "NoAutoRebootWithLoggOnUsers"),
        ("HKLM:\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\AU",
         "AUPowerManagement") | % {

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

```
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Windows Update Automatic Restart"
                }
            }
            1
            {
                Write-Host "Enabling [~] Windows Update Automatic Restart"
                $This.Output[0].Set(0)
                $This.Output[1].Remove()
                $This.Output[2].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Windows Update Automatic Restart"
                $This.Output[0].Set(1)
                $This.Output[1].Set(1)
                $This.Output[2].Set(0)
            }
        }
    }
}
```

Class [AppAutoDownload : ControlTemplate]

Class [RestartOnUpdate : ControlTemplate]

```
Class AppAutoDownload : ControlTemplate
{
    AppAutoDownload()
    {
        $This.Name        = "AppAutoDownload"
        $This.DisplayName = "Consumer App Auto Download"
        $This.Value       = 1
        $This.Description = "Provisioned Windows Store applications are downloaded"
        $This.Options     = "Skip", "Enable*", "Disable"

        ("HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsStore\WindowsUpdate",
        "AutoDownload"),
        ("HKLM:\SOFTWARE\Policies\Microsoft\Windows\CloudContent",
        "DisableWindowsConsumerFeatures") | % {

            $This.Registry($_[0],$_[1])
        }
    }
    [String] CloudCache()
    {
        Return "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\CloudStore\Store\Cache\DefaultAccount"
    }
    [String] PlaceHolder()
    {
        Return "*windows.data.placeholdertilecollection\Current"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] App Auto Download"
                }
            }
```

```
            1
            {
                Write-Host "Enabling [~] App Auto Download"
                $This.Output[0].Set(0)
                $This.Output[1].Remove()
            }
            2
            {
                Write-Host "Disabling [~] App Auto Download"
                $This.Output[0].Set(2)
                $This.Output[1].Set(1)
                If ($This.GetWinVersion() -le 1803)
                {
                    $Key  = Get-ChildItem $This.CloudCache() -Recurse | ? Name -like $This.Placeholder()
                    $Data = (Get-ItemProperty -Path $Key.PSPath).Data
                    Set-ItemProperty -Path $Key -Name Data -Type Binary -Value $Data[0..15] -Verbose
                    Stop-Process -Name ShellExperienceHost -Force
                }
            }
        }
    }
}
```

_____/
_____/ Class [AppAutoDownload : ControlTemplate]
Class [UpdateAvailablePopup : ControlTemplate] /_____\
/_____

```
Class UpdateAvailablePopup : ControlTemplate
{
    UpdateAvailablePopup()
    {
        $This.Name        = "UpdateAvailablePopup"
        $This.DisplayName = "Update Available Pop-up"
        $This.Value       = 1
        $This.Description = "If an update is available, a (pop-up/notification) will appear"
        $This.Options     = "Skip", "Enable*", "Disable"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Update Available Popup"
                }
            }
            1
            {
                Write-Host "Enabling [~] Update Available Popup"
                $This.MUSNotify()  | % {
                    ICACLS $_ /remove:d '"Everyone"'
                    ICACLS $_ /grant ('Everyone' + ':(OI)(CI)F')
                    ICACLS $_ /setowner 'NT SERVICE\TrustedInstaller'
                    ICACLS $_ /remove:g '"Everyone"'
                }
            }
            2
            {
                Write-Host "Disabling [~] Update Available Popup"
                $This.MUSNotify() | % {

                    Takeown /f $_
                    ICACLS $_ /deny '"Everyone":(F)'
                }
            }
        }
    }
    [String[]] MUSNotify()
    {
```

```
        Return @("","ux" | % { "$Env:windir\System32\musnotification$_.exe" })
    }
}
```

Enum [WindowsUpdateType]

Enum type for Windows Update related control objects.

```
Enum WindowsUpdateType
{
    UpdateMSProducts
    CheckForWindowsUpdate
    WinUpdateType
    WinUpdateDownload
    UpdateMSRT
    UpdateDriverRestartOnUpdate
    AppAutoDownload
    UpdateAvailablePopup
}
```

Class [WindowsUpdateList]

List of enumeration objects for Windows Update related control objects with index and description, for GUI.

```
Class WindowsUpdateList
{
    [Object]  $Output
    WindowsUpdateList()
    {
        $This.Output     = @( )
        ForEach ($Name in [System.Enum]::GetNames([WindowsUpdateType]))
        {
            $Name
            $Item        = Switch ($Name)
            {
                UpdateMSProducts      { [UpdateMSProducts]::New()      }
                CheckForWindowsUpdate { [CheckForWindowsUpdate]::New() }
                WinUpdateType         { [WinUpdateType]::New()         }
                WinUpdateDownload     { [WinUpdateDownload]::New()     }
                UpdateMSRT            { [UpdateMSRT]::New()            }
                UpdateDriver          { [UpdateDriver]::New()          }
                RestartOnUpdate       { [RestartOnUpdate]::New()       }
                AppAutoDownload       { [AppAutoDownload]::New()       }
                UpdateAvailablePopup  { [UpdateAvailablePopup]::New()  }
            }
            $This.Output += $Item
        }

        $This.Output | % { $_.Source = "WindowsUpdate" }
    }
}
```

Class [UAC : ControlTemplate]

```
Class UAC : ControlTemplate
{
    UAC()
    {
        $This.Name        = "UAC"
        $This.DisplayName = "User Access Control"
        $This.Value       = 2
```

```
        $This.Description = "Sets restrictions/permissions for programs"
        $This.Options     = "Skip", "Lower", "Normal*", "Higher"

        ($This.RegPath(),"ConsentPromptBehaviorAdmin"),
        ($This.RegPath(),"PromptOnSecureDesktop") | % {

            $This.Registry($_[0],$_[1])
        }
    }
    [String] RegPath()
    {
        Return "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] UAC Level"
                }
            }
            1
            {
                Write-Host "Setting [~] UAC Level (Low)"
                $This.Output[0].Set(0)
                $This.Output[1].Set
            }
            2
            {
                Write-Host "Setting [~] UAC Level (Default)"
                $This.Output[0].Set(5)
                $This.Output[1].Set(1)
            }
            3
            {
                Write-Host "Setting [~] UAC Level (High)"
                $This.Output[0].Set(2)
                $This.Output[1].Set(1)
            }
        }
    }
}
```

_____/
_____/ Class [UAC : ControlTemplate]
Class [SharingMappedDrives : ControlTemplate] /_____\
/_____

```
Class SharingMappedDrives : ControlTemplate
{
    SharingMappedDrives()
    {
        $This.Name        = "SharingMappedDrives"
        $This.DisplayName = "Share Mapped Drives"
        $This.Value       = 2
        $This.Description = "Shares any mapped drives to all users on the machine"
        $This.Options     = "Skip", "Enable", "Disable*"

        $This.Registry($This.RegPath(),"EnableLinkedConnections")
    }
    [String] RegPath()
    {
        Return "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
```

```
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Sharing mapped drives between users"
                }
            }
            1
            {
                Write-Host "Enabling [~] Sharing mapped drives between users"
                $This.Output[0].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Sharing mapped drives between users"
                $This.Output[0].Remove()
            }
        }
    }
}
```

_____/
_____/ Class [SharingMappedDrives : ControlTemplate]
Class [AdminShares : ControlTemplate] /_____\
/_____

```
Class AdminShares : ControlTemplate
{
    AdminShares()
    {
        $This.Name        = "AdminShares"
        $This.DisplayName = "Administrative File Shares"
        $This.Value       = 1
        $This.Description = "Reveals default system administration file shares"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry("HKLM:\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters","AutoShareWks")
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Hidden administrative shares"
                }
            }
            1
            {
                Write-Host "Enabling [~] Hidden administrative shares"
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Hidden administrative shares"
                $This.Output[0].Set(0)
            }
        }
    }
}
```

_____/
_____/ Class [AdminShares : ControlTemplate]
Class [Firewall : ControlTemplate] /_____\
/_____

```
Class Firewall : ControlTemplate
{
    Firewall()
    {
```

```
        $This.Name        = "Firewall"
        $This.DisplayName = "Firewall"
        $This.Value       = 1
        $This.Description = "Enables the default firewall profile"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry('HKLM:\SOFTWARE\Policies\Microsoft\WindowsFirewall\StandardProfile','EnableFirewall')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Firewall Profile"
                }
            }
            1
            {
                Write-Host "Enabling [~] Firewall Profile"
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Firewall Profile"
                $This.Output[0].Set(0)
            }
        }
    }
}
```

_____/
_____/ Class [Firewall : ControlTemplate]
Class [WinDefender : ControlTemplate] /_____\
/_____

```
Class WinDefender : ControlTemplate
{
    WinDefender()
    {
        $This.Name        = "WinDefender"
        $This.DisplayName = "Windows Defender"
        $This.Value       = 1
        $This.Description = "Toggles Windows Defender, system default (Anti-Virus/Malware) utility"
        $This.Options     = "Skip", "Enable*", "Disable"

        ("HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender",
        "DisableAntiSpyware"),
        ("HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run",
        "WindowsDefender"),
        ("HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run",
        "SecurityHealth"),
        ("HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender\Spynet",
        $Null),
        ("HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender\Spynet",
        "SpynetReporting"),
        ("HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender\Spynet",
        "SubmitSamplesConsent") | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
```

```
                    Write-Host "Skipping [!] Windows Defender"
                }
            }
            1
            {
                Write-Host "Enabling [~] Windows Defender"
                $This.Output[0].Remove()
                Switch ($This.GetWinVersion())
                {
                    {$_ -lt 1703}
                    {
                        $This.Output[1].Set("ExpandString","`"%ProgramFiles%\Windows Defender\MSASCuiL.exe`"")
                    }
                    Default
                    {
                        $This.Output[2].Set("ExpandString","`"%ProgramFiles%\Windows Defender\MSASCuiL.exe`"")

                    }
                }
                $This.Output[3].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Windows Defender"
                Switch ($This.GetWinVersion())
                {
                    {$_ -lt 1703}
                    {
                        $This.Output[1].Remove()
                    }
                    Default
                    {
                        $This.Output[2].Remove()
                    }
                }
                $This.Output[0].Set(1)
                $This.Output[4].Set(0)
                $This.Output[5].Set(2)
            }
        }
    }
}
```

Class [HomeGroups : ControlTemplate] Class [WinDefender : ControlTemplate]

```
Class HomeGroups : ControlTemplate
{
    HomeGroups()
    {
        $This.Name        = "HomeGroups"
        $This.DisplayName = "Home Groups"
        $This.Value       = 1
        $This.Description = "Toggles the use of home groups, essentially a home-based workgroup"
        $This.Options     = "Skip", "Enable*", "Disable"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Home groups services"
                }
            }
            1
            {
                Write-Host "Enabling [~] Home groups services"
```

```
                Set-Service   -Name HomeGroupListener -StartupType Manual
                Set-Service   -Name HomeGroupProvider -StartupType Manual
                Start-Service -Name HomeGroupProvider
            }
            2
            {
                Write-Host "Disabling [~] Home groups services"
                Stop-Service  -Name HomeGroupListener
                Set-Service   -Name HomeGroupListener -StartupType Disabled
                Stop-Service  -Name HomeGroupProvider
                Set-Service   -Name HomeGroupProvider -StartupType Disabled
            }
        }
    }
}
```

Class [RemoteAssistance : ControlTemplate] / Class [HomeGroups : ControlTemplate]

```
Class RemoteAssistance : ControlTemplate
{
    RemoteAssistance()
    {
        $This.Name        = "RemoteAssistance"
        $This.DisplayName = "Remote Assistance"
        $This.Value       = 1
        $This.Description = "Toggles the ability to use Remote Assistance"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry("HKLM:\SYSTEM\CurrentControlSet\Control\Remote Assistance","fAllowToGetHelp")
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Remote Assistance"
                }
            }
            1
            {
                Write-Host "Enabling [~] Remote Assistance"
                $This.Output[0].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Remote Assistance"
                $This.Output[0].Set(0)
            }
        }
    }
}
```

Class [RemoteDesktop : ControlTemplate] / Class [RemoteAssistance : ControlTemplate]

```
Class RemoteDesktop : ControlTemplate
{
    RemoteDesktop()
    {
        $This.Name        = "RemoteDesktop"
        $This.DisplayName = "Remote Desktop"
        $This.Value       = 2
        $This.Description = "Toggles the ability to use Remote Desktop"
```

```
        $This.Options     = "Skip", "Enable", "Disable*"

        ("HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server",
         "fDenyTSConnections"),
        ("HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp",
         "UserAuthentication") | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Remote Desktop"
                }
            }
            1
            {
                Write-Host "Enabling [~] Remote Desktop"
                $This.Output[0].Set(0)
                $This.Output[1].Set(0)
            }
            2
            {
                Write-Host "Disabling [~] Remote Desktop"
                $This.Output[0].Set(1)
                $This.Output[1].Set(1)
            }
        }
    }
}
```

_____/
_____/ Class [RemoteDesktop : ControlTemplate]
Enum [ServiceType] /_____\
/_____

```
Enum ServiceType
{
    UAC
    SharingMappedDrives
    AdminShares
    Firewall
    WinDefender
    HomeGroups
    RemoteAssistance
    RemoteDesktop
}
```

_____/
_____/ Enum [ServiceType]
Class [ServiceList] /_____\
/_____

```
Class ServiceList
{
    [Object]  $Output
    ServiceList()
    {
        $This.Output     = @( )
        ForEach ($Name in [System.Enum]::GetNames([ServiceType]))
        {
            $Item        = Switch ($Name)
            {
                UAC                 { [UAC]::New()                 }
                SharingMappedDrives { [SharingMappedDrives]::New() }
```

```
                AdminShares         { [AdminShares]::New()         }
                Firewall            { [Firewall]::New()            }
                WinDefender         { [WinDefender]::New()         }
                Homegroups          { [HomeGroups]::New()          }
                RemoteAssistance    { [RemoteAssistance]::New()    }
                RemoteDesktop       { [RemoteDesktop]::New()       }
            }
            $Item.Source  = "Service"
            $This.Output += $Item
        }
    }
}
```

Class [ServiceList] /

Class [CastToDevice : ControlTemplate] /

```
    Class CastToDevice : ControlTemplate
    {
        CastToDevice()
        {
            $This.Name        = "CastToDevice"
            $This.DisplayName = "Cast To Device"
            $This.Value       = 1
            $This.Description = "Adds a context menu item for casting to a device"
            $This.Options     = "Skip", "Enable*", "Disable"

            ("HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Shell Extensions\Blocked",
            "{7AD84985-87B4-4a16-BE58-8B72A5B390F7}") | % {

                $This.Registry($_[0],$_[1])
            }
        }
        SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
        {
            Switch ($Mode)
            {
                0
                {
                    If ($ShowSkipped)
                    {
                        Write-Host "Skipping [!] 'Cast to device' context menu item"
                    }
                }
                1
                {
                    Write-Host "Enabling [~] 'Cast to device' context menu item"
                    $This.Output[0].Remove()
                }
                2
                {
                    Write-Host "Disabling [~] 'Cast to device' context menu item"
                    $This.Output[0].Set("String","Play to Menu")
                }
            }
        }
    }
```

Class [CastToDevice : ControlTemplate] /

Class [PreviousVersions : ControlTemplate] /

```
Class PreviousVersions : ControlTemplate
{
    PreviousVersions()
    {
        $This.Name        = "PreviousVersions"
        $This.DisplayName = "Previous Versions"
```

```
        $This.Value       = 1
        $This.Description = "Adds a context menu item to select a previous version of a file"
        $This.Options     = "Skip", "Enable*", "Disable"

        ("HKCR:\AllFilesystemObjects\$($This.ShellEx())",
        $Null),
        ("HKCR:\CLSID\{450D8FBA-AD25-11D0-98A8-0800361B1103}\$($This.ShellEx())",
        $Null),
        ("HKCR:\Directory\$($This.ShellEx())",
        $Null),
        ("HKCR:\Drive\$($This.ShellEx())",
        $Null) | % {

            $This.Registry($_[0],$_[1])
        }
    }
    [String] ShellEx()
    {
        Return "shellex\ContextMenuHandlers\{596AB062-B4D2-4215-9F74-E9109B0A8153}"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] 'Previous versions' context menu item"
                }
            }
            1
            {
                Write-Host "Enabling [~] 'Previous versions' context menu item"
                $This.Output[0].Get()
                $This.Output[1].Get()
                $This.Output[2].Get()
                $This.Output[3].Get()
            }
            2
            {
                Write-Host "Disabling [~] 'Previous versions' context menu item"
                $This.Output[0].Remove()
                $This.Output[1].Remove()
                $This.Output[2].Remove()
                $This.Output[3].Remove()
            }
        }
    }
}
```

Class [PreviousVersions : ControlTemplate]

Class [IncludeInLibrary : ControlTemplate]

```
Class IncludeInLibrary : ControlTemplate
{
    IncludeInLibrary()
    {
        $This.Name        = "IncludeInLibrary"
        $This.DisplayName = "Include in Library"
        $This.Value       = 1
        $This.Description = "Adds a context menu item to include a selection in library items"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry("HKCR:\Folder\ShellEx\ContextMenuHandlers\Library Location","(Default)")
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
```

```
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] 'Include in Library' context menu item"
                }
            }
            1
            {
                Write-Host "Enabling [~] 'Include in Library' context menu item"
                $This.Output[0].Set("String","{3dad6c5d-2167-4cae-9914-f99e41c12cfa}")
            }
            2
            {
                Write-Host "Disabling [~] 'Include in Library' context menu item"
                $This.Output[0].Set("String","")
            }
        }
    }
}
```

_____/
_____/ Class [IncludeInLibrary : ControlTemplate]
Class [PinToStart : ControlTemplate] /_____\
/_____

```
Class PinToStart : ControlTemplate
{
    PinToStart()
    {
        $This.Name        = "PinToStart"
        $This.DisplayName = "Pin to Start"
        $This.Value       = 1
        $This.Description = "Adds a context menu item to pin an item to the start menu"
        $This.Options     = "Skip", "Enable*", "Disable"

        ('HKCR:\*\shellex\ContextMenuHandlers\{90AA3A4E-1CBA-4233-B8BB-535773D48449}',
         '(Default)'),
        ('HKCR:\*\shellex\ContextMenuHandlers\{a2a9545d-a0c2-42b4-9708-a0b2badd77c8}',
         '(Default)'),
        ('HKCR:\Folder\shellex\ContextMenuHandlers\PintoStartScreen',
         '(Default)'),
        ('HKCR:\exefile\shellex\ContextMenuHandlers\PintoStartScreen',
         '(Default)'),
        ('HKCR:\Microsoft.Website\shellex\ContextMenuHandlers\PintoStartScreen',
         '(Default)'),
        ('HKCR:\mscfile\shellex\ContextMenuHandlers\PintoStartScreen',
         '(Default)') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] 'Pin to Start' context menu item"
                }
            }
            1
            {
                Write-Host "Enabling [~] 'Pin to Start' context menu item"
                $This.Output[0].Set("String","Taskband Pin")
                $This.Output[1].Set("String","Start Menu Pin")
                $This.Output[2].Set("String","{470C0EBD-5D73-4d58-9CED-E91E22E23282}")
                $This.Output[3].Set("String","{470C0EBD-5D73-4d58-9CED-E91E22E23282}")
                $This.Output[4].Set("String","{470C0EBD-5D73-4d58-9CED-E91E22E23282}")
                $This.Output[5].Set("String","{470C0EBD-5D73-4d58-9CED-E91E22E23282}")
```

```
            }
            2
            {
                Write-Host "Disabling [~] 'Pin to Start' context menu item"
                $This.Output[0].Remove()
                $This.Output[1].Remove()
                $This.Output[2].Set("String","")
                $This.Output[3].Set("String","")
                $This.Output[4].Set("String","")
                $This.Output[5].Set("String","")
            }
        }
    }
}
```

Class [PinToQuickAccess : ControlTemplate] / Class [PinToStart : ControlTemplate]

```
Class PinToQuickAccess : ControlTemplate
{
    PinToQuickAccess()
    {
        $This.Name        = "PinToQuickAccess"
        $This.DisplayName = "Pin to Quick Access"
        $This.Value       = 1
        $This.Description = "Adds a context menu item to pin an item to the Quick Access bar"
        $This.Options     = "Skip", "Enable*", "Disable"

        ('HKCR:\Folder\shell\pintohome',
        'MUIVerb'),
        ('HKCR:\Folder\shell\pintohome',
        'AppliesTo'),
        ('HKCR:\Folder\shell\pintohome\command',
        'DelegateExecute'),
        ('HKLM:\SOFTWARE\Classes\Folder\shell\pintohome',
        'MUIVerb'),
        ('HKLM:\SOFTWARE\Classes\Folder\shell\pintohome',
        'AppliesTo'),
        ('HKLM:\SOFTWARE\Classes\Folder\shell\pintohome\command',
        'DelegateExecute') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    [String] ParseName()
    {
        Return 'System.ParsingName:<>"::{679f85cb-0220-4080-b29b-5540cc05aab6}"',
               'System.ParsingName:<>"::{645FF040-5081-101B-9F08-00AA002F954E}"',
               'System.IsFolder:=System.StructuredQueryType.Boolean#True' -join " AND "
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] 'Pin to Quick Access' context menu item"
                }
            }
            1
            {
                Write-Host "Enabling [~] 'Pin to Quick Access' context menu item"
                $This.Output[0].Set("String",'@shell32.dll,-51377')
                $This.Output[1].Set("String",$This.ParseName())
                $This.Output[2].Set("String","{b455f46e-e4af-4035-b0a4-cf18d2f6f28e}")
                $This.Output[3].Set("String",'@shell32.dll,-51377')
                $This.Output[4].Set("String",$This.ParseName())
                $This.Output[5].Set("String","{b455f46e-e4af-4035-b0a4-cf18d2f6f28e}")
```

```
            }
            2
            {
                Write-Host "Disabling [~] 'Pin to Quick Access' context menu item"
                $This.Output[0].Name = $Null
                $This.Output[0].Remove()
                $This.Output[3].Name = $Null
                $This.Output[3].Remove()
            }
        }
    }
}
```

_____/
_____/ Class [PinToQuickAccess : ControlTemplate]
Class [ShareWith : ControlTemplate] /_____\
/_____

```
Class ShareWith : ControlTemplate
{
    ShareWith()
    {
        $This.Name        = "PinToQuickAccess"
        $This.DisplayName = "Pin to Quick Access"
        $This.Value       = 1
        $This.Description = "Adds a context menu item to share a file with..."
        $This.Options     = "Skip", "Enable*", "Disable"

        ('HKCR:\*\shellex\ContextMenuHandlers\Sharing',
         '(Default)'),
        ('HKCR:\Directory\shellex\ContextMenuHandlers\Sharing',
         '(Default)'),
        ('HKCR:\Directory\shellex\CopyHookHandlers\Sharing',
         '(Default)'),
        ('HKCR:\Drive\shellex\ContextMenuHandlers\Sharing',
         '(Default)'),
        ('HKCR:\Directory\shellex\PropertySheetHandlers\Sharing',
         '(Default)'),
        ('HKCR:\Directory\Background\shellex\ContextMenuHandlers\Sharing',
         '(Default)'),
        ('HKCR:\LibraryFolder\background\shellex\ContextMenuHandlers\Sharing',
         '(Default)'),
        ('HKCR:\*\shellex\ContextMenuHandlers\ModernSharing',
         '(Default)') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] 'Share with' context menu item"
                }
            }
            1
            {
                Write-Host "Enabling [~] 'Share with' context menu item"
                0..7 | % { $This.Output[$_].Set("String","{f81e9010-6ea4-11ce-a7ff-00aa003ca9f6}") }
            }
            2
            {
                Write-Host "Disabling [~] 'Share with' context menu item"
                0..7 | % { $This.Output[$_].Set("String","") }
            }
        }
    }
}
```

Class [ShareWith : ControlTemplate]

Class [SendTo : ControlTemplate]

```
Class SendTo : ControlTemplate
{
    SendTo()
    {
        $This.Name        = "SendTo"
        $This.DisplayName = "Send To"
        $This.Value       = 1
        $This.Description = "Adds a context menu item to send an item to..."
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry("HKCR:\AllFilesystemObjects\shellex\ContextMenuHandlers\SendTo","(Default)")
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] 'Send to' context menu item"
                }
            }
            1
            {
                Write-Host "Enabling [~] 'Send to' context menu item"
                $This.Output[0].Set("String","{7BA4C740-9E81-11CF-99D3-00AA004AE837}")
            }
            2
            {
                Write-Host "Disabling [~] 'Send to' context menu item"
                $This.Output[0].Name = $Null
                $This.Output[0].Remove()
            }
        }
    }
}
```

Class [SendTo : ControlTemplate]

Enum [ContextType]

```
Enum ContextType
{
    CastToDevice
    PreviousVersions
    IncludeInLibrary
    PinToStart
    PinToQuickAccess
    ShareWith
    SendTo
}
```

Enum [ContextType]

Class [ContextList]

```
Class ContextList
{
    [Object]  $Output
    ContextList()
    {
        $This.Output      = @( )
```

```
        ForEach ($Name in [System.Enum]::GetNames([ContextType]))
        {
            $Item         = Switch ($Name)
            {
                CastToDevice     { [CastToDevice]::New()     }
                PreviousVersions { [PreviousVersions]::New() }
                IncludeInLibrary { [IncludeInLibrary]::New() }
                PinToStart       { [PinToStart]::New()       }
                PinToQuickAccess { [PinToQuickAccess]::New() }
                ShareWith        { [ShareWith]::New()        }
                SendTo           { [SendTo]::New()           }
            }
            $Item.Source  = "Context"
            $This.Output += $Item
        }
    }
}
```

Class [ContextList]

Class [BatteryUIBar : ControlTemplate]

```
Class BatteryUIBar : ControlTemplate
{
    BatteryUIBar()
    {
        $This.Name        = "BatteryUIBar"
        $This.DisplayName = "Battery UI Bar"
        $This.Value       = 1
        $This.Description = "Toggles the battery UI bar element style"
        $This.Options     = "Skip", "New*", "Classic"

        $This.Registry('HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\ImmersiveShell','UseWin32BatteryFlyout')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Battery UI Bar"
                }
            }
            1
            {
                Write-Host "Setting [~] Battery UI Bar (New)"
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Setting [~] Battery UI Bar (Old)"
                $This.Output[0].Set(1)
            }
        }
    }
}
```

Class [BatteryUIBar : ControlTemplate]

Class [ClockUIBar : ControlTemplate]

```
Class ClockUIBar : ControlTemplate
{
    ClockUIBar()
    {
        $This.Name        = "ClockUIBar"
        $This.DisplayName = "Clock UI Bar"
```

```
        $This.Value       = 1
        $This.Description = "Toggles the clock UI bar element style"
        $This.Options     = "Skip", "New*", "Classic"

        ('HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\ImmersiveShell',
        'UseWin32TrayClockExperience') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Clock UI Bar"
                }
            }
            1
            {
                Write-Host "Setting [~] Clock UI Bar (New)"
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Setting [~] Clock UI Bar (Old)"
                $This.Output[0].Set(1)
            }
        }
    }
}
```

_____/
_____/ Class [ClockUIBar : ControlTemplate]
Class [VolumeControlBar : ControlTemplate] /_____\
/_____

```
Class VolumeControlBar : ControlTemplate
{
    VolumeControlBar()
    {
        $This.Name        = "VolumeControlBar"
        $This.DisplayName = "Volume Control Bar"
        $This.Value       = 1
        $This.Description = "Toggles the volume control bar element style"
        $This.Options     = "Skip", "New (X-Axis)*", "Classic (Y-Axis)"

        $This.Registry('HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\MTCUVC','EnableMtcUvc')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Volume Control Bar"
                }
            }
            1
            {
                Write-Host "Enabling [~] Volume Control Bar (Horizontal)"
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Volume Control Bar (Vertical)"
                $This.Output[0].Set(0)
```

```
            }
        }
    }
}
```

_____/
_____/ Class [VolumeControlBar : ControlTemplate]
Class [TaskBarSearchBox : ControlTemplate] /_____\
/_____

```
Class TaskBarSearchBox : ControlTemplate
{
    TaskBarSearchBox()
    {
        $This.Name        = "TaskBarSearchBox"
        $This.DisplayName = "Taskbar Search Box"
        $This.Value       = 1
        $This.Description = "Toggles the taskbar search box element"
        $This.Options     = "Skip", "Show*", "Hide"

        $This.Registry("HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Search","SearchboxTaskbarMode")
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Taskbar 'Search Box' button"
                }
            }
            1
            {
                Write-Host "Enabling [~] Taskbar 'Search Box' button"
                $This.Output[0].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Taskbar 'Search Box' button"
                $This.Output[0].Set(0)
            }
        }
    }
}
```

_____/
_____/ Class [TaskBarSearchBox : ControlTemplate]
Class [TaskViewButton : ControlTemplate] /_____\
/_____

```
Class TaskViewButton : ControlTemplate
{
    TaskViewButton()
    {
        $This.Name        = "VolumeControlBar"
        $This.DisplayName = "Volume Control Bar"
        $This.Value       = 1
        $This.Description = "Toggles the volume control bar element style"
        $This.Options     = "Skip", "New (X-Axis)*", "Classic (Y-Axis)"

        ('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced',
         'ShowTaskViewButton') | % {

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

    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
```

```
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Task View button"
                }
            }
            1
            {
                Write-Host "Enabling [~] Task View button"
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Task View button"
                $This.Output[0].Set(0)
            }
        }
    }
}
```

Class [TaskViewButton : ControlTemplate]

Class [TaskbarIconSize : ControlTemplate]

```
Class TaskbarIconSize : ControlTemplate
{
    TaskbarIconSize()
    {
        $This.Name        = "TaskbarIconSize"
        $This.DisplayName = "Taskbar Icon Size"
        $This.Value       = 1
        $This.Description = "Toggles the taskbar icon size"
        $This.Options     = "Skip", "Normal*", "Small"

        $This.Registry('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced','TaskbarSmallIcons')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Icon size in taskbar"
                }
            }
            1
            {
                Write-Host "Enabling [~] Icon size in taskbar"
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Icon size in taskbar"
                $This.Output[0].Set(1)
            }
        }
    }
}
```

Class [TaskbarIconSize : ControlTemplate]

Class [TaskbarGrouping : ControlTemplate]

```
Class TaskbarGrouping : ControlTemplate
```

```
{
    TaskbarGrouping()
    {
        $This.Name        = "TaskbarGrouping"
        $This.DisplayName = "Taskbar Grouping"
        $This.Value       = 2
        $This.Description = "Toggles the grouping of icons in the taskbar"
        $This.Options     = "Skip", "Never", "Always*","When needed"

        $This.Registry('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced','TaskbarGlomLevel')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Group Taskbar Items"
                }
            }
            1
            {
                Write-Host "Setting [~] Group Taskbar Items (Never)"
                $This.Output[0].Set(2)
            }
            2
            {
                Write-Host "Setting [~] Group Taskbar Items (Always)"
                $This.Output[0].Set(0)
            }
            3
            {
                Write-Host "Setting [~] Group Taskbar Items (When needed)"
                $This.Output[0].Set(1)
            }
        }
    }
}
```

_____/ _____/ Class [TaskbarGrouping : ControlTemplate] /_____\ Class [TrayIcons : ControlTemplate] /_____\ /_____

```
Class TrayIcons : ControlTemplate
{
    TrayIcons()
    {
        $This.Name        = "TrayIcons"
        $This.DisplayName = "Tray Icons"
        $This.Value       = 1
        $This.Description = "Toggles whether the tray icons are shown or hidden"
        $This.Options     = "Skip", "Auto*", "Always show"

        ('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer',
         'EnableAutoTray'),
        ('HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer',
         'EnableAutoTray') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
```

```
                    Write-Host "Skipping [!] Tray Icons"
                }
            }
            1
            {
                Write-Host "Setting [~] Tray Icons (Hiding)"
                $This.Output[0].Set(1)
                $This.Output[1].Set(1)
            }
            2
            {
                Write-Host "Setting [~] Tray Icons (Showing)"
                $This.Output[0].Set(0)
                $This.Output[1].Set(0)
            }
        }
    }
}
```

_____/
_____/ Class [TrayIcons : ControlTemplate]
Class [SecondsInClock : ControlTemplate] /_____\
/_____

```
Class SecondsInClock : ControlTemplate
{
    SecondsInClock()
    {
        $This.Name        = "SecondsInClock"
        $This.DisplayName = "Seconds in clock"
        $This.Value       = 1
        $This.Description = "Toggles the clock/time shows the seconds"
        $This.Options     = "Skip", "Show", "Hide*"

        ('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced',
         'ShowSecondsInSystemClock') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Seconds in Taskbar clock"
                }
            }
            1
            {
                Write-Host "Enabling [~] Seconds in Taskbar clock"
                $This.Output[0].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Seconds in Taskbar clock"
                $This.Output[0].Set(0)
            }
        }
    }
}
```

_____/
_____/ Class [SecondsInClock : ControlTemplate]
Class [LastActiveClick : ControlTemplate] /_____\
/_____

```
Class LastActiveClick : ControlTemplate
```

```
{
    LastActiveClick()
    {
        $This.Name        = "LastActiveClick"
        $This.DisplayName = "Last Active Click"
        $This.Value       = 2
        $This.Description = "Makes taskbar buttons open the last active window"
        $This.Options     = "Skip", "Enable", "Disable*"

        $This.Registry('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced','LastActiveClick')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Last active click"
                }
            }
            1
            {
                Write-Host "Enabling [~] Last active click"
                $This.Output[0].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Last active click"
                $This.Output[0].Set(0)
            }
        }
    }
}
```

Class [LastActiveClick : ControlTemplate]

Class [TaskbarOnMultiDisplay : ControlTemplate]

```
Class TaskbarOnMultiDisplay : ControlTemplate
{
    TaskbarOnMultiDisplay()
    {
        $This.Name        = "TaskbarOnMultiDisplay"
        $This.DisplayName = "Taskbar on multiple displays"
        $This.Value       = 1
        $This.Description = "Displays the taskbar on each display if there are multiple screens"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry('HKCU:\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced','MMTaskbarEnabled')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Taskbar on Multiple Displays"
                }
            }
            1
            {
                Write-Host "Enabling [~] Taskbar on Multiple Displays"
                $This.Output[0].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Taskbar on Multiple Displays"
```

```
                $This.Output[0].Set(0)
            }
        }
    }
}
```

Class [TaskbarOnMultiDisplay : ControlTemplate]

Class [TaskbarButtonDisplay : ControlTemplate]

```
Class TaskbarButtonDisplay : ControlTemplate
{
    TaskbarButtonDisplay()
    {
        $This.Name        = "TaskbarButtonDisplay"
        $This.DisplayName = "Multi-display taskbar"
        $This.Value       = 2
        $This.Description = "Defines where the taskbar button should be if there are multiple screens"
        $This.Options     = "Skip", "All", "Current Window*","Main + Current Window"

        $This.Registry('HKCU:\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced','MMTaskbarMode')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Taskbar buttons on multiple displays"
                }
            }
            1
            {
                Write-Host "Setting [~] Taskbar buttons, multi-display (All taskbars)"
                $This.Output[0].Set(0)
            }
            2
            {
                Write-Host "Setting [~] Taskbar buttons, multi-display (Taskbar where window is open)"
                $This.Output[0].Set(2)
            }
            3
            {
                Write-Host "Setting [~] Taskbar buttons, multi-display (Main taskbar + where window is open)"
                $This.Output[0].Set(1)
            }
        }
    }
}
```

Class [TaskbarButtonDisplay : ControlTemplate]

Enum [TaskbarType]

```
Enum TaskbarType
{
    BatteryUIBar
    ClockUIBar
    VolumeControlBar
    TaskbarSearchBox
    TaskViewButton
    TaskbarIconSize
    TaskbarGrouping
    TrayIconsSecondsInClock
    LastActiveClick
}
```

Class [TaskbarList]

```
Class TaskbarList
{
    [Object]   $Output
    TaskbarList()
    {
        $This.Output       = @( )
        ForEach ($Name in [System.Enum]::GetNames([TaskbarType]))
        {
            $Item          = Switch ($Name)
            {
                BatteryUIBar     { [BatteryUIBar]::New()     }
                ClockUIBar       { [ClockUIBar]::New()       }
                VolumeControlBar { [VolumeControlBar]::New() }
                TaskbarSearchBox { [TaskbarSearchBox]::New() }
                TaskViewButton   { [TaskViewButton]::New()   }
                TaskbarIconSize  { [TaskbarIconSize]::New()  }
                TaskbarGrouping  { [TaskbarGrouping]::New()  }
                TrayIcons        { [TrayIcons]::New()        }
            }
            $This.Output += $Item
        }
        $This.Output | % { $_.Source = "Taskbar" }
    }
}
```

Class [StartMenuWebSearch : ControlTemplate]

```
Class StartMenuWebSearch : ControlTemplate
{
    StartMenuWebSearch()
    {
        $This.Name        = "StartMenuWebSearch"
        $This.DisplayName = "Start Menu Web Search"
        $This.Value       = 1
        $This.Description = "Allows the start menu search box to search the internet"
        $This.Options     = "Skip", "Enable*", "Disable"

        ('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Search',
         'BingSearchEnabled'),
        ('HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Search',
         'DisableWebSearch') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Bing Search in Start Menu"
                }
            }
            1
            {
                Write-Host "Enabling [~] Bing Search in Start Menu"
                $This.Output[0].Remove()
                $This.Output[1].Remove()
            }
            2
            {
```

```
                Write-Host "Disabling [~] Bing Search in Start Menu"
                $This.Output[0].Set(0)
                $This.Output[1].Set(1)
            }
        }
    }
}
```

Class [StartSuggestions : ControlTemplate] Class [StartMenuWebSearch : ControlTemplate]

```
Class StartSuggestions : ControlTemplate
{
    StartSuggestions()
    {
        $This.Name        = "StartSuggestions"
        $This.DisplayName = "Start Menu Suggestions"
        $This.Value       = 1
        $This.Description = "Toggles the suggested apps in the start menu"
        $This.Options     = "Skip", "Enable*", "Disable"

        ($This.RegPath(),"ContentDeliveryAllowed"),
        ($This.RegPath(),"OemPreInstalledAppsEnabled"),
        ($This.RegPath(),"PreInstalledAppsEnabled"),
        ($This.RegPath(),"PreInstalledAppsEverEnabled"),
        ($This.RegPath(),"SilentInstalledAppsEnabled"),
        ($This.RegPath(),"SystemPaneSuggestionsEnabled"),
        ($This.RegPath(),"Start_TrackProgs"),
        ($This.RegPath(),"SubscribedContent-314559Enabled"),
        ($This.RegPath(),"SubscribedContent-310093Enabled"),
        ($This.RegPath(),"SubscribedContent-338387Enabled"),
        ($This.RegPath(),"SubscribedContent-338388Enabled"),
        ($This.RegPath(),"SubscribedContent-338389Enabled"),
        ($This.RegPath(),"SubscribedContent-338393Enabled"),
        ($This.RegPath(),"SubscribedContent-338394Enabled"),
        ($This.RegPath(),"SubscribedContent-338396Enabled"),
        ($This.RegPath(),"SubscribedContent-338398Enabled") | % {

            $This.Registry($_[0],$_[1])
        }
    }
    [String] RegPath()
    {
        Return "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\ContentDeliveryManager"
    }
    [String] CloudCache()
    {
        Return "HKCU:","SOFTWARE","Microsoft","Windows","CurrentVersion","CloudStore","Store",
        "Cache","DefaultAccount","*windows.data.placeholdertilecollection","Current" -join '\'
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Start Menu Suggestions"
                }
            }
            1
            {
                Write-Host "Enabling [~] Start Menu Suggestions"
                0..15 | % { $This.Output[$_].Set(1) }
            }
            2
            {
                Write-Host "Disabling [~] Start Menu Suggestions"
                0..15 | % { $This.Output[$_].Set(0) }
```

```
                If ($This.GetWinVersion() -ge 1803)
                {
                    $Key = Get-ItemProperty -Path $This.CloudCache()
                    Set-ItemProperty -Path $Key.PSPath -Name Data -Type Binary -Value $Key.Data[0..15]
                    Stop-Process -Name ShellExperienceHost -Force
                }
            }
        }
    }
}
```

Class [StartSuggestions : ControlTemplate]

Class [MostUsedAppStartMenu : ControlTemplate]

```
Class MostUsedAppStartMenu : ControlTemplate
{
    MostUsedAppStartMenu()
    {
        $This.Name        = "MostUsedAppStartMenu"
        $This.DisplayName = "Most Used Applications"
        $This.Value       = 1
        $This.Description = "Toggles the most used applications in the start menu"
        $This.Options     = "Skip", "Show*", "Hide"

        $This.Registry('HKCU:\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced','Start_TrackProgs')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Most used apps in Start Menu"
                }
            }
            1
            {
                Write-Host "Enabling [~] Most used apps in Start Menu"
                $This.Output[0].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Most used apps in Start Menu"
                $This.Output[0].Set(0)
            }
        }
    }
}
```

Class [MostUsedAppStartMenu : ControlTemplate]

Class [RecentItemsFrequent : ControlTemplate]

```
Class RecentItemsFrequent : ControlTemplate
{
    RecentItemsFrequent()
    {
        $This.Name        = "RecentItemsFrequent"
        $This.DisplayName = "Recent Items Frequent"
        $This.Value       = 1
        $This.Description = "Toggles the most recent frequently used (apps/items) in the start menu"
        $This.Options     = "Skip", "Enable*", "Disable"

        ('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\HideDesktopIcons\ClassicStartMenu',
        "Start_TrackDocs") | % {
```

```
            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Recent items and frequent places"
                }
            }
            1
            {
                Write-Host "Enabling [~] Recent items and frequent places"
                $This.Output[0].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Recent items and frequent places"
                $This.Output[0].Set(0)
            }
        }
    }
}
```

_____/ Class [RecentItemsFrequent : ControlTemplate] /_____\

Class [UnpinItems : ControlTemplate] /_____/

```
Class UnpinItems : ControlTemplate
{
    UnpinItems()
    {
        $This.Name        = "UnpinItems"
        $This.DisplayName = "Unpin Items"
        $This.Value       = 0
        $This.Description = "Toggles the unpin (apps/items) from the start menu"
        $This.Options     = "Skip", "Enable"
    }
    [String] RegPath()
    {
        Return "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\CloudStore\Store\Cache\DefaultAccount"
    }
    [String] Collection()
    {
        Return "*start.tilegrid`$windows.data.curatedtilecollection.tilecollection\Current"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Unpinning Items"
                }
            }
            1
            {
                Write-Host "Enabling [~] Unpinning Items"
                If ($This.GetWinVersion() -le 1709)
                {
                    ForEach ($Item in Get-ChildItem $This.RegPath() -Include *.group -Recurse)
                    {
                        $Path = "{0}\Current" -f $Item.PsPath
                        $Data = (Get-ItemProperty $Path -Name Data).Data -join ","
                        $Data = $Data.Substring(0, $Data.IndexOf(",0,202,30") + 9) + ",0,202,80,0,0"
```

```
                        Set-ItemProperty $Path -Name Data -Type Binary -Value $Data.Split(",")
                    }
                }
                Else
                {
                    $Key     = Get-ItemProperty -Path "$($This.RegPath())\$($This.Collection())"
                    $Data    = $Key.Data[0..25] + ([Byte[]](202,50,0,226,44,1,1,0,0))
                    Set-ItemProperty -Path $Key.PSPath -Name Data -Type Binary -Value $Data
                    Stop-Process -Name ShellExperienceHost -Force
                }
            }
        }
    }
}
```

Enum [StartMenuType]

```
Enum StartMenuType
{
    StartMenuWebSearch
    StartSuggestions
    MostUsedAppStartMenu
    RecentItemsFrequent
    UnpinItems
}
```

Class [StartMenuList]

```
Class StartMenuList
{
    [Object]  $Output
    StartMenuList()
    {
        $This.Output      = @( )
        ForEach ($Name in [System.Enum]::GetNames([StartMenuType]))
        {
            $Item         = Switch ($Name)
            {
                StartMenuWebSearch   { [StartMenuWebSearch]::New()   }
                StartSuggestions     { [StartSuggestions]::New()     }
                MostUsedAppStartMenu { [MostUsedAppStartMenu]::New() }
                RecentItemsFrequent  { [RecentItemsFrequent]::New()  }
                UnpinItems           { [UnpinItems]::New()           }
            }
            $Item.Source  = "StartMenu"
            $This.Output += $Item
        }
    }
}
```

Class [AccessKeyPrompt : ControlTemplate]

```
Class AccessKeyPrompt : ControlTemplate
{
    AccessKeyPrompt()
    {
        $This.Name        = "AccessKeyPrompt"
        $This.DisplayName = "Access Key Prompt"
        $This.Value       = 1
        $This.Description = "Toggles the accessibility keys (menus/prompts)"
        $This.Options     = "Skip", "Enable*", "Disable"
```

```
        ('HKCU:\Control Panel\Accessibility\StickyKeys',
         "Flags"),
        ('HKCU:\Control Panel\Accessibility\ToggleKeys',
         "Flags"),
        ('HKCU:\Control Panel\Accessibility\Keyboard Response',
         "Flags") | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Accessibility keys prompts"
                }
            }
            1
            {
                Write-Host "Enabling [~] Accessibility keys prompts"
                $This.Output[0].Set("String",510)
                $This.Output[1].Set("String",62)
                $This.Output[2].Set("String",126)
            }
            2
            {
                Write-Host "Disabling [~] Accessibility keys prompts"
                $This.Output[0].Set("String",506)
                $This.Output[1].Set("String",58)
                $This.Output[2].Set("String",122)
            }
        }
    }
}
```

Class [F1HelpKey : ControlTemplate] Class [AccessKeyPrompt : ControlTemplate]

```
Class F1HelpKey : ControlTemplate
{
    F1HelpKey()
    {
        $This.Name        = "F1HelpKey"
        $This.DisplayName = "F1 Help Key"
        $This.Value       = 1
        $This.Description = "Toggles the F1 help menu/prompt"
        $This.Options     = "Skip", "Enable*", "Disable"

        ($This.RegPath(),
        $Null),
        ("$($This.RegPath()))\win32",
        "(Default)"),
        ("$($This.RegPath())\win64",
        "(Default)") | % {

            $This.Registry($_[0],$_[1])
        }
    }
    [String] RegPath()
    {
        Return "HKCU:\Software\Classes\TypeLib\{8cec5860-07a1-11d9-b15e-000d56bfe6ee}\1.0\0"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
```

```
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] F1 Help Key"
                }
            }
            1
            {
                Write-Host "Enabling [~] F1 Help Key"
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Disabling [~] F1 Help Key"
                $This.Output[1].Set("String","")
                If ([Environment]::Is64BitOperatingSystem)
                {
                $This.Output[2].Set("String","")
                }
            }
        }
    }
}
```

```
Class AutoPlay : ControlTemplate
{
    AutoPlay()
    {
        $This.Name        = "AutoPlay"
        $This.DisplayName = "AutoPlay"
        $This.Value       = 1
        $This.Description = "Toggles autoplay for inserted discs or drives"
        $This.Options     = "Skip", "Enable*", "Disable"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Autoplay"
                }
            }
            1
            {
                Write-Host "Enabling [~] Autoplay"
                $This.Output[0].Set(0)
            }
            2
            {
                Write-Host "Disabling [~] Autoplay"
                $This.Output[0].Set(1)
            }
        }
    }
}
```

```
Class AutoRun : ControlTemplate
{
    AutoRun()
    {
        $This.Name        = "AutoRun"
        $This.DisplayName = "AutoRun"
        $This.Value       = 1
        $This.Description = "Toggles autorun for programs on an inserted discs or drives"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer','NoDriveTypeAutoRun')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Autorun"
                }
            }
            1
            {
                Write-Host "Enabling [~] Autorun"
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Autorun"
                $This.Output[0].Set(255)
            }
        }
    }
}
```

Class [AutoRun : ControlTemplate]

Class [PidInTitleBar : ControlTemplate]

```
Class PidInTitleBar : ControlTemplate
{
    PidInTitleBar()
    {
        $This.Name        = "PidInTitleBar"
        $This.DisplayName = "Process ID"
        $This.Value       = 2
        $This.Description = "Toggles the process ID in a window title bar"
        $This.Options     = "Skip", "Show", "Hide*"

        $This.Registry('HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer','ShowPidInTitle')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Process ID on Title bar"
                }
            }
            1
            {
                Write-Host "Enabling [~] Process ID on Title bar"
                $This.Output[0].Set(1)
            }
            2
            {
```

```
                Write-Host "Disabling [~] Process ID on Title bar"
                $This.Output[0].Remove()
            }
        }
    }
}
```

_____/
_____/ Class [PidInTitleBar : ControlTemplate]
Class [AeroSnap : ControlTemplate] /_____\
/_____

```
Class AeroSnap : ControlTemplate
{
    AeroSnap()
    {
        $This.Name        = "AeroSnap"
        $This.DisplayName = "AeroSnap"
        $This.Value       = 1
        $This.Description = "Toggles the ability to snap windows to the sides of the screen"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry('HKCU:\Control Panel\Desktop','WindowArrangementActive')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Aero Snap"
                }
            }
            1
            {
                Write-Host "Enabling [~] Aero Snap"
                $This.Output[0].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Aero Snap"
                $This.Output[0].Set(0)
            }
        }
    }
}
```

_____/
_____/ Class [AeroSnap : ControlTemplate]
Class [AeroShake : ControlTemplate] /_____\
/_____

```
Class AeroShake : ControlTemplate
{
    AeroShake()
    {
        $This.Name        = "AeroShake"
        $This.DisplayName = "AeroShake"
        $This.Value       = 1
        $This.Description = "Toggles ability to minimize ALL windows by jiggling the active window title bar"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry('HKCU:\Software\Policies\Microsoft\Windows\Explorer','NoWindowMinimizingShortcuts')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
```

```
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Aero Shake"
                }
            }
            1
            {
                Write-Host "Enabling [~] Aero Shake"
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Aero Shake"
                $This.Output[0].Set(1)
            }
        }
    }
}
```

_____/
_____/ Class [AeroShake : ControlTemplate]
Class [KnownExtensions : ControlTemplate] /_____\
/_____

```
Class KnownExtensions : ControlTemplate
{
    KnownExtensions()
    {
        $This.Name        = "KnownExtensions"
        $This.DisplayName = "Known File Extensions"
        $This.Value       = 2
        $This.Description = "Shows known (mime-types/file extensions)"
        $This.Options     = "Skip", "Show", "Hide*"

        $This.Registry('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced','HideFileExt')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Known File Extensions"
                }
            }
            1
            {
                Write-Host "Enabling [~] Known File Extensions"
                $This.Output[0].Set(0)
            }
            2
            {
                Write-Host "Disabling [~] Known File Extensions"
                $This.Output[0].Set(1)
            }
        }
    }
}
```

_____/
_____/ Class [KnownExtensions : ControlTemplate]
Class [HiddenFiles : ControlTemplate] /_____\
/_____

```
Class HiddenFiles : ControlTemplate
{
    HiddenFiles()
    {
```

```
        $This.Name        = "HiddenFiles"
        $This.DisplayName = "Show Hidden Files"
        $This.Value       = 2
        $This.Description = "Shows all hidden files"
        $This.Options     = "Skip", "Show", "Hide*"

        $This.Registry('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced','Hidden')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Hidden Files"
                }
            }
            1
            {
                Write-Host "Enabling [~] Hidden Files"
                $This.Output[0].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Hidden Files"
                $This.Output[0].Set(2)
            }
        }
    }
}
```

Class [SystemFiles : ControlTemplate] Class [HiddenFiles : ControlTemplate]

```
Class SystemFiles : ControlTemplate
{
    SystemFiles()
    {
        $This.Name        = "SystemFiles"
        $This.DisplayName = "Show System Files"
        $This.Value       = 2
        $This.Description = "Shows all system files"
        $This.Options     = "Skip", "Show", "Hide*"

        $This.Registry('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced','ShowSuperHidden')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] System Files"
                }
            }
            1
            {
                Write-Host "Enabling [~] System Files"
                $This.Output[0].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] System Files"
                $This.Output[0].Set(0)
            }
        }
```

```
    }
}
```

_____/
_____/ Class [SystemFiles : ControlTemplate]
Class [ExplorerOpenLoc : ControlTemplate] /_____\
/_____

```
Class ExplorerOpenLoc : ControlTemplate
{
    ExplorerOpenLoc()
    {
        $This.Name        = "ExplorerOpenLoc"
        $This.DisplayName = "Explorer Open Location"
        $This.Value       = 1
        $This.Description = "Default path/location opened with a new explorer window"
        $This.Options     = "Skip", "Quick Access*", "This PC"

        $This.Registry('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced','LaunchTo')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Default Explorer view to Quick Access"
                }
            }
            1
            {
                Write-Host "Enabling [~] Default Explorer view to Quick Access"
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Default Explorer view to Quick Access"
                $This.Output[0].Set(1)
            }
        }
    }
}
```

_____/
_____/ Class [ExplorerOpenLoc : ControlTemplate]
Class [RecentFileQuickAccess : ControlTemplate] /_____\
/_____

```
Class RecentFileQuickAccess : ControlTemplate
{
    RecentFileQuickAccess()
    {
        $This.Name        = "RecentFileQuickAccess"
        $This.DisplayName = "Recent File Quick Access"
        $This.Value       = 1
        $This.Description = "Shows recent files in the Quick Access menu"
        $This.Options     = "Skip", "Show/Add*", "Hide", "Remove"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Recent Files in Quick Access"
                }
            }
```

```
            1
            {
                Write-Host "Setting [~] Recent Files in Quick Access (Showing)"
                $This.Output[0].Set(1)
                $This.Output[1].Set("String","Recent Items Instance Folder")
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[2].Set("String","Recent Items Instance Folder")
                }
            }
            2
            {
                Write-Host "Setting [~] Recent Files in Quick Access (Hiding)"
                $This.Output[0].Set(0)
            }
            3
            {
                Write-Host "Setting [~] Recent Files in Quick Access (Removing)"
                $This.Output[0].Set(0)
                $This.Output[1].Remove()
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[2].Remove()
                }
            }
        }
    }
}
```

Class [FrequentFoldersQuickAccess : ControlTemplate]

```
Class FrequentFoldersQuickAccess : ControlTemplate
{
    FrequentFoldersQuickAccess()
    {
        $This.Name        = "FrequentFoldersQuickAccess"
        $This.DisplayName = "Frequent Folders Quick Access"
        $This.Value       = 1
        $This.Description = "Show frequently used folders in the Quick Access menu"
        $This.Options     = "Skip", "Show*", "Hide"

        $This.Registry('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer','ShowFrequent')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Frequent folders in Quick Access"
                }
            }
            1
            {
                Write-Host "Enabling [~] Frequent folders in Quick Access"
                $This.Output[0].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Frequent folders in Quick Access"
                $This.Output[0].Set(0)
            }
        }
    }
}
```

_____/ Class [FrequentFoldersQuickAccess : ControlTemplate]
Class [WinContentWhileDrag : ControlTemplate] /----------\
/----------

```
Class WinContentWhileDrag : ControlTemplate
{
    WinContentWhileDrag()
    {
        $This.Name        = "WinContentWhileDrag"
        $This.DisplayName = "Window Content while dragging"
        $This.Value       = 1
        $This.Description = "Show the content of a window while it is being dragged/moved"
        $This.Options     = "Skip", "Show*", "Hide"

        $This.Registry('HKCU:\Control Panel\Desktop','DragFullWindows')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Window content while dragging"
                }
            }
            1
            {
                Write-Host "Enabling [~] Window content while dragging"
                $This.Output[0].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Window content while dragging"
                $This.Output[0].Set(0)
            }
        }
    }
}
```

_____/ Class [WinContentWhileDrag : ControlTemplate]
Class [StoreOpenWith : ControlTemplate] /----------\
/----------

```
Class StoreOpenWith : ControlTemplate
{
    StoreOpenWith()
    {
        $This.Name        = "StoreOpenWith"
        $This.DisplayName = "Store Open With..."
        $This.Value       = 1
        $This.Description = "Toggles the ability to use the Microsoft Store to open an unknown file/program"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry('HKLM:\SOFTWARE\Policies\Microsoft\Windows\Explorer','NoUseStoreOpenWith')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Search Windows Store for Unknown Extensions"
                }
            }
            1
            {
                Write-Host "Enabling [~] Search Windows Store for Unknown Extensions"
```

```
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Search Windows Store for Unknown Extensions"
                $This.Output[0].Set(1)
            }
        }
    }
}
```

_____/
_____/ Class [StoreOpenWith : ControlTemplate]
Class [WinXPowerShell : ControlTemplate] /_____\
/_____

```
Class WinXPowerShell : ControlTemplate
{
    WinXPowerShell()
    {
        $This.Name        = "WinXPowerShell"
        $This.DisplayName = "Win X PowerShell"
        $This.Value       = 1
        $This.Description = "Toggles whether (Win + X) opens PowerShell or a Command Prompt"
        $This.Options     = "Skip", "PowerShell*", "Command Prompt"

        ('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced',
        'DontUsePowerShellOnWinX') | % {

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

        If ($This.GetWinVersion() -lt 1703)
        {
            $This.Value   = 2
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] (Win+X) PowerShell/Command Prompt"
                }
            }
            1
            {
                Write-Host "Enabling [~] (Win+X) PowerShell/Command Prompt"
                $This.Output[0].Set(0)
            }
            2
            {
                Write-Host "Disabling [~] (Win+X) PowerShell/Command Prompt"
                $This.Output[0].Set(1)
            }
        }
    }
}
```

_____/
_____/ Class [WinXPowerShell : ControlTemplate]
Class [TaskManagerDetails : ControlTemplate] /_____\
/_____

```
Class TaskManagerDetails : ControlTemplate
{
    TaskManagerDetails()
    {
```

```
        $This.Name        = "TaskManagerDetails"
        $This.DisplayName = "Task Manager Details"
        $This.Value       = 2
        $This.Description = "Toggles whether the task manager details are shown"
        $This.Options     = "Skip", "Show", "Hide*"

        $This.Registry('HKCU:\Software\Microsoft\Windows\CurrentVersion\TaskManager',"Preferences")
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Task Manager Details"
                }
            }
            1
            {
                Write-Host "Enabling [~] Task Manager Details"
                $Path         = $This.Output[0].Path
                $Task         = Start-Process -WindowStyle Hidden -FilePath taskmgr.exe -PassThru
                $Collect      = @( )
                $Timeout      = 0
                $TM           = $Null
                Do
                {
                    Start-Sleep -Milliseconds 100
                    $TM       = Get-ItemProperty -Path $Path | % Preferences
                    $Collect += 100
                    $TimeOut  = $Collect -join "+" | Invoke-Expression
                }
                Until ($TM -or $Timeout -ge 30000)
                Stop-Process $Task
                $TM[28]       = 0
                $This.Output[0].Set("Binary",$TM)
            }
            2
            {
                Write-Host "Disabling [~] Task Manager Details"
                $TM           = $This.Output[0].Get().Preferences
                $TM[28]       = 1
                $This.Output[0].Set("Binary",$TM)
            }
        }
    }
}
```

Class [ReopenAppsOnBoot : ControlTemplate]

Class [TaskManagerDetails : ControlTemplate]

```
Class ReopenAppsOnBoot : ControlTemplate
{
    ReopenAppsOnBoot()
    {
        $This.Name        = "ReopenAppsOnBoot"
        $This.DisplayName = "Reopen apps at boot"
        $This.Value       = 1
        $This.Description = "Toggles applications to reopen at boot time"
        $This.Options     = "Skip", "Enable*", "Disable"

        ('HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System',
        'DisableAutomaticRestartSignOn') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
```

```
    {
        If ($This.GetWinVersion() -eq 1709)
        {
            Switch ($Mode)
            {
                0
                {
                    If ($ShowSkipped)
                    {
                        Write-Host "Skipping [!] Reopen applications at boot time"
                    }
                }
                1
                {
                    Write-Host "Enabling [~] Reopen applications at boot time"
                    $This.Output[0].Set(0)
                }
                2
                {
                    Write-Host "Disabling [~] Reopen applications at boot time"
                    $This.Output[0].Set(1)
                }
            }
        }
    }
}
```

Class [Timeline : ControlTemplate]

Class [ReopenAppsOnBoot : ControlTemplate]

```
Class Timeline : ControlTemplate
{
    Timeline()
    {
        $This.Name        = "Timeline"
        $This.DisplayName = "Timeline"
        $This.Value       = 1
        $This.Description = "Toggles Windows Timeline, for recovery of items at a prior point in time"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry('HKLM:\SOFTWARE\Policies\Microsoft\Windows\System','EnableActivityFeed')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        If ($This.GetWinVersion())
        {
            Switch ($Mode)
            {
                0
                {
                    If ($ShowSkipped)
                    {
                        Write-Host "Skipping [!] Windows Timeline"
                    }
                }
                1
                {
                    Write-Host "Enabling [~] Windows Timeline"
                    $This.Output[0].Set(1)
                }
                2
                {
                    Write-Host "Disabling [~] Windows Timeline"
                    $This.Output[0].Set(0)
                }
            }
        }
    }
}
```

Class [LongFilePath : ControlTemplate]

```
Class LongFilePath : ControlTemplate
{
    LongFilePath()
    {
        $This.Name        = "LongFilePath"
        $This.DisplayName = "Long File Path"
        $This.Value       = 1
        $This.Description = "Toggles whether file paths are longer, or not"
        $This.Options     = "Skip", "Enable", "Disable*"

        ('HKLM:\SYSTEM\CurrentControlSet\Control\FileSystem',
        'LongPathsEnabled'),
        ('HKLM:\SYSTEM\ControlSet001\Control\FileSystem',
        'LongPathsEnabled') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Long file path"
                }
            }
            1
            {
                Write-Host "Enabling [~] Long file path"
                $This.Output[0].Set(1)
                $This.Output[1].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Long file path"
                $This.Output[0].Remove()
                $This.Output[1].Remove()
            }
        }
    }
}
```

Class [AppHibernationFile : ControlTemplate]

```
Class AppHibernationFile : ControlTemplate
{
    AppHibernationFile()
    {
        $This.Name        = "AppHibernationFile"
        $This.DisplayName = "App Hibernation File"
        $This.Value       = 1
        $This.Description = "Toggles the system swap file use"
        $This.Options     = "Skip", "Enable*", "Disable"

        ("HKLM:\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management",
        "SwapfileControl") | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
```

```
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] App Hibernation File (swapfile.sys)"
                }
            }
            1
            {
                Write-Host "Enabling [~] App Hibernation File (swapfile.sys)"
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Disabling [~] App Hibernation File (swapfile.sys)"
                $This.Output[0].Set(0)
            }
        }
    }
}
```

Class [AppHibernationFile : ControlTemplate]

Enum [ExplorerType]

```
Enum ExplorerType
{
    AccessKeyPrompt
    F1HelpKey
    AutoPlay
    AutoRun
    PidInTitleBar
    RecentFileQuickAccess
    FrequentFoldersQuickAccess
    WinContentWhileDrag
    StoreOpenWith
    LongFilePath
    ExplorerOpenLoc
    WinXPowerShell
    AppHibernationFile
    Timeline
    AeroSnap
    AeroShake
    KnownExtensions
    HiddenFiles
    SystemFiles
    TaskManager
    ReopenApps
}
```

Enum [ExplorerType]

Class [ExplorerList]

```
Class ExplorerList
{
    [Object]  $Output
    ExplorerList()
    {
        $This.Output = @( )
        ForEach ($Name in [System.Enum]::GetNames([ExplorerType]))
        {
            $Item = Switch ($Name)
            {
                AccessKeyPrompt            { [AccessKeyPrompt]::New()            }
                F1HelpKey                  { [F1HelpKey]::New()                  }
```

```
                AutoPlay                   { [AutoPlay]::New()                   }
                AutoRun                    { [AutoRun]::New()                    }
                PidInTitleBar              { [PidInTitleBar]::New()              }
                RecentFileQuickAccess      { [RecentFileQuickAccess]::New()      }
                FrequentFoldersQuickAccess { [FrequentFoldersQuickAccess]::New() }
                WinContentWhileDrag        { [WinContentWhileDrag]::New()        }
                StoreOpenWith              { [StoreOpenWith]::New()              }
                LongFilePath               { [LongFilePath]::New()               }
                ExplorerOpenLoc            { [ExplorerOpenLoc]::New()            }
                WinXPowerShell             { [WinXPowerShell]::New()             }
                AppHibernationFile         { [AppHibernationFile]::New()         }
                Timeline                   { [Timeline]::New()                   }
                AeroSnap                   { [AeroSnap]::New()                   }
                AeroShake                  { [AeroShake]::New()                  }
                KnownExtensions            { [KnownExtensions]::New()            }
                HiddenFiles                { [HiddenFiles]::New()                }
                SystemFiles                { [SystemFiles]::New()                }
                TaskManagerDetails         { [TaskManagerDetails]::New()         }
                ReopenAppsOnBoot           { [ReopenAppsOnBoot]::New()           }
            }
            $This.Output += $Item
        }

        $This.Output | % { $_.Source = "Explorer" }
    }
}
```

Class [ExplorerList]

Class [DesktopIconInThisPC : ControlTemplate]

```
Class DesktopIconInThisPC : ControlTemplate
{
    DesktopIconInThisPC()
    {
        $This.Name        = "DesktopIconInThisPC"
        $This.DisplayName = "Desktop [Explorer]"
        $This.Value       = 1
        $This.Description = "Toggles the Desktop icon in 'This PC'"
        $This.Options     = "Skip", "Show/Add*", "Hide", "Remove"

        ForEach ($X in 0,1)
        {
            ($This.Path($X),$Null),
            ("$($This.Path($X))\PropertyBag",$Null),
            ("$($This.Path($X))\PropertyBag","ThisPCPolicy") | % {

                $This.Registry($_[0],$_[1])
            }
        }
    }
    [String] Path([UInt32]$Slot)
    {
        Return "HKLM:","SOFTWARE",@($Null,"WOW6432Node")[$Slot],"Microsoft","Windows","CurrentVersion",
        "Explorer","FolderDescriptions","{B4BFCC3A-DB2C-424C-B029-7FE99A87C641}" -join "\"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Desktop folder in This PC"
                }
            }
            1
            {
                Write-Host "Enabling [~] Desktop folder in This PC (Shown)"
                $This.Output[0].Get()
```

```
                $This.Output[1].Get()
                $This.Output[2].Set("String","Show")
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[3].Get()
                    $This.Output[4].Get()
                    $This.Output[5].Set("String","Show")
                }
            }
            2
            {
                Write-Host "Setting [~] Desktop folder in This PC (Hidden)"
                $This.Output[2].Set("String","Hide")
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[5].Set("String","Hide")
                }
            }
            3
            {
                Write-Host "Setting [~] Desktop folder in This PC (None)"
                $This.Output[1].Remove()
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[5].Remove()
                }
            }
        }
    }
}
```

_____/ Class [DesktopIconInThisPC : ControlTemplate] /_____\
Class [DocumentsIconInThisPC : ControlTemplate] /_____/

```
Class DocumentsIconInThisPC : ControlTemplate
{
    DocumentsIconInThisPC()
    {
        $This.Name        = "DocumentsIconInThisPC"
        $This.DisplayName = "Documents [Explorer]"
        $This.Value       = 1
        $This.Description = "Toggles the Documents icon in 'This PC'"
        $This.Options     = "Skip", "Show/Add*", "Hide", "Remove"

        ForEach ($X in 0,1)
        {
            ($This.Path($X,0,0),$Null),
            ($This.Path($X,0,1),$Null),
            ("$($This.Path($X,1,2))\PropertyBag",$Null),
            ("$($This.Path($X,1,2))\PropertyBag","ThisPCPolicy"),
            ("$($This.Path($X,1,2))\PropertyBag","BaseFolderID") | % {

                $This.Registry($_[0],$_[1])
            }
        }
    }
    [String] Path([UInt32]$Slot,[UInt32]$Base,[UInt32]$Guid)
    {
        Return "HKLM:","SOFTWARE",@($Null,"WOW6432Node")[$Slot],"Microsoft","Windows",
        "CurrentVersion","Explorer",@("MyComputer\NameSpace","FolderDescriptions")[$Base],
        $This.Guid($Guid) -join "\"
    }
    [String] Guid([UInt32]$Slot)
    {
        Return @("{A8CDFF1C-4878-43be-B5FD-F8091C1C60D0}",
                 "{d3162b92-9365-467a-956b-92703aca08af}",
                 "{f42ee2d3-909f-4907-8871-4c22fc0bf756}")[$Slot]
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
```

```
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Documents folder in This PC"
                }
            }
            1
            {
                Write-Host "Enabling [~] Documents folder in This PC (Shown)"
                $This.Output[0].Get()
                $This.Output[1].Get()
                $This.Output[2].Get()
                $This.Output[3].Set("String","Show")
                $This.Output[4].Set("String","{FDD39AD0-238F-46AF-ADB4-6C85480369C7}")
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[5].Get()
                    $This.Output[6].Get()
                    $This.Output[7].Get()
                    $This.Output[8].Set("String","Show")
                }
            }
            2
            {
                Write-Host "Setting [~] Documents folder in This PC (Hidden)"
                $This.Output[3].Set("String","Hide")
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[8].Set("String","Hide")
                }
            }
            3
            {
                Write-Host "Setting [~] Documents folder in This PC (None)"
                $This.Output[0].Remove()
                $This.Output[1].Remove()
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[5].Remove()
                    $This.Output[6].Remove()
                }
            }
        }
    }
}
```

_____/ Class [DocumentsIconInThisPC : ControlTemplate]
_____/
Class [DownloadsIconInThisPC : ControlTemplate] /_____\
/_____

```
Class DownloadsIconInThisPC : ControlTemplate
{
    DownloadsIconInThisPC()
    {
        $This.Name        = "DownloadsIconInThisPC"
        $This.DisplayName = "Downloads [Explorer]"
        $This.Value       = 1
        $This.Description = "Toggles the Downloads icon in 'This PC'"
        $This.Options     = "Skip", "Show/Add*", "Hide", "Remove"

        ForEach ($X in 0,1)
        {
            ($This.Path($X,0,0),$Null),
            ($This.Path($X,0,1),$Null),
            ("$($This.Path($X,1,2))\PropertyBag",$Null),
            ("$($This.Path($X,1,2))\PropertyBag","ThisPCPolicy"),
            ("$($This.Path($X,1,2))\PropertyBag","BaseFolderID") | % {
```

```
                $This.Registry($_[0],$_[1])
            }
        }
    }
    [String] Path([UInt32]$Slot,[UInt32]$Base,[UInt32]$Guid)
    {
        Return "HKLM:","SOFTWARE",@($Null,"WOW6432Node")[$Slot],"Microsoft","Windows",
        "CurrentVersion","Explorer",@("MyComputer\NameSpace","FolderDescriptions")[$Base],
        $This.Guid($Guid) -join "\"
    }
    [String] Guid([UInt32]$Slot)
    {
        Return @("{374DE290-123F-4565-9164-39C4925E467B}",
                 "{088e3905-0323-4b02-9826-5d99428e115f}",
                 "{7d83ee9b-2244-4e70-b1f5-5393042af1e4}")[$Slot]
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Downloads folder in This PC"
                }
            }
            1
            {
                Write-Host "Enabling [~] Downloads folder in This PC (Shown)"
                $This.Output[0].Get()
                $This.Output[1].Get()
                $This.Output[2].Get()
                $This.Output[3].Set("String","Show")
                $This.Output[4].Set("String","{374DE290-123F-4565-9164-39C4925E467B}")
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[5].Get()
                    $This.Output[6].Get()
                    $This.Output[7].Get()
                    $This.Output[8].Set("String","Show")
                }
            }
            2
            {
                Write-Host "Setting [~] Downloads folder in This PC (Hidden)"
                $This.Output[3].Set("String","Hide")
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[8].Set("String","Hide")
                }
            }
            3
            {
                Write-Host "Setting [~] Documents folder in This PC (None)"
                $This.Output[0].Remove()
                $This.Output[1].Remove()
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[5].Remove()
                    $This.Output[6].Remove()
                }
            }
        }
    }
}
```

```
                                                                                      ________________________________________________/
\______________________________________________________________________/ Class [DownloadsIconInThisPC : ControlTemplate]
  Class [MusicIconInThisPC : ControlTemplate] /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

```
Class MusicIconInThisPC : ControlTemplate
{
    MusicIconInThisPC()
    {
        $This.Name        = "MusicIconInThisPC"
        $This.DisplayName = "Music [Explorer]"
        $This.Value       = 1
        $This.Description = "Toggles the Music icon in 'This PC'"
        $This.Options     = "Skip", "Show/Add*", "Hide", "Remove"

        ForEach ($X in 0,1)
        {
            ($This.Path($X,0,0),$Null),
            ($This.Path($X,0,1),$Null),
            ("$($This.Path($X,1,2))\PropertyBag",$Null),
            ("$($This.Path($X,1,2))\PropertyBag","ThisPCPolicy"),
            ("$($This.Path($X,1,2))\PropertyBag","BaseFolderID") | % {

                $This.Registry($_[0],$_[1])
            }
        }
    }
    [String] Path([UInt32]$Slot,[UInt32]$Base,[UInt32]$Guid)
    {
        Return "HKLM:","SOFTWARE",@($Null,"WOW6432Node")[$Slot],"Microsoft","Windows",
        "CurrentVersion","Explorer",@("MyComputer\NameSpace","FolderDescriptions")[$Base],
        $This.Guid($Guid) -join "\"
    }
    [String] Guid([UInt32]$Slot)
    {
        Return @("{1CF1260C-4DD0-4ebb-811F-33C572699FDE}",
                 "{3dfdf296-dbec-4fb4-81d1-6a3438bcf4de}",
                 "{a0c69a99-21c8-4671-8703-7934162fcf1d}")[$Slot]
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Music folder in This PC"
                }
            }
            1
            {
                Write-Host "Enabling [~] Music folder in This PC (Shown)"
                $This.Output[0].Get()
                $This.Output[1].Get()
                $This.Output[2].Get()
                $This.Output[3].Set("String","Show")
                $This.Output[4].Set("String","{4BD8D571-6D19-48D3-BE97-422220080E43}")
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[5].Get()
                    $This.Output[6].Get()
                    $This.Output[7].Get()
                    $This.Output[8].Set("String","Show")
                }
            }
            2
            {
                Write-Host "Setting [~] Music folder in This PC (Hidden)"
                $This.Output[3].Set("String","Hide")
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[8].Set("String","Hide")
                }
            }
            3
            {
```

```
                Write-Host "Setting [~] Music folder in This PC (None)"
                $This.Output[0].Remove()
                $This.Output[1].Remove()
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[5].Remove()
                    $This.Output[6].Remove()
                }
            }
        }
    }
}
```

Class [PicturesIconInThisPC : ControlTemplate]

Class [MusicIconInThisPC : ControlTemplate]

```
Class PicturesIconInThisPC : ControlTemplate
{
    PicturesIconInThisPC()
    {
        $This.Name        = "PicturesIconInThisPC"
        $This.DisplayName = "Pictures [Explorer]"
        $This.Value       = 1
        $This.Description = "Toggles the Pictures icon in 'This PC'"
        $This.Options     = "Skip", "Show/Add*", "Hide", "Remove"

        ForEach ($X in 0,1)
        {
            ($This.Path($X,0,0),$Null),
            ($This.Path($X,0,1),$Null),
            ("$($This.Path($X,1,2))\PropertyBag",$Null),
            ("$($This.Path($X,1,2))\PropertyBag","ThisPCPolicy"),
            ("$($This.Path($X,1,2))\PropertyBag","BaseFolderID") | % {

                $This.Registry($_[0],$_[1])
            }
        }
    }
    [String] Path([UInt32]$Slot,[UInt32]$Base,[UInt32]$Guid)
    {
        Return "HKLM:","SOFTWARE",@($Null,"WOW6432Node")[$Slot],"Microsoft","Windows",
        "CurrentVersion","Explorer",@("MyComputer\NameSpace","FolderDescriptions")[$Base],
        $This.Guid($Guid) -join "\"
    }
    [String] Guid([UInt32]$Slot)
    {
        Return @("{24ad3ad4-a569-4530-98e1-ab02f9417aa8}",
                 "{3ADD1653-EB32-4cb0-BBD7-DFA0ABB5ACCA}",
                 "{0ddd015d-b06c-45d5-8c4c-f59713854639}")[$Slot]
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Pictures folder in This PC"
                }
            }
            1
            {
                Write-Host "Enabling [~] Pictures folder in This PC (Shown)"
                $This.Output[0].Get()
                $This.Output[1].Get()
                $This.Output[2].Get()
                $This.Output[3].Set("String","Show")
                $This.Output[4].Set("String","{33E28130-4E1E-4676-835A-98395C3BC3BB}")
                If ([Environment]::Is64BitOperatingSystem)
```

```
                {
                    $This.Output[5].Get()
                    $This.Output[6].Get()
                    $This.Output[7].Get()
                    $This.Output[8].Set("String","Show")
                }
            }
            2
            {
                Write-Host "Setting [~] Pictures folder in This PC (Hidden)"
                $This.Output[3].Set("String","Hide")
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[8].Set("String","Hide")
                }
            }
            3
            {
                Write-Host "Setting [~] Pictures folder in This PC (None)"
                $This.Output[0].Remove()
                $This.Output[1].Remove()
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[5].Remove()
                    $This.Output[6].Remove()
                }
            }
        }
    }
}
```

_____/ _____/ Class [PicturesIconInThisPC : ControlTemplate]
Class [VideosIconInThisPC : ControlTemplate] /_____\
/_____

```
Class VideosIconInThisPC : ControlTemplate
{
    VideosIconInThisPC()
    {
        $This.Name        = "VideosIconInThisPC"
        $This.DisplayName = "Videos [Explorer]"
        $This.Value       = 1
        $This.Description = "Toggles the Videos icon in 'This PC'"
        $This.Options     = "Skip", "Show/Add*", "Hide", "Remove"

        ForEach ($X in 0,1)
        {
            ($This.Path($X,0,0),$Null),
            ($This.Path($X,0,1),$Null),
            ("$($This.Path($X,1,2))\PropertyBag",$Null),
            ("$($This.Path($X,1,2))\PropertyBag","ThisPCPolicy"),
            ("$($This.Path($X,1,2))\PropertyBag","BaseFolderID") | % {

                $This.Registry($_[0],$_[1])
            }
        }
    }
    [String] Path([UInt32]$Slot,[UInt32]$Base,[UInt32]$Guid)
    {
        Return "HKLM:","SOFTWARE",@($Null,"WOW6432Node")[$Slot],"Microsoft","Windows",
        "CurrentVersion","Explorer",@("MyComputer\NameSpace","FolderDescriptions")[$Base],
        $This.Guid($Guid) -join "\"
    }
    [String] Guid([UInt32]$Slot)
    {
        Return @("{A0953C92-50DC-43bf-BE83-3742FED03C9C}",
                 "{f86fa3ab-70d2-4fc7-9c99-fcbf05467f3a}",
                 "{35286a68-3c57-41a1-bbb1-0eae73d76c95}")[$Slot]
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
```

```
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Videos folder in This PC"
                }
            }
            1
            {
                Write-Host "Enabling [~] Videos folder in This PC (Shown)"
                $This.Output[0].Get()
                $This.Output[1].Get()
                $This.Output[2].Get()
                $This.Output[3].Set("String","Show")
                $This.Output[4].Set("String","{18989B1D-99B5-455B-841C-AB7C74E4DDFC}")
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[5].Get()
                    $This.Output[6].Get()
                    $This.Output[7].Get()
                    $This.Output[8].Set("String","Show")
                }
            }
            2
            {
                Write-Host "Setting [~] Videos folder in This PC (Hidden)"
                $This.Output[3].Set("String","Hide")
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[8].Set("String","Hide")
                }
            }
            3
            {
                Write-Host "Setting [~] Videos folder in This PC (None)"
                $This.Output[0].Remove()
                $This.Output[1].Remove()
                If ([Environment]::Is64BitOperatingSystem)
                {
                    $This.Output[5].Remove()
                    $This.Output[6].Remove()
                }
            }
        }
    }
}
```

_____/ Class [VideosIconInThisPC : ControlTemplate]
_____/ Class [ThreeDObjectsIconInThisPC : ControlTemplate] /_____\
/_____

```
Class ThreeDObjectsIconInThisPC : ControlTemplate
{
    ThreeDObjectsIconInThisPC()
    {
        $This.Name        = "ThreeDObjectsIconInThisPC"
        $This.DisplayName = "3D Objects [Explorer]"
        $This.Value       = 1
        $This.Description = "Toggles the 3D Objects icon in 'This PC'"
        $This.Options     = "Skip", "Show/Add*", "Hide", "Remove"

        ForEach ($X in 0,1)
        {
            ("$($This.Path($X))\$($This.Guid(0))",$Null),
            ("$($This.Path($X))\$($This.Guid(1))\PropertyBag",$Null),
            ("$($This.Path($X))\$($This.Guid(1))\PropertyBag","ThisPCPolicy") | % {

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

```
        }
    }
    [String] Path([UInt32]$Slot)
    {
        Return "HKLM:","SOFTWARE",@($Null,"WOW6432Node")[$Slot],"Microsoft","Windows","CurrentVersion",
        "Explorer","FolderDescriptions" -join "\"
    }
    [String] Guid([UInt32]$Slot)
    {
        Return @("{0DB7E03F-FC29-4DC6-9020-FF41B59E513A}",
                 "{31C0DD25-9439-4F12-BF41-7FF4EDA38722}")[$Slot]
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        If ($This.GetWinVersion() -ge 1709)
        {
            Switch ($Mode)
            {
                0
                {
                    If ($ShowSkipped)
                    {
                        Write-Host "Skipping [!] 3D Objects folder in This PC"
                    }
                }
                1
                {
                    Write-Host "Enabling [~] 3D Objects folder in This PC (Shown)"
                    $This.Output[0].Get()
                    $This.Output[1].Get()
                    $This.Output[2].Set("String","Show")
                    If ([Environment]::Is64BitOperatingSystem)
                    {
                        $This.Output[3].Get()
                        $This.Output[4].Get()
                        $This.Output[5].Set("String","Show")
                    }
                }
                2
                {
                    Write-Host "Setting [~] 3D Objects folder in This PC (Hidden)"
                    $This.Output[2].Set("String","Hide")
                    If ([Environment]::Is64BitOperatingSystem)
                    {
                        $This.Output[5].Set("String","Hide")
                    }
                }
                3
                {
                    Write-Host "Setting [~] 3D Objects folder in This PC (None)"
                    $This.Output[1].Remove()
                    If ([Environment]::Is64BitOperatingSystem)
                    {
                        $This.Output[5].Remove()
                    }
                }
            }
        }
    }
}
```

Enum [ThisPCIconType] Class [ThreeDObjectsIconInThisPC : ControlTemplate]

```
Enum ThisPCIconType
{
    DesktopIconInThisPC
    DocumentsIconInThisPC
    DownloadsIconInThisPC
    MusicIconInThisPC
```

```
    PicturesIconInThisPC
    VideosIconInThisPC
    ThreeDObjectsIconInThisPC
}
```

Enum [ThisPCIconType]

Class [ThisPCIconList]

```
Class ThisPCIconList
{
    [Object]  $Output
    ThisPCIconList()
    {
        $This.Output  = @( )
        ForEach ($Name in [System.Enum]::GetNames([ThisPCIconType]))
        {
            $Item = Switch ($Name)
            {
                DesktopIconInThisPC       { [DesktopIconInThisPC]::New()       }
                DocumentsIconInThisPC     { [DocumentsIconInThisPC]::New()     }
                DownloadsIconInThisPC     { [DownloadsIconInThisPC]::New()     }
                MusicIconInThisPC         { [MusicIconInThisPC]::New()         }
                PicturesIconInThisPC      { [PicturesIconInThisPC]::New()      }
                VideosIconInThisPC        { [VideosIconInThisPC]::New()        }
                ThreeDObjectsIconInThisPC { [ThreeDObjectsIconInThisPC]::New() }
            }
            $Item.Source  = "ThisPC"
            $This.Output += $Item
        }
    }
}
```

Class [ThisPCIconList]

Class [ThisPCOnDesktop : ControlTemplate]

```
Class ThisPCOnDesktop : ControlTemplate
{
    ThisPCOnDesktop()
    {
        $This.Name        = "ThisPCOnDesktop"
        $This.DisplayName = "This PC [Desktop]"
        $This.Value       = 2
        $This.Description = "Toggles the 'This PC' icon on the desktop"
        $This.Options     = "Skip", "Show", "Hide*"

        ForEach ($Item in "ClassicStartMenu","NewStartPanel")
        {
            $This.Registry("$($This.RegPath())\$Item",'{20D04FE0-3AEA-1069-A2D8-08002B30309D}')
        }
    }
    [String] RegPath()
    {
        Return "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\HideDesktopIcons"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] This PC Icon on desktop"
                }
            }
            1
            {
```

```
                Write-Host "Setting [~] This PC Icon on desktop (Shown)"
                $This.Output[0].Set(0)
                $This.Output[0].Set(0)
            }
            2
            {
                Write-Host "Setting [~] This PC Icon on desktop (Hidden)"
                $This.Output[0].Set(1)
                $This.Output[0].Set(1)
            }
        }
    }
}
```

Class [NetworkOnDesktop : ControlTemplate]

Class [ThisPCOnDesktop : ControlTemplate]

```
Class NetworkOnDesktop : ControlTemplate
{
    NetworkOnDesktop()
    {
        $This.Name        = "NetworkOnDesktop"
        $This.DisplayName = "Network [Desktop]"
        $This.Value       = 2
        $This.Description = "Toggles the 'Network' icon on the desktop"
        $This.Options     = "Skip", "Show", "Hide*"

        ForEach ($Item in "ClassicStartMenu","NewStartPanel")
        {
            $This.Registry("$($This.RegPath())\$Item",'{F02C1A0D-BE21-4350-88B0-7367FC96EF3C}')
        }
    }
    [String] RegPath()
    {
        Return "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\HideDesktopIcons"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Network Icon on desktop"
                }
            }
            1
            {
                Write-Host "Setting [~] Network Icon on desktop (Shown)"
                $This.Output[0].Set(0)
                $This.Output[0].Set(0)
            }
            2
            {
                Write-Host "Setting [~] Network Icon on desktop (Hidden)"
                $This.Output[0].Set(1)
                $This.Output[0].Set(1)
            }
        }
    }
}
```

Class [NetworkOnDesktop : ControlTemplate]

Class [RecycleBinOnDesktop : ControlTemplate]

```
Class RecycleBinOnDesktop : ControlTemplate
```

```
{
    RecycleBinOnDesktop()
    {
        $This.Name        = "RecycleBinOnDesktop"
        $This.DisplayName = "Recycle Bin [Desktop]"
        $This.Value       = 2
        $This.Description = "Toggles the 'Recycle Bin' icon on the desktop"
        $This.Options     = "Skip", "Show", "Hide*"

        ForEach ($Item in "ClassicStartMenu","NewStartPanel")
        {
            $This.Registry("$($This.RegPath())\$Item",'{645FF040-5081-101B-9F08-00AA002F954E}')
        }
    }
    [String] RegPath()
    {
        Return "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\HideDesktopIcons"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Recycle Bin Icon on desktop"
                }
            }
            1
            {
                Write-Host "Setting [~] Recycle Bin Icon on desktop (Shown)"
                $This.Output[0].Set(0)
                $This.Output[0].Set(0)
            }
            2
            {
                Write-Host "Setting [~] Recycle Bin Icon on desktop (Hidden)"
                $This.Output[0].Set(1)
                $This.Output[0].Set(1)
            }
        }
    }
}
```

```
                                                                          ______________________________________________/
\________________________________________________________________________/ Class [RecycleBinOnDesktop : ControlTemplate]
  Class [UsersFileOnDesktop : ControlTemplate] /------------------------------------------------------------------------\
/----------------------------------------------
```

```
Class UsersFileOnDesktop : ControlTemplate
{
    UsersFileOnDesktop()
    {
        $This.Name        = "UsersFileOnDesktop"
        $This.DisplayName = "My Documents [Desktop]"
        $This.Value       = 2
        $This.Description = "Toggles the 'Users File' icon on the desktop"
        $This.Options     = "Skip", "Show", "Hide*"

        ForEach ($Item in "ClassicStartMenu","NewStartPanel")
        {
            $This.Registry("$($This.RegPath())\$Item",'{59031a47-3f72-44a7-89c5-5595fe6b30ee}')
        }
    }
    [String] RegPath()
    {
        Return "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\HideDesktopIcons"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
```

```
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Users file Icon on desktop"
                }
            }
            1
            {
                Write-Host "Setting [~] Users file Icon on desktop (Shown)"
                $This.Output[0].Set(0)
                $This.Output[0].Set(0)
            }
            2
            {
                Write-Host "Setting [~] Users file Icon on desktop (Hidden)"
                $This.Output[0].Set(1)
                $This.Output[0].Set(1)
            }
        }
    }
}
```

_____/
_____/ Class [UsersFileOnDesktop : ControlTemplate]
Class [ControlPanelOnDesktop : ControlTemplate] /_____\
/_____

```
Class ControlPanelOnDesktop : ControlTemplate
{
    ControlPanelOnDesktop()
    {
        $This.Name        = "ControlPanelOnDesktop"
        $This.DisplayName = "Control Panel [Desktop]"
        $This.Value       = 2
        $This.Description = "Toggles the 'Control Panel' icon on the desktop"
        $This.Options     = "Skip", "Show", "Hide*"

        ForEach ($Item in "ClassicStartMenu","NewStartPanel")
        {
            $This.Registry("$($This.RegPath())\$Item",'{5399E694-6CE5-4D6C-8FCE-1D8870FDCBA0}')
        }
    }
    [String] RegPath()
    {
        Return "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\HideDesktopIcons"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Control Panel Icon on desktop"
                }
            }
            1
            {
                Write-Host "Setting [~] Control Panel Icon on desktop (Shown)"
                $This.Output[0].Set(0)
                $This.Output[0].Set(0)
            }
            2
            {
                Write-Host "Setting [~] Control Panel Icon on desktop (Hidden)"
                $This.Output[0].Set(1)
                $This.Output[0].Set(1)
            }
        }
```

```
    }
}
```

_____/
_____/ Class [ControlPanelOnDesktop : ControlTemplate]
Enum [DesktopIconType] /_____\
/_____

```
Enum DesktopIconType
{
    ThisPCOnDesktop
    NetworkOnDesktop
    RecycleBinOnDesktop
    UsersFileOnDesktop
    ControlPanelOnDesktop
}
```

_____/
_____/ Enum [DesktopIconType]
Class [DesktopIconList] /_____\
/_____

```
Class DesktopIconList
{
    [Object]  $Output
    DesktopIconList()
    {
        $This.Output = @( )
        ForEach ($Name in [System.Enum]::GetNames([DesktopIconType]))
        {
            $Item = Switch ($Name)
            {
                ThisPCOnDesktop       { [ThisPCOnDesktop]::New()   }
                NetworkOnDesktop      { [NetworkOnDesktop]::New()      }
                RecycleBinOnDesktop   { [RecycleBinOnDesktop]::New()   }
                UsersFileOnDesktop    { [UsersFileOnDesktop]::New()    }
                ControlPanelOnDesktop { [ControlPanelOnDesktop]::New() }
            }
            $Item.Source  = "Desktop"
            $This.Output += $Item
        }
    }
}
```

_____/
_____/ Class [DesktopIconList]
Class [LockScreen : ControlTemplate] /_____\
/_____

```
Class LockScreen : ControlTemplate
{
    LockScreen()
    {
        $This.Name        = "LockScreen"
        $This.DisplayName = "Lock Screen"
        $This.Value       = 1
        $This.Description = "Toggles the lock screen"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry('HKLM:\SOFTWARE\Policies\Microsoft\Windows\Personalization','NoLockScreen')
    }
    [String] Argument()
    {
        $Item = "HKLM","SOFTWARE","Microsoft","Windows","CurrentVersion","Authentication",
                "LogonUI","SessionData" -join "\"
        Return "add $Item /t REG_DWORD /v AllowLockScreen /d 0 /f"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
```

```
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Lock Screen"
                }
            }
            1
            {
                Write-Host "Enabling [~] Lock Screen"
                If ($This.GetWinVersion() -ge 1607)
                {
                    Unregister-ScheduledTask -TaskName "Disable LockScreen" -Confirm:$False -Verbose
                }
                Else
                {
                    $This.Output[0].Remove()
                }
            }
            2
            {
                Write-Host "Disabling [~] Lock Screen"
                If ($This.GetWinVersion() -ge 1607)
                {
                    $Service             = New-Object -ComObject Schedule.Service
                    $Service.Connect()
                    $Task                = $Service.NewTask(0)
                    $Task.Settings.DisallowStartIfOnBatteries = $False
                    $Trigger             = $Task.Triggers.Create(9)
                    $Trigger             = $Task.Triggers.Create(11)
                    $Trigger.StateChange = 8
                    $Action              = $Task.Actions.Create(0)
                    $Action.Path         = 'Reg.exe'
                    $Action.Arguments    = $This.Argument()
                    $Service.GetFolder('\').RegisterTaskDefinition('Disable LockScreen',$Task,6,
                                                                   'NT AUTHORITY\SYSTEM',$Null,4)
                }
                Else
                {
                    $This.Output[0].Set(1)
                }
            }
        }
    }
}
```

Class [LockScreen : ControlTemplate]

Class [LockScreenPassword : ControlTemplate]

```
Class LockScreenPassword : ControlTemplate
{
    LockScreenPassword()
    {
        $This.Name        = "LockScreenPassword"
        $This.DisplayName = "Lock Screen Password"
        $This.Value       = 1
        $This.Description = "Toggles the lock screen password"
        $This.Options     = "Skip", "Enable*", "Disable"

        ("HKLM:\Software\Policies\Microsoft\Windows\Control Panel\Desktop",
         "ScreenSaverIsSecure"),
        ("HKCU:\Software\Policies\Microsoft\Windows\Control Panel\Desktop",
         "ScreenSaverIsSecure") | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
```

```
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Lock Screen Password"
                }
            }
            1
            {
                Write-Host "Enabling [~] Lock Screen Password"
                $This.Output[0].Set(1)
                $This.Output[1].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Lock Screen Password"
                $This.Output[0].Set(0)
                $This.Output[1].Set(0)
            }
        }
    }
}
```

Class [LockScreenPassword : ControlTemplate]

Class [PowerMenuLockScreen : ControlTemplate]

```
Class PowerMenuLockScreen : ControlTemplate
{
    PowerMenuLockScreen()
    {
        $This.Name        = "PowerMenuLockScreen"
        $This.DisplayName = "Power Menu Lock Screen"
        $This.Value       = 1
        $This.Description = "Toggles the power menu on the lock screen"
        $This.Options     = "Skip", "Show*", "Hide"

        $This.Registry('HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System','shutdownwithoutlogon')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Power Menu on Lock Screen"
                }
            }
            1
            {
                Write-Host "Enabling [~] Power Menu on Lock Screen"
                $This.Output[0].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Power Menu on Lock Screen"
                $This.Output[0].Set(0)
            }
        }
    }
}
```

Class [PowerMenuLockScreen : ControlTemplate]

Class [CameraOnLockScreen : ControlTemplate]

```
Class CameraOnLockScreen : ControlTemplate
{
    CameraOnLockScreen()
    {
        $This.Name        = "CameraOnLockScreen"
        $This.DisplayName = "Camera On Lock Screen"
        $This.Value       = 1
        $This.Description = "Toggles the camera on the lock screen"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry('HKLM:\SOFTWARE\Policies\Microsoft\Windows\Personalization','NoLockScreenCamera')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Camera at Lockscreen"
                }
            }
            1
            {
                Write-Host "Enabling [~] Camera at Lockscreen"
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Camera at Lockscreen"
                $This.Output[0].Set(1)
            }
        }
    }
}
```

```
Enum LockScreenType
{
    LockScreen
    LockScreenPassword
    PowerMenuLockScreen
    CameraOnLockScreen
}
```

```
Class LockScreenList
{
    [Object]  $Output
    LockScreenList()
    {
        $This.Output = @( )
        ForEach ($Name in [System.Enum]::GetNames([LockScreenType]))
        {
            $Item = Switch ($Name)
            {
                LockScreen          { [LockScreen]::New()          }
                LockScreenPassword  { [LockScreenPassword]::New()  }
                PowerMenuLockScreen { [PowerMenuLockScreen]::New() }
                CameraOnLockScreen  { [CameraOnLockScreen]::New()  }
            }
            $Item.Source  = "LockScreen"
```

```
            $This.Output += $Item
        }
    }
}
```

Class [LockScreenList]

Class [ScreenSaver : ControlTemplate]

```
Class ScreenSaver : ControlTemplate
{
    ScreenSaver()
    {
        $This.Name        = "ScreenSaver"
        $This.DisplayName = "Screen Saver"
        $This.Value       = 1
        $This.Description = "Toggles the screen saver"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry("HKCU:\Control Panel\Desktop","ScreenSaveActive")
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Screensaver"
                }
            }
            1
            {
                Write-Host "Enabling [~] Screensaver"
                $This.Output[0].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Screensaver"
                $This.Output[0].Set(0)
            }
        }
    }
}
```

Class [ScreenSaver : ControlTemplate]

Class [AccountProtectionWarn : ControlTemplate]

```
Class AccountProtectionWarn : ControlTemplate
{
    AccountProtectionWarn()
    {
        $This.Name        = "AccountProtectionWarn"
        $This.DisplayName = "Account Protection Warning"
        $This.Value       = 1
        $This.Description = "Toggles system security account protection warning"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry('HKCU:\SOFTWARE\Microsoft\Windows Security Health\State',
                       'AccountProtection_MicrosoftAccount_Disconnected')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        If ($This.GetWinVersion() -ge 1803)
        {
            Switch ($Mode)
            {
```

```
                0
                {
                    If ($ShowSkipped)
                    {
                        Write-Host "Skipping [!] Account Protection Warning"
                    }
                }
                1
                {
                    Write-Host "Enabling [~] Account Protection Warning"
                    $This.Output[0].Remove()
                }
                2
                {
                    Write-Host "Disabling [~] Account Protection Warning"
                    $This.Output[0].Set(1)
                }
            }
        }
    }
}
```

Class [ActionCenter : ControlTemplate] Class [AccountProtectionWarn : ControlTemplate]

```
Class ActionCenter : ControlTemplate
{
    ActionCenter()
    {
        $This.Name        = "ActionCenter"
        $This.DisplayName = "Action Center"
        $This.Value       = 1
        $This.Description = "Toggles system action center"
        $This.Options     = "Skip", "Enable*", "Disable"

        ('HKCU:\SOFTWARE\Policies\Microsoft\Windows\Explorer',
         'DisableNotificationCenter'),
        ('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\PushNotifications',
         'ToastEnabled') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Action Center"
                }
            }
            1
            {
                Write-Host "Enabling [~] Action Center"
                $This.Output[0].Remove()
                $This.Output[1].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Action Center"
                $This.Output[0].Set(1)
                $This.Output[1].Set(0)
            }
        }
    }
}
```

Class [StickyKeyPrompt : ControlTemplate]

```
Class StickyKeyPrompt : ControlTemplate
{
    StickyKeyPrompt()
    {
        $This.Name        = "StickyKeyPrompt"
        $This.DisplayName = "Sticky Key Prompt"
        $This.Value       = 1
        $This.Description = "Toggles the sticky keys prompt/dialog"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry('HKCU:\Control Panel\Accessibility\StickyKeys','Flags')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Sticky Key Prompt"
                }
            }
            1
            {
                Write-Host "Enabling [~] Sticky Key Prompt"
                $This.Output[0].Set("String",510)
            }
            2
            {
                Write-Host "Disabling [~] Sticky Key Prompt"
                $This.Output[0].Set("String",506)
            }
        }
    }
}
```

Class [NumbLockOnStart : ControlTemplate]

```
Class NumbLockOnStart : ControlTemplate
{
    NumbLockOnStart()
    {
        $This.Name        = "NumbLockOnStart"
        $This.DisplayName = "Number lock on start"
        $This.Value       = 2
        $This.Description = "Toggles whether the number lock key is engaged upon start"
        $This.Options     = "Skip", "Enable", "Disable*"

        $This.Registry('HKU:\.DEFAULT\Control Panel\Keyboard','InitialKeyboardIndicators')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Num Lock on startup"
                }
            }
            1
            {
```

```
                Write-Host "Enabling [~] Num Lock on startup"
                $This.Output[0].Set(2147483650)
            }
            2
            {
                Write-Host "Disabling [~] Num Lock on startup"
                $This.Output[0].Set(2147483648)
            }
        }
    }
}
```

Class [NumbLockOnStart : ControlTemplate]

Class [F8BootMenu : ControlTemplate]

```
Class F8BootMenu : ControlTemplate
{
    F8BootMenu()
    {
        $This.Name        = "F8BootMenu"
        $This.DisplayName = "F8 Boot Menu"
        $This.Value       = 2
        $This.Description = "Toggles whether the F8 boot menu can be access upon boot"
        $This.Options     = "Skip", "Enable", "Disable*"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] F8 Boot menu options"
                }
            }
            1
            {
                Write-Host "Enabling [~] F8 Boot menu options"
                bcdedit /set `{current`} bootmenupolicy Legacy
            }
            2
            {
                Write-Host "Disabling [~] F8 Boot menu options"
                bcdedit /set `{current`} bootmenupolicy Standard
            }
        }
    }
}
```

Class [F8BootMenu : ControlTemplate]

Class [RemoteUACAcctToken : ControlTemplate]

```
Class RemoteUACAcctToken : ControlTemplate
{
    RemoteUACAcctToken()
    {
        $This.Name        = "RemoteUACAcctToken"
        $This.DisplayName = "Remote UAC Account Token"
        $This.Value       = 2
        $This.Description = "Toggles the local account token filter policy to mitigate remote connections"
        $This.Options     = "Skip", "Enable", "Disable*"

        $This.Registry('HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System',
                       'LocalAccountTokenFilterPolicy')
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
```

```
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Remote UAC Local Account Token Filter"
                }
            }
            1
            {
                Write-Host "Enabling [~] Remote UAC Local Account Token Filter"
                $This.Output[0].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] Remote UAC Local Account Token Filter"
                $This.Output[0].Remove()
            }
        }
    }
}
```

_____/
_____/ Class [RemoteUACAcctToken : ControlTemplate]
Class [HibernatePower : ControlTemplate] /_____\
/_____

```
Class HibernatePower : ControlTemplate
{
    HibernatePower()
    {
        $This.Name        = "HibernatePower"
        $This.DisplayName = "Hibernate Power"
        $This.Value       = 0
        $This.Description = "Toggles the hibernation power option"
        $This.Options     = "Skip", "Enable", "Disable"

        ('HKLM:\SYSTEM\CurrentControlSet\Control\Power',
         'HibernateEnabled'),
        ('HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\FlyoutMenuSettings',
         'ShowHibernateOption') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Hibernate Option"
                }
            }
            1
            {
                Write-Host "Enabling [~] Hibernate Option"
                $This.Output[0].Set(1)
                $This.Output[1].Set(1)
                powercfg /HIBERNATE ON
            }
            2
            {
                Write-Host "Disabling [~] Hibernate Option"
                $This.Output[0].Set(0)
                $This.Output[1].Set(0)
                powercfg /HIBERNATE OFF
            }
```

```
        }
    }
}
```

Class [HibernatePower : ControlTemplate]

Class [SleepPower : ControlTemplate]

```
Class SleepPower : ControlTemplate
{
    SleepPower()
    {
        $This.Name        = "SleepPower"
        $This.DisplayName = "Sleep Power"
        $This.Value       = 1
        $This.Description = "Toggles the sleep power option"
        $This.Options     = "Skip", "Enable*", "Disable"

        $This.Registry('HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\FlyoutMenuSettings',
                       "ShowSleepOption")
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Sleep Option"
                }
            }
            1
            {
                Write-Host "Enabling [~] Sleep Option"
                $This.Output[0].Set(1)
                powercfg /SETACVALUEINDEX SCHEME_CURRENT SUB_BUTTONS SBUTTONACTION 1
                powercfg /SETDCVALUEINDEX SCHEME_CURRENT SUB_BUTTONS SBUTTONACTION 1
            }
            2
            {
                Write-Host "Disabling [~] Sleep Option"
                $This.Output[0].Set(0)
                powercfg /SETACVALUEINDEX SCHEME_CURRENT SUB_BUTTONS SBUTTONACTION 0
                powercfg /SETDCVALUEINDEX SCHEME_CURRENT SUB_BUTTONS SBUTTONACTION 0
            }
        }
    }
}
```

Class [SleepPower : ControlTemplate]

Enum [MiscellaneousType]

```
Enum MiscellaneousType
{
    ScreenSaver
    AccountProtectionWarn
    ActionCenter
    StickyKeyPrompt
    NumblockOnStart
    F8BootMenu
    RemoteUACAcctToken
    HibernatePower
    SleepPower
}
```

Enum [MiscellaneousType]

Class [MiscellaneousList]

```
Class MiscellaneousList
{
    [Object]  $Output
    MiscellaneousList()
    {
        $This.Output = @( )
        ForEach ($Name in [System.Enum]::GetNames([MiscellaneousType]))
        {
            $Item = Switch ($Name)
            {
                ScreenSaver           { [ScreenSaver]::New()           }
                AccountProtectionWarn { [AccountProtectionWarn]::New() }
                ActionCenter          { [ActionCenter]::New()          }
                StickyKeyPrompt       { [StickyKeyPrompt]::New()       }
                NumblockOnStart       { [NumblockOnStart]::New()       }
                F8BootMenu            { [F8BootMenu]::New()            }
                RemoteUACAcctToken    { [RemoteUACAcctToken]::New()    }
                HibernatePower        { [HibernatePower]::New()        }
                SleepPower            { [SleepPower]::New()            }
            }
            $Item.Source  = "Miscellaneous"
            $This.Output += $Item
        }
    }
}
```

Class [MiscellaneousList]

Class [PVFileAssociation : ControlTemplate]

```
Class PVFileAssociation : ControlTemplate
{
    PVFileAssociation()
    {
        $This.Name        = "PVFileAssociation"
        $This.DisplayName = "Photo Viewer File Association"
        $This.Value       = 2
        $This.Description = "Associates common image types with Photo Viewer"
        $This.Options     = "Skip", "Enable", "Disable*"

        ("HKCR:\Paint.Picture\shell\open","MUIVerb"),
        ("HKCR:\giffile\shell\open","MUIVerb"),
        ("HKCR:\jpegfile\shell\open","MUIVerb"),
        ("HKCR:\pngfile\shell\open","MUIVerb"),
        ("HKCR:\Paint.Picture\shell\open\command","(Default)"),
        ("HKCR:\giffile\shell\open\command","(Default)"),
        ("HKCR:\jpegfile\shell\open\command","(Default)"),
        ("HKCR:\pngfile\shell\open\command","(Default)"),
        ("HKCR:\giffile\shell\open","CommandId"),
        ("HKCR:\giffile\shell\open\command","DelegateExecute") | % {

            $This.Registry($_[0],$_[1])
        }
    }
    [String] RunDll32()
    {
        Return "{0} `"{1}`", {2}" -f "%SystemRoot%\System32\rundll32.exe",
                                     "%ProgramFiles%\Windows Photo Viewer\PhotoViewer.dll",
                                     "ImageView_Fullscreen %1"
    }
    [String] IExplore()
    {
        $Item = "$Env:SystemDrive\Program Files\Internet Explorer\iexplore.exe"
        Return "`"$Item`" %1"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
```

```
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Photo Viewer File Association"
                }
            }
            1
            {
                Write-Host "Enabling [~] Photo Viewer File Association"
                0..3 | % {

                    $This.Output[$_  ].Set("ExpandString",
                                           "@%ProgramFiles%\Windows Photo Viewer\photoviewer.dll,-3043")
                    $This.Output[$_+4].Set("ExpandString",
                                           $This.RunDll32())
                }
            }
            2
            {
                Write-Host "Disabling [~] Photo Viewer File Association"
                $This.Output[0] | % { $_.Clear(); $_.Remove() }
                $This.Output[1].Remove()
                $This.Output[2] | % { $_.Clear(); $_.Remove() }
                $This.Output[3] | % { $_.Clear(); $_.Remove() }
                $This.Output[5].Set("String",
                                    $This.IExplore())
                $This.Output[8].Set("String",
                                    "IE.File")
                $This.Output[9].Set("String",
                                    "{17FE9752-0B5A-4665-84CD-569794602F5C}")
            }
        }
    }
}
```

_____/ Class [PVFileAssociation : ControlTemplate] /_____\
Class [PVOpenWithMenu : ControlTemplate] /_____/

```
Class PVOpenWithMenu : ControlTemplate
{
    PVOpenWithMenu()
    {
        $This.Name        = "PVOpenWithMenu"
        $This.DisplayName = "Photo Viewer 'Open with' Menu"
        $This.Value       = 2
        $This.Description = "Allows image files to be opened with Photo Viewer"
        $This.Options     = "Skip", "Enable", "Disable*"

        ('HKCR:\Applications\photoviewer.dll\shell\open',$Null),
        ('HKCR:\Applications\photoviewer.dll\shell\open\command',$Null),
        ('HKCR:\Applications\photoviewer.dll\shell\open\DropTarget',$Null),
        ('HKCR:\Applications\photoviewer.dll\shell\open','MuiVerb'),
        ('HKCR:\Applications\photoviewer.dll\shell\open\command','(Default)'),
        ('HKCR:\Applications\photoviewer.dll\shell\open\DropTarget','Clsid') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    [String] RunDll32()
    {
        Return "{0} `"{1}`", {2}" -f "%SystemRoot%\System32\rundll32.exe",
                                     "%ProgramFiles%\Windows Photo Viewer\PhotoViewer.dll",
                                     "ImageView_Fullscreen %1"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
```

```
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] 'Open with Photo Viewer' context menu item"
                }
            }
            1
            {
                Write-Host "Enabling [~] 'Open with Photo Viewer' context menu item"
                $This.Output[1].Get()
                $This.Output[2].Get()
                $This.Output[3].Set("String",
                                    "@photoviewer.dll,-3043")
                $This.Output[4].Set("ExpandString",
                                    $This.RunDll32())
                $This.Output[5].Set("String",
                                    "{FFE2A43C-56B9-4bf5-9A79-CC6D4285608A}")
            }
            2
            {
                Write-Host "Disabling [~] 'Open with Photo Viewer' context menu item"
                $This.Output[0].Remove()
            }
        }
    }
}
```

_____/
_____/ Class [PVOpenWithMenu : ControlTemplate]
Enum [PhotoViewerType] /_____\
/_____

```
Enum PhotoViewerType
{
    PVFileAssociation
    PVOpenWithMenu
}
```

_____/
_____/ Enum [PhotoViewerType]
Class [PhotoViewerList] /_____\
/_____

```
Class PhotoViewerList
{
    [Object]  $Output
    PhotoViewerList()
    {
        $This.Output = @( )
        ForEach ($Name in [System.Enum]::GetNames([PhotoViewerType]))
        {
            $Item = Switch ($Name)
            {
                PVFileAssociation { [PVFileAssociation]::New() }
                PVOpenWithMenu    { [PVOpenWithMenu]::New()    }
            }
            $Item.Source  = "PhotoViewer"
            $This.Output += $Item
        }
    }
}
```

_____/
_____/ Class [PhotoViewerList]
Class [OneDrive : ControlTemplate] /_____\
/_____

```
Class OneDrive : ControlTemplate
{
```

```
    OneDrive()
    {
        $This.Name        = "OneDrive"
        $This.DisplayName = "OneDrive"
        $This.Value       = 1
        $This.Description = "Toggles Microsoft OneDrive, which comes with the operating system"
        $This.Options     = "Skip", "Enable*", "Disable"

        ('HKLM:\SOFTWARE\Policies\Microsoft\Windows\OneDrive',
         'DisableFileSyncNGSC'),
        ('HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced',
         'ShowSyncProviderNotifications') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] OneDrive"
                }
            }
            1
            {
                Write-Host "Enabling [~] OneDrive"
                $This.Output[0].Remove()
                $This.Output[1].Set(1)
            }
            2
            {
                Write-Host "Disabling [~] OneDrive"
                $This.Output[0].Set(1)
                $This.Output[1].Set(0)
            }
        }
    }
}
```

Class [OneDrive : ControlTemplate]

Class [OneDriveInstall : ControlTemplate]

```
Class OneDriveInstall : ControlTemplate
{
    OneDriveInstall()
    {
        $This.Name        = "OneDriveInstall"
        $This.DisplayName = "OneDriveInstall"
        $This.Value       = 1
        $This.Description = "Installs/Uninstalls Microsoft OneDrive, which comes with the operating system"
        $This.Options     = "Skip", "Installed*", "Uninstall"

        ("HKCR:\CLSID\{018D5C66-4533-4307-9B53-224DE2ED1FE6}",$Null),
        ("HKCR:\Wow6432Node\CLSID\{018D5C66-4533-4307-9B53-224DE2ED1FE6}",$Null) | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
```

```
                    Write-Host "Skipping [!] OneDrive Install"
                }
            }
            1
            {
                Write-Host "Enabling [~] OneDrive Install"
                If ($This.TestPath())
                {
                    Start-Process $This.GetOneDrivePath() -NoNewWindow
                }
            }
            2
            {
                Write-Host "Disabling [~] OneDrive Install"
                If ($This.TestPath())
                {
                    Stop-Process -Name OneDrive -Force
                    Start-Sleep -Seconds 3
                    Start-Process $This.GetOneDrivePath() "/uninstall" -NoNewWindow -Wait
                    Start-Sleep -Seconds 3

                    ForEach ($Path in "$Env:USERPROFILE\OneDrive",
                                      "$Env:LOCALAPPDATA\Microsoft\OneDrive",
                                      "$Env:PROGRAMDATA\Microsoft OneDrive",
                                      "$Env:WINDIR\OneDriveTemp",
                                      "$Env:SYSTEMDRIVE\OneDriveTemp")
                    {
                        Remove-Item $Path -Force -Recurse
                    }

                    $This.Output[0].Remove()
                    $This.Output[1].Remove()
                }
            }
        }
    }
    [String] GetOneDrivePath()
    {
        $Item = @("System32","SysWOW64")[[Environment]::Is64BitOperatingSystem]
        Return "$Env:Windir\$Item\OneDriveSetup.exe"
    }
    [Bool] TestPath()
    {
        Return Test-Path $This.GetOneDrivePath() -PathType Leaf
    }
}
```

Class [XboxDVR : ControlTemplate] Class [OneDriveInstall : ControlTemplate]

```
Class XboxDVR : ControlTemplate
{
    XboxDVR()
    {
        $This.Name        = "XboxDVR"
        $This.DisplayName = "Xbox DVR"
        $This.Value       = 1
        $This.Description = "Toggles Microsoft Xbox DVR"
        $This.Options     = "Skip", "Enable*", "Disable"

        ('HKCU:\System\GameConfigStore','GameDVR_Enabled'),
        ('HKLM:\SOFTWARE\Policies\Microsoft\Windows\GameDVR','AllowGameDVR') | % {

            $This.Registry($_[0],$_[1])
        }
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
```

```
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Xbox DVR"
                }
            }
            1
            {
                Write-Host "Enabling [~] Xbox DVR"
                $This.Output[0].Set(1)
                $This.Output[0].Remove()
            }
            2
            {
                Write-Host "Disabling [~] Xbox DVR"
                $This.Output[0].Set(0)
                $This.Output[1].Set(0)
            }
        }
    }
}
```

Class [MediaPlayer : ControlTemplate]

Class [XboxDVR : ControlTemplate]

```
Class MediaPlayer : ControlTemplate
{
    MediaPlayer([Object]$Features)
    {
        $This.Name        = "MediaPlayer"
        $This.DisplayName = "Windows Media Player"
        $This.Value       = 1
        $This.Description = "Toggles Microsoft Windows Media Player, which comes with the operating system"
        $This.Options     = "Skip", "Installed*", "Uninstall"

        $This.Output      = @($Features | ? FeatureName -match MediaPlayback)
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Windows Media Player"
                }
            }
            1
            {
                Write-Host "Enabling [~] Windows Media Player"
                $This.Output[0] | ? State -ne Enabled | % {

                    Enable-WindowsOptionalFeature -Online -FeatureName WindowsMediaPlayer -NoRestart
                }

                If (!!$?)
                {
                    $This.Output[0].State = "Enabled"
                }
            }
            2
            {
                Write-Host "Disabling [~] Windows Media Player"
                $This.Output[0] | ? State -eq Enabled | % {

                    Disable-WindowsOptionalFeature -Online -FeatureName WindowsMediaPlayer -NoRestart
                }
```

```
                If (!!$?)
                {
                    $This.Output[0].State = "Disabled"
                }
            }
        }
    }
}
```

Class [MediaPlayer : ControlTemplate]

Class [WorkFolders : ControlTemplate]

```
Class WorkFolders : ControlTemplate
{
    WorkFolders([Object]$Features)
    {
        $This.Name        = "WorkFolders"
        $This.DisplayName = "Work Folders"
        $This.Value       = 1
        $This.Description = "Toggles the WorkFolders-Client, which comes with the operating system"
        $This.Options     = "Skip", "Installed*", "Uninstall"

        $This.Output      = @($Features | ? FeatureName -match WorkFolders-Client)
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Work Folders Client"
                }
            }
            1
            {
                Write-Host "Enabling [~] Work Folders Client"
                $This.Output[0] | ? State -ne Enabled | % {

                    Enable-WindowsOptionalFeature -Online -FeatureName WorkFolders-Client -NoRestart
                }

                If (!!$?)
                {
                    $This.Output[0].State = "Enabled"
                }
            }
            2
            {
                Write-Host "Disabling [~] Work Folders Client"
                $This.Output[0] | ? State -eq Enabled | % {

                    Disable-WindowsOptionalFeature -Online -FeatureName WorkFolders-Client -NoRestart
                }

                If (!!$?)
                {
                    $This.Output[0].State = "Disabled"
                }
            }
        }
    }
}
```

Class [WorkFolders : ControlTemplate]

Class [FaxAndScan : ControlTemplate]

```
Class FaxAndScan : ControlTemplate
{
    FaxAndScan([Object]$Features)
    {
        $This.Name        = "FaxAndScan"
        $This.DisplayName = "Fax and Scan"
        $This.Value       = 1
        $This.Description = "Toggles the FaxServicesClientPackage, which comes with the operating system"
        $This.Options     = "Skip", "Installed*", "Uninstall"

        $This.Output      = @($Features | ? FeatureName -match FaxServicesClientPackage)
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Fax And Scan"
                }
            }
            1
            {
                Write-Host "Enabling [~] Fax And Scan"
                $This.Output[0] | ? State -ne Enabled | % {

                    Enable-WindowsOptionalFeature -Online -FeatureName FaxServicesClientPackage -NoRestart
                }

                If (!!$?)
                {
                    $This.Output[0].State = "Enabled"
                }
            }
            2
            {
                Write-Host "Disabling [~] Fax And Scan"
                $This.Output[0] | ? State -eq Enabled | % {

                    Disable-WindowsOptionalFeature -Online -FeatureName FaxServicesClientPackage -NoRestart
                }

                If (!!$?)
                {
                    $This.Output[0].State = "Disabled"
                }
            }
        }
    }
}
```

_____/
_____/ Class [FaxAndScan : ControlTemplate]
Class [LinuxSubsystem : ControlTemplate] /_____\
/_____

```
Class LinuxSubsystem : ControlTemplate
{
    LinuxSubsystem([Object]$Features)
    {
        $This.Name        = "LinuxSubsystem"
        $This.DisplayName = "Linux Subsystem (WSL)"
        $This.Value       = 2
        $This.Description = "For Windows 1607+, this toggles the $($This.Feature())"
        $This.Options     = "Skip", "Installed", "Uninstall*"

        $This.Output      = @($Features | ? FeatureName -match $This.Feature())

        'AllowDevelopmentWithoutDevLicense','AllowAllTrustedApps' | % {
```

```
            $This.Registry('HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\AppModelUnlock',$_)
        }
    }
    [String] Feature()
    {
        Return "Microsoft-Windows-Subsystem-Linux"
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        If ($This.GetWinVersion() -gt 1607)
        {
            Switch ($Mode)
            {
                0
                {
                    If ($ShowSkipped)
                    {
                        Write-Host "Skipping [!] Linux Subsystem"
                    }
                }
                1
                {
                    Write-Host "Enabling [~] Linux Subsystem"
                    $This.Output[0] | ? State -ne Enabled | % {

                        Enable-WindowsOptionalFeature -Online -FeatureName $This.Feature() -NoRestart
                    }

                    If (!!$?)
                    {
                        $This.Output[0].State = "Enabled"
                    }
                }
                2
                {
                    Write-Host "Disabling [~] Linux Subsystem"
                    $This.Output[0] | ? State -eq Enabled | % {

                        Disable-WindowsOptionalFeature -Online -FeatureName $This.Feature() -NoRestart
                    }

                    If (!!$?)
                    {
                        $This.Output[0].State = "Disabled"
                    }
                }
            }
        }
        Else
        {
            Write-Host "Error [!] This version of Windows does not support (WSL/Windows Subsystem for Linux)"
        }
    }
}
```

_____/
_____/ Class [LinuxSubsystem : ControlTemplate]
Enum [WindowsAppsType] /_____\
/_____

```
Enum WindowsAppsType
{
    OneDrive
    OneDriveInstall
    XboxDVR
    MediaPlayer
    WorkFolders
    FaxAndScan
    LinuxSubsystem
}
```

_____/

Enum [WindowsAppsType]

Class [WindowsAppsList]

```
Class WindowsAppsList
{
    [Object] $Features = [WindowsOptionalFeatures]::New().Output
    [Object]   $Output
    WindowsAppsList()
    {
        $This.Output      = @( )
        ForEach ($Name in [System.Enum]::GetNames([WindowsAppsType]))
        {
            $Item         = Switch ($Name)
            {
                OneDrive        { [OneDrive]::New()                     }
                OneDriveInstall { [OneDriveInstall]::New()              }
                XboxDVR         { [XboxDVR]::New()                      }
                MediaPlayer     { [MediaPlayer]::New($This.Features)    }
                WorkFolders     { [WorkFolders]::New($This.Features)    }
                FaxAndScan      { [FaxAndScan]::New($This.Features)     }
                LinuxSubsystem  { [LinuxSubsystem]::New($This.Features) }
            }
            $Item.Source  = "WindowsApps"
            $This.Output += $Item
        }
    }
}
```

Class [WindowsAppsList]

Enum [WindowsOptionalStateType]

```
Enum WindowsOptionalStateType
{
    Disabled
    DisabledWithPayloadRemoved
    Enabled
}
```

Enum [WindowsOptionalStateType]

Class [WindowsOptionalStateSlot]

```
Class WindowsOptionalStateSlot
{
    [UInt32] $Index
    [String] $Type
    [String] $Description
    WindowsOptionalStateSlot([String]$Type)
    {
        $This.Type = [WindowsOptionalStateType]::$Type
        $This.Index = [UInt32][WindowsOptionalStateType]::$Type
    }
    [String] ToString()
    {
        Return $This.Type
    }
}
```

Class [WindowsOptionalStateSlot]

Class [WindowsOptionalStateList]

```
Class WindowsOptionalStateList
{
    [Object] $Output
```

```
    WindowsOptionalStateList()
    {
        $This.Output = @( )
        [System.Enum]::GetNames([WindowsOptionalStateType]) | % { $This.Add($_) }
    }
    Add([String]$Name)
    {
        $Item             = [WindowsOptionalStateSlot]::New($Name)
        $Item.Description = Switch ($Name)
        {
            Disabled                  { "Feature is disabled"                   }
            DisabledWithPayloadRemoved { "Feature is disabled, payload is removed" }
            Enabled                   { "Feature is enabled"                    }
        }
        $This.Output += $Item
    }
    [Object] Get([String]$Type)
    {
        Return $This.Output | ? Type -eq $Type
    }
}
```

_____/
_____/ Class [WindowsOptionalStateList]
Class [WindowsOptionalFeature] /_____\
/_____

```
Class WindowsOptionalFeature
{
    [UInt32] $Index
    [String] $FeatureName
    [Object] $State
    Hidden [String] $Path
    Hidden [UInt32] $Online
    Hidden [String] $WinPath
    Hidden [String] $SysDrivePath
    Hidden [UInt32] $RestartNeeded
    Hidden [String] $LogPath
    Hidden [String] $ScratchDirectory
    Hidden [String] $LogLevel
    WindowsOptionalFeature([UInt32]$Index,[Object]$List,[Object]$Object)
    {
        $This.Index            = $Index
        $This.FeatureName      = $Object.FeatureName
        $This.State            = $List.Get($Object.State)
        $This.Path             = $Object.Path
        $This.Online           = $Object.Online
        $This.WinPath          = $Object.WinPath
        $This.SysDrivePath     = $Object.SysDrivePath
        $This.RestartNeeded    = $Object.RestartNeeded
        $This.LogPath          = $Object.LogPath
        $This.ScratchDirectory = $Object.ScratchDirectory
        $This.LogLevel         = $Object.LogLevel
    }
}
```

_____/
_____/ Class [WindowsOptionalFeature]
Class [WindowsOptionalFeatures] /_____\
/_____

```
Class WindowsOptionalFeatures
{
    [Object] $State
    [Object] $Output
    WindowsOptionalFeatures()
    {
        $This.State  = [WindowsOptionalStateList]::New()
        $This.Output = @( )
        Get-WindowsOptionalFeature -Online | Sort-Object FeatureName | % {
```

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

_____/
_____/ Class [WindowsOptionalFeatures]
Class [AppXTemplate] /_____\
/_____

```
Class AppXTemplate
{
    [String] $AppXName
    [String] $CName
    [String] $Varname
    AppXTemplate([String]$Line)
    {
        $Split = $Line.Split("/")
        $This.AppXName = $Split[0]
        $This.CName    = $Split[1]
        $This.Varname  = $Split[2]
    }
}
```

_____/
_____/ Class [AppXTemplate]
Class [AppXProfile] /_____\
/_____

```
Class AppXProfile
{
    [Object] $Output
    AppXProfile()
    {
        $This.Output = $This.Defaults() | % { [AppXTemplate]$_ }
    }
    [String[]] Defaults()
    {
        $Item = "{0}.3DBuilder/3DBuilder/APP_3DBuilder;{0}.{0}3DViewer/3DViewer/APP_3DViewer;{0}.BingWeather" +
        "/Bing Weather/APP_BingWeather;{0}.CommsPhone/Phone/APP_CommsPhone;{0}.{1}communicationsapps/Calenda" +
        "r & Mail/APP_Communications;{0}.GetHelp/{0}s Self-Help/APP_GetHelp;{0}.Getstarted/Get Started Link/" +
        "APP_Getstarted;{0}.Messaging/Messaging/APP_Messaging;{0}.{0}OfficeHub/Get Office Link/APP_{0}OffHub" +
        ";{0}.MovieMoments/Movie Moments/APP_MovieMoments;4DF9E0F8.Netflix/Netflix/APP_Netflix;{0}.Office.On" +
        "eNote/Office OneNote/APP_OfficeOneNote;{0}.Office.Sway/Office Sway/APP_OfficeSway;{0}.OneConnect/On" +
        "e Connect/APP_OneConnect;{0}.People/People/APP_People;{0}.{1}.Photos/Photos/APP_Photos;{0}.SkypeApp" +
        "/Skype/APP_SkypeApp1;{0}.{0}SolitaireCollection/{0} Solitaire/APP_SolitaireCollect;{0}.{0}StickyNot" +
        "es/Sticky Notes/APP_StickyNotes;{0}.{1}SoundRecorder/Voice Recorder/APP_VoiceRecorder;{0}.{1}Alarms" +
        "/Alarms and Clock/APP_{1}Alarms;{0}.{1}Calculator/Calculator/APP_{1}Calculator;{0}.{1}Camera/Camera" +
        "/APP_{1}Camera;{0}.{1}Feedback/{1} Feedback/APP_{1}Feedbak1;{0}.{1}FeedbackHub/{1} Feedback Hub/APP" +
        "_{1}Feedbak2;{0}.{1}Maps/Maps/APP_{1}Maps;{0}.{1}Phone/Phone Companion/APP_{1}Phone;{0}.{1}Store/{0}" +
        "} Store/APP_{1}Store;{0}.Wallet/Stores Credit and Debit Card Information/APP_{1}Wallet;{0}.Xbox.TCU" +
        "I/Xbox Title-callable UI/App_XboxTCUI;{0}.XboxApp/Xbox App for {1} PC/App_XboxApp;{0}.XboxGameOverl" +
        "ay/Xbox In-Game Overlay/App_XboxGameOverlay;{0}.XboxGamingOverlay/Xbox Gaming Overlay UI/App_XboxGa" +
        "mingOverlay;{0}.XboxIdentityProvider/Xbox Identity Provider/App_XboxIdentityProvider;{0}.XboxSpeech" +
        "toTextOverlay/Xbox Speech-to-Text UI/App_XboxSpeechToText;{0}.ZuneMusic/Groove Music/APP_ZuneMusic;" +
        "{0}.ZuneVideo/Groove Video/APP_ZuneVideo;"

        Return $Item -f "Microsoft","Windows"
    }
}
```

_____/
_____/ Class [AppXProfile]
Class [AppXObject] /_____\
/_____

```
Class AppXObject
{
    [UInt32]           $Index
    [UInt32]         $Profile
```

```
    Hidden [String]     $CName
    Hidden [String]   $VarName
    [Version]         $Version
    [String]      $PackageName
    [String]      $DisplayName
    [String]      $PublisherID
    [UInt32]     $MajorVersion
    [UInt32]     $MinorVersion
    [UInt32]            $Build
    [UInt32]         $Revision
    [UInt32]     $Architecture
    [String]       $ResourceID
    [String]  $InstallLocation
    [Object]          $Regions
    [String]             $Path
    [UInt32]           $Online
    [String]           $WinPath
    [string]     $SysDrivePath
    [UInt32]    $RestartNeeded
    [String]          $LogPath
    [String] $ScratchDirectory
    [String]         $LogLevel
    [Int32]              $Slot
    AppXObject([UInt32]$Index,[Object]$AppXProfile,[Object]$Object)
    {
        $This.Index            = $Index
        $This.Version          = $Object.Version
        $This.PackageName      = $Object.PackageName
        $This.DisplayName      = $Object.DisplayName
        $This.PublisherId      = $Object.PublisherId
        $This.MajorVersion     = $Object.MajorVersion
        $This.MinorVersion     = $Object.MinorVersion
        $This.Build            = $Object.Build
        $This.Revision         = $Object.Revision
        $This.Architecture     = $Object.Architecture
        $This.ResourceId       = $Object.ResourceId
        $This.InstallLocation  = $Object.InstallLocation
        $This.Regions          = $Object.Regions
        $This.Path             = $Object.Path
        $This.Online           = $Object.Online
        $This.WinPath          = $Object.WinPath
        $This.SysDrivePath     = $Object.SysDrivePath
        $This.RestartNeeded    = $Object.RestartNeeded
        $This.LogPath          = $Object.LogPath
        $This.ScratchDirectory = $Object.ScratchDirectory
        $This.LogLevel         = $Object.LogLevel

        If ($Object.DisplayName -in $AppXProfile.AppXName)
        {
            $Item              = $AppXProfile | ? AppXName -match $This.DisplayName
            $This.Profile      = 1
            $This.CName        = $Item.CName
            $This.VarName      = $Item.VarName
            $This.Slot         = 0
        }
        Else
        {
                $This.Profile      = 0
                $This.Slot         = -1
        }
    }
}
```

_____/
_____/ Class [AppXObject]
Class [AppXList] /_____\

```
Class AppXList
{
    [Object] $Profile = [AppXProfile]::New().Output
    [Object] $Output
```

```
    AppXList()
    {
        $This.Output    = @( )
        Get-AppxProvisionedPackage -Online | % {
            $This.Output += [AppXObject]::New($This.Output.Count,$This.Profile,$_) }
    }
}
```

Class [AppXList]

Class [DisableVariousTasks]

This class is not implemented (currently/yet).

```
Class DisableVariousTasks
{
    [UInt32] $Mode
    [Object] $Output
    DisableVariousTasks()
    {
        $This.Output = @()
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped,[Object[]]$TaskList)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
                    Write-Host "Skipping [!] Various Scheduled Tasks"
                }
            }
            1
            {
                Write-Host "Enabling [~] Various Scheduled Tasks"
                $TaskList | % { Get-ScheduledTask -TaskName $_ | Enable-ScheduledTask }
            }
            2
            {
                Write-Host "Disabling [~] Various Scheduled Tasks"
                $TaskList | % { Get-ScheduledTask -TaskName $_ | Disable-ScheduledTask }
            }
        }
    }
}
```

Class [DisableVariousTasks]

Class [ScreenSaverWaitTime]

This class is not implemented (currently/yet).

```
Class ScreenSaverWaitTime
{
    [UInt32] $Mode
    [Object] $Output
    ScreenSaverWaitTime()
    {
        $This.Output = @([Registry]::New('HKLM:\Software\Policies\Microsoft\Windows','ScreensaveTimeout'))
    }
    SetMode([UInt32]$Mode,[UInt32]$ShowSkipped)
    {
        Switch ($Mode)
        {
            0
            {
                If ($ShowSkipped)
                {
```

```
                    Write-Host "Skipping [!] ScreenSaver Wait Time"
                }
            }
            1
            {
                Write-Host "Enabling [~] ScreenSaver Wait Time"
            }
            2
            {
                Write-Host "Disabling [~] ScreenSaver Wait Time"
            }
        }
    }
}
```

_____/ Class [ScreenSaverWaitTime]

Class [SystemControl] /

So, the idea behind this class is to act as a class factory, and it will assemble all of the above classes that pertain to individual “Source” objects. It is pretty simple to understand where these classes attain some level of relevance to one another.

The REAL idea here, is to allow the GUI to be able to control all of the objects above, through this class. Simply put, using the method `SetMode([UInt32]$Mode)` is going to VARY, dependent on the control object.

If I want to apply the settings for each individual object selected in the GUI...?
Then... filtering the objects will be incredibly easy to do, and further to that point...?
So will resetting the object back to their default settings.

Simply put, there is a high level of SCALABILITY and CONTROL, as well as ROOM FOR EXPANSION, with this approach.

```
Class SystemControl
{
    [Object]                    $Output
    [Object]                   $Feature
    [Object]                      $AppX
    SystemControl()
    {
        $This.Reset()
    }
    Reset()
    {
        $This.Output     = @( )
        $This.Output    += [PrivacyList]::New().Output
        $This.Output    += [WindowsUpdateList]::New().Output
        $This.Output    += [ServiceList]::New().Output
        $This.Output    += [ContextList]::New().Output
        $This.Output    += [TaskBarList]::New().Output
        $This.Output    += [StartMenuList]::New().Output
        $This.Output    += [ExplorerList]::New().Output
        $This.Output    += [ThisPCIconList]::New().Output
        $This.Output    += [DesktopIconList]::New().Output
        $This.Output    += [LockScreenList]::New().Output
        $This.Output    += [MiscellaneousList]::New().Output
        $This.Output    += [PhotoViewerList]::New().Output
        $This.Output    += [WindowsAppsList]::New().Output
        $This.Feature    = [WindowsOptionalFeatures]::New().Output
        $This.AppX       = [AppXList]::New().Output
    }
    [Void] Toggle([Object]$Item)
    {
        $Item = Switch ($Item)
        {
            0 { 1 }
            1 { 0 }
        }
    }
}
```

_____/ Class [SystemControl]

Enum [XboxType]

```
Enum XboxType
{
    XblAuthManager
    XblGameSave
    XboxNetApiSvc
    XboxGipSvc
    xbgm
}
```

Enum [NetTCPType]

```
Enum NetTCPType
{
    NetMsmqActivator
    NetPipeActivator
    NetTcpActivator
}
```

Enum [PidType]

```
Enum PidType
{
    BcastDVRUserService
    DevicePickerUserSvc
    DevicesFlowUserSvc
    PimIndexMaintenanceSvc
    PrintWorkflowUserSvc
    UnistoreSvc
    UserDataSvc
    WpnUserService
}
```

Enum [SkipType]

```
Enum SkipType
{
    AppXSVC
    BrokerInfrastructure
    ClipSVC
    CoreMessagingRegistrar
    DcomLaunch
    EntAppSvc
    gpsvc
    LSM
    MpsSvc
    msiserver
    NgcCtnrSvc
    NgcSvc
    RpcEptMapper
    RpcSs
    Schedule
    SecurityHealthService
    sppsvc
    StateRepository
    SystemEventsBroker
    tiledatamodelsvc
    WdNisSvc
    WinDefend
```

```
}
```

_____/ Enum [SkipType]
Class [SkipItem] /_____\

In MadBomb122's program, [https://github.com/madbomb122/BlackViperScript], there are several options pertaining to skipping services and whatnot. Those are able to be seen in the GUI, though CURRENTLY, they are not rigged.

That is mainly because (1) of (2) things... (1) either I have to implement things that require testing, or (2) MadBomb122 might be able to implement his knowledge into the script so that it achieves feature parity with his original script.

```
Class SkipItem
{
    [String] $Type
    [String] $Name
    SkipItem([String]$Type,[String]$Name)
    {
        $This.Type     = $Type
        $This.Name     = $Name
    }
}
```

_____/ Class [SkipItem]
Class [SkipList] /_____\

This is strictly meant for cataloging the services that will be able to be (filtered/skipped).
As it currently stands, I have additional ideas on how to make this portion of the utility look even cooler.

```
Class SkipList
{
    [Object] $Output
    SkipList()
    {
        $This.Output = @( )
        $ProcessID   = (Get-Service | ? ServiceType -eq 224)[-1].Name.Split("_")[1]

        ForEach ($Item in [System.Enum]::GetNames([XboxType]))
        {
            $This.SkipItem("Xbox",$Item)
        }

        ForEach ($Item in [System.Enum]::GetNames([NetTCPType]))
        {
            $This.SkipItem("NetTCP",$Item)
        }

        ForEach ($Item in [System.Enum]::GetNames([PidType]))
        {
            $This.SkipItem( "Skip","$Item`_$ProcessID")
        }

        ForEach ($Item in [System.Enum]::GetNames([SkipType]))
        {
            $This.SkipItem( "Skip",$Item)
        }
    }
    SkipItem([String]$Type,[String]$Name)
    {
        $This.Output += [SkipItem]::New($Type,$Name)
    }
}
```

_____/ Class [SkipList]

Enum [ProfileType]

So, the original utility features Charles 'Black Viper' Sparks' service configuration which WAS a (*.csv) based file, however I have implemented various additions and included it into the script.

The idea is to have all of the components available and able to be updated on the fly, like, automatically.

Perhaps there may be a Windows service configuration samurai out there, somewhere...
Nobody knows if that's a real person, or a group of people, or not...
But, if they are a real (person/group of people)...?
Then it stands to reason...
These samurais practice the holy sacred art, of knowing what the hell they're (saying/doing).

Occasionally, this samurai may have to come out of a shadowy alcove, whip out the (nunchakus/facthoods), and tell any potential falsehoods: “There's the door...”

This enumeration type specifically deals with profiles that are geared toward specific machines, and it should be SOMEWHAT self-explanatory.

```
Enum ProfileType
{
    HomeMax
    HomeMin
    ProMax
    ProMin
    DesktopSafeMax
    DesktopSafeMin
    DesktopTweakedMax
    DesktopTweakedMin
    LaptopSafeMax
    LaptopSafeMin
}
```

Class [ProfileSlot]

Each item in the enumeration list above is converted into a slot object, with a numerical index and description.

```
Class ProfileSlot
{
    [UInt32] $Index
    [String] $Type
    [String] $Description
    ProfileSlot([String]$Type)
    {
        $This.Type  = [ProfileType]::$Type
        $This.Index = [UInt32][ProfileType]::$Type
    }
    [String] ToString()
    {
        Return $This.Type
    }
}
```

Class [ProfileList]

Then, all of those (slot objects/forces) are combined like:
1) Voltron, 2) Captain Planet, or 3) the Power Rangers' Megazord...

```
Class ProfileList
{
    [Object] $Output
    ProfileList()
    {
        $This.Output = @( )
```

```
        ForEach ($Name in [System.Enum]::GetNames([ProfileType]))
        {
            $Item = [ProfileSlot]::New($Name)
            $Item.Description = Switch ($Name)
            {
                HomeMax
                {
                    "Windows (10|11) Home, Default/Maximum"
                }
                HomeMin
                {
                    "Windows (10|11) Home, Default/Minimum"
                }
                ProMax
                {
                    "Windows (10|11) Pro, Default/Maximum"
                }
                ProMin
                {
                    "Windows (10|11) Pro, Default/Minimum"
                }
                DesktopSafeMax
                {
                    "Desktop (General), Safe Maximum"
                }
                DesktopSafeMin
                {
                    "Desktop (General), Safe Minimum"
                }
                DesktopTweakedMax
                {
                    "Desktop (General), Tweaked Maximum"
                }
                DesktopTweakedMin
                {
                    "Desktop (General), Tweaked Minimum"
                }
                LaptopSafeMax
                {
                    "Laptop (General), Safe Maximum"
                }
                LaptopSafeMin
                {
                    "Laptop (General), Safe Minimum"
                }
            }
            $This.Output += $Item
        }
    }
}
```

Class [ProfileList]

Class [ViperBombConfig]

Technically, this class may not even be necessary, however it helps to provide some level of control for the GUI to have default values and such. The way that this is currently set up, is similar to a STRUCT in C#/C++.

```
Class ViperBombConfig
{
    [UInt32]        $BypassBuild = 0
    [UInt32]      $BypassEdition = 0
    [UInt32]       $BypassLaptop = 0
    [UInt32]      $DisplayActive = 1
    [UInt32]    $DisplayInactive = 1
    [UInt32]     $DisplaySkipped = 1
    [UInt32]       $MiscSimulate = 0
    [UInt32]           $MiscXbox = 1
    [UInt32]         $MiscChange = 0
    [UInt32]   $MiscStopDisabled = 0
    [UInt32]          $DevErrors = 0
```

```
    [UInt32]             $DevLog = 0
    [UInt32]         $DevConsole = 0
    [UInt32]          $DevReport = 0
    [String]    $LogServiceLabel = "Service.log"
    [String]     $LogScriptLabel = "Script.log"
    [String]           $RegLabel = "Backup.reg"
    [String]           $CsvLabel = "Backup.csv"
    [Object]            $Filter = [SkipList]::New().Output
    [Object]           $Profile = [ProfileList]::New().Output
}
```

Class [ViperBombConfig]

Class [ViperBombTime]

```
# // ------------------------------------------------------
# // | Used to track console logging, similar to Stopwatch |
# // ------------------------------------------------------

Class ViperBombTime
{
    [String]   $Name
    [DateTime] $Time
    [UInt32]    $Set
    ViperBombTime([String]$Name)
    {
        $This.Name = $Name
        $This.Time = [DateTime]::MinValue
        $This.Set  = 0
    }
    Toggle()
    {
        $This.Time = [DateTime]::Now
        $This.Set  = 1
    }
    [String] ToString()
    {
        Return $This.Time.ToString()
    }
}
```

Class [ViperBombTime]

Class [ViperBombStatus]

```
# // ---------------------------------------
# // | Single object that displays a status |
# // ---------------------------------------

Class ViperBombStatus
{
    [UInt32]   $Index
    [String] $Elapsed
    [Int32]    $State
    [String]  $Status
    ViperBombStatus([UInt32]$Index,[String]$Time,[Int32]$State,[String]$Status)
    {
        $This.Index   = $Index
        $This.Elapsed = $Time
        $This.State   = $State
        $This.Status  = $Status
    }
    [String] ToString()
    {
        Return "[{0}] (State: {1}/Status: {2})" -f $This.Elapsed, $This.State, $This.Status
    }
}
```

Class [ViperBombStatus]

Class [ViperBombStatusBank]

```
# // ------------------------------------------------------------------------
# // | A collection of status objects, uses itself to create/update messages |
# // ------------------------------------------------------------------------

Class ViperBombStatusBank
{
    [Object]    $Start
    [Object]      $End
    [String]     $Span
    [Object]   $Status
    [Object]   $Output
    ViperBombStatusBank()
    {
        $This.Reset()
    }
    [String] Elapsed()
    {
        Return @(Switch ($This.End.Set)
        {
            0 { [Timespan]([DateTime]::Now-$This.Start.Time) }
            1 { [Timespan]($This.End.Time-$This.Start.Time) }
        })
    }
    [Void] SetStatus()
    {
        $This.Status = [ViperBombStatus]::New($This.Output.Count,
                                              $This.Elapsed(),
                                              $This.Status.State,
                                              $This.Status.Status)
    }
    [Void] SetStatus([Int32]$State,[String]$Status)
    {
        $This.Status = [ViperBombStatus]::New($This.Output.Count,
                                              $This.Elapsed(),
                                              $State,
                                              $Status)
    }
    Initialize()
    {
        If ($This.Start.Set -eq 1)
        {
            $This.Update(-1,"Start [!] Error: Already initialized, try a different operation or reset.")
        }
        $This.Start.Toggle()
        $This.Update(0,"Running [~] ($($This.Start))")
    }
    Finalize()
    {
        If ($This.End.Set -eq 1)
        {
            $This.Update(-1,"End [!] Error: Already initialized, try a different operation or reset.")
        }
        $This.End.Toggle()
        $This.Span = $This.Elapsed()
        $This.Update(100,"Complete [+] ($($This.End)), Total: ($($This.Span))")
    }
    Reset()
    {
        $This.Start  = [ViperBombTime]::New("Start")
        $This.End    = [ViperBombTime]::New("End")
        $This.Span   = $Null
        $This.Status = $Null
        $This.Output = [System.Collections.ObjectModel.ObservableCollection[Object]]::New()
    }
    Write()
    {
        $This.Output.Add($This.Status)
    }
    [Object] Update([Int32]$State,[String]$Status)
```

```
    {
        $This.SetStatus($State,$Status)
        $This.Write()
        Return $This.Last()
    }
    [Object] Current()
    {
        $This.Update($This.Status.State,$This.Status.Status)
        Return $This.Last()
    }
    [Object] Last()
    {
        Return $This.Output[$This.Output.Count-1]
    }
    [String] ToString()
    {
        If (!$This.Span)
        {
            Return $This.Elapsed()
        }
        Else
        {
            Return $This.Span
        }
    }
}
```

Class [ViperBombController] Class [ViperBombStatusBank]

```
Class ViperBombController
{
    [Object] $Console
    [Object] $Module
    [Object] $System
    [Object] $Xaml
    [String] $Caption
    [String] $Platform
    [String] $PSVersion
    [String] $Type
    [Object] $HotFix
    [Object] $Config
    [Object] $Service
    [Object] $Control
    ViperBombController()
    {
        $This.Console     = $This.StatusBank()
        $This.Update(0,"Loading [~] $($This.Label())")

        Import-Module FightingEntropy
        $This.Module      = Get-FEModule -Mode 1

        $This.Update(0,"Loading [~] AppX")
        Import-Module AppX

        $This.System      = $This.GetSystemDetails()

        $This.System.ComputerSystem | % { $_.Memory = "{0} GB" -f [Regex]::Matches($_.Memory,"\d+\.\d{2}").Value
}
        $This.Caption     = $This.Module.OS.Caption
        $This.Platform    = $This.Module.OS.Platform
        $This.PSVersion   = $This.Module.OS.PSVersion
        $This.Type        = $This.Module.OS.Type

        $This.HotFix      = $This.GetHotFix()

        $This.Config      = $This.GetConfig()
        $This.Control     = $This.GetSystemControl()
        $This.Xaml        = $This.GetViperBombXaml()
    }
```

```
    [Void] Reset([Object]$xSender,[Object]$Content)
    {
        $xSender.Items.Clear()
        ForEach ($Object in $Content)
        {
            [Void]$xSender.Items.Add($Object)
        }
    }
    [Object] GetViperBombXaml()
    {
        $This.Update(0,"Gathering [~] Xaml Interface")
        Return [XamlWindow][ViperBombXaml]::Content
    }
    [Object] GetSystemControl()
    {
        $This.Update(0,"Gathering [~] System Control")
        Return [SystemControl]::New()
    }
    [Object] GetServices()
    {
        $This.Update(0,"Gathering [~] System/Services")
        Return [ServiceControl]::New()
    }
    [Object] GetConfig()
    {
        $This.Update(0,"Gathering [~] Default/Config")
        Return [ViperBombConfig]::New()
    }
    [Object] GetSystemDetails()
    {
        $This.Update(0,"Gathering [~] System/Details")
        Return Get-SystemDetails
    }
    [Object] GetHotFix()
    {
        $This.Update(0,"Gathering [~] OS/Hotfix Details")
        Return [HotFixList]::New()
    }
    [String] Label()
    {
        Return "[FightingEntropy($([Char]960))][System Control Extension Utility]"
    }
    [Object] StatusBank()
    {
        $Item = [ViperBombStatusBank]::New()
        $Item.Initialize()
        Return $Item
    }
    Update([UInt32]$Mode,[String]$State)
    {
        $This.Console.Update($Mode,$State)
        Write-Host $This.Console.Last()
    }
    SetSlot([UInt32]$Slot)
    {
        If (!$This.Service)
        {
            $This.Service = $This.GetServices()
        }

        $This.Service.SetSlot($Slot)
        $This.Reset($This.Xaml.IO.Service,$This.Service.Output)
    }
    StageXamlEvent()
    {
        # [Provide alternate variable for event handlers]
        $Ctrl            = $This

        ######################### First Tab ##############################
        # [Module OS items]
        $Ctrl.Reset($Ctrl.Xaml.IO.OS,$Ctrl.Module.OS)
```

```powershell
        # [Bios Information]
        $Ctrl.Reset($Ctrl.Xaml.IO.BiosInformation,$Ctrl.System.BiosInformation)

        # [Bios Information Extension]
        $Object = ForEach ($Property in "SmBiosPresent","SmBiosVersion","SmBiosMajor","SmBiosMinor",
        "SystemBiosMajor","SystemBiosMinor")
        {
            [PSNoteProperty]::New($Property,$Ctrl.System.BiosInformation.$Property)
        }
        $Ctrl.Reset($Ctrl.Xaml.IO.BiosInformationExtension,$Object)

        # [Operating System]
        $Ctrl.Reset($Ctrl.Xaml.IO.OperatingSystem,$Ctrl.System.OperatingSystem)

        # [Hot Fix]
        $Ctrl.Reset($Ctrl.Xaml.IO.HotFix,$Ctrl.HotFix.Output)

        # [Computer System]
        $Ctrl.Reset($Ctrl.Xaml.IO.ComputerSystem,$Ctrl.System.ComputerSystem)

        # [Computer System Extension]
        $Object = ForEach ($Property in "UUID","Chassis","BiosUefi","AssetTag")
        {
            [PSNoteProperty]::New($Property,$Ctrl.System.ComputerSystem.$Property)
        }
        $Ctrl.Reset($Ctrl.Xaml.IO.ComputerSystemExtension,$Object)

        # [Processor]
        $Ctrl.Reset($Ctrl.Xaml.IO.Processor,$Ctrl.System.Processor.Output)

        # [Processor Event Trigger(s)]
        $Ctrl.Xaml.IO.Processor.Add_SelectionChanged(
        {
            $Index = $Ctrl.Xaml.IO.Processor.SelectedIndex
            If ($Index -ne -1)
            {
                $Object = ForEach ($Property in "ProcessorId","DeviceId","Speed","Cores","Used",
                "Logical","Threads")
                {
                    [PSNoteProperty]::New($Property,$Ctrl.System.Processor.Output[$Index].$Property)
                }
                $Ctrl.Reset($Ctrl.Xaml.IO.ProcessorExtension,$Object)
            }
        })

        # [Disk]
        $Ctrl.Reset($Ctrl.Xaml.IO.Disk,$Ctrl.System.Disk.Output)

        # [Disk Event Trigger(s)]
        $Ctrl.Xaml.IO.Disk.Add_SelectionChanged(
        {
            $Index = $Ctrl.Xaml.IO.Disk.SelectedIndex
            If ($Index -ne -1)
            {
                # [Disk Extension]
                $Object = ForEach ($Property in "HealthStatus","BusType","UniqueId","Location")
                {
                    [PSNoteProperty]::New($Property,$Ctrl.System.Disk.Output[$Index].$Property)
                }

                $Ctrl.Reset($Ctrl.Xaml.IO.DiskExtension,$Object)

                # [Disk Partition(s)]
                $Ctrl.Reset($Ctrl.Xaml.IO.DiskPartition,$Ctrl.System.Disk.Output[$Index].Partition.Output)

                # [Disk Volume(s)]
                $Ctrl.Reset($Ctrl.Xaml.IO.DiskVolume,$Ctrl.System.Disk.Output[$Index].Volume.Output)
            }
        })

        # [Network]
        $Ctrl.Reset($Ctrl.Xaml.IO.Network,$Ctrl.System.Network.Output)
```

```powershell
# [Network Event Trigger(s)]
$Ctrl.Xaml.IO.Network.Add_SelectionChanged(
{
    $Index = $Ctrl.Xaml.IO.Network.SelectedIndex
    If ($Index -ne -1)
    {
        $Object = ForEach ($Property in "IPAddress","SubnetMask","Gateway","DnsServer",
        "DhcpServer","MacAddress")
        {
            [PSNoteProperty]::New($Property,$Ctrl.System.Network.Output[$Index].$Property)
        }

        $Ctrl.Reset($Ctrl.Xaml.IO.NetworkExtension,$Object)
    }
})

######################## Second Tab ##############################
$Ctrl.Reset($Ctrl.Xaml.IO.ServiceSlot,@(0..9))

$Ctrl.Xaml.IO.ServiceSlot.Add_SelectionChanged(
{
    $Index = $Ctrl.Xaml.IO.ServiceSlot.SelectedIndex
    $Ctrl.Xaml.IO.ServiceDisplay.Items.Clear()
    $Ctrl.Xaml.IO.ServiceDisplay.Items.Add($Ctrl.Config.Profile[$Index])
    $Ctrl.SetSlot($Ctrl.Xaml.IO.ServiceSlot.SelectedIndex)
})

$Name = Switch -Regex ($Ctrl.System.OperatingSystem.Caption)
{
    Home { "HomeMax" } "(Pro|Server)" { "ProMax" }
}
$Ctrl.Xaml.IO.ServiceSlot.SelectedIndex = $Ctrl.Config.Profile | ? Type -eq $Name | % Index

$Ctrl.Xaml.IO.ServiceGet.Add_Click(
{
    $Ctrl.SetSlot($Ctrl.Xaml.IO.ServiceSlot.SelectedIndex)
})

$Ctrl.Xaml.IO.Service.Add_SelectionChanged(
{
    $Slot = $Ctrl.Xaml.IO.Service.SelectedItem
    If ($Slot.Index -ne -1)
    {
        $Object = ForEach ($Property in "Name","DisplayName","PathName","Description")
        {
            [PSNoteProperty]::New($Property,$Slot.$Property)
        }
        $Ctrl.Reset($Ctrl.Xaml.IO.ServiceExtension,$Object)
    }
})

$Ctrl.Xaml.IO.ServiceFilter.Add_TextChanged(
{
    Start-Sleep -Milliseconds 50
    $Property  = $Ctrl.Xaml.IO.ServiceProperty.SelectedItem.Content
    $Text      = $Ctrl.Xaml.IO.ServiceFilter.Text
    $List      = Switch -Regex ($Text)
    {
        ""
        {
            $Ctrl.Service.Output | ? $Property -match $Text
        }
        Default
        {
            $Ctrl.Service.Output
        }
    }

    $Ctrl.Reset($Ctrl.Xaml.IO.Service,$List)
})
```

```powershell
        ################## Third Tab ##########################
        # [Control Subtab]
        $Ctrl.Reset($Ctrl.Xaml.IO.ControlOutput,$Ctrl.Control.Output)

        $Ctrl.Xaml.IO.ControlOutput.Add_SelectionChanged(
        {
            $Index = $Ctrl.Xaml.IO.ControlOutput.SelectedIndex
            $Ctrl.Xaml.IO.ControlOutputExtension.Items.Clear()
            If ($Index -ne -1)
            {
                $Object = ForEach ($Property in "Name","DisplayName","Value","Description")
                {
                    [PSNoteProperty]::New($Property,$Ctrl.Xaml.IO.ControlOutput.SelectedItem.$Property)
                }

                $Ctrl.Reset($Ctrl.Xaml.IO.ControlOutputExtension,$Object)
            }
        })

        $Ctrl.Xaml.IO.ControlSlot.Add_SelectionChanged(
        {
            $Slot = $Ctrl.Xaml.IO.ControlSlot.SelectedItem.Content
            $List = Switch ($Slot)
            {
                Default
                {
                    $Ctrl.Control.Output | ? Source -eq $Slot
                }
                All
                {
                    $Ctrl.Control.Output
                }
            }

            $Ctrl.Reset($Ctrl.Xaml.IO.ControlOutput,$List)
        })

        $Ctrl.Xaml.IO.ControlFilter.Add_TextChanged(
        {
            Start-Sleep -Milliseconds 50
            $Property = $Ctrl.Xaml.IO.ControlProperty.SelectedItem.Content
            $Text     = $Ctrl.Xaml.IO.ControlFilter.Text
            $List     = Switch -Regex ($Text)
            {
                ""
                {
                    $Ctrl.Control.Output | ? $Property -match $Text
                }
                Default
                {
                    $Ctrl.Control.Output
                }
            }

            $Ctrl.Reset($Ctrl.Xaml.IO.ControlOutput,$List)
        })

        # [WindowsFeatures Subtab]
        $Ctrl.Reset($Ctrl.Xaml.IO.ControlFeature,$Ctrl.Control.Feature)

        $Ctrl.Xaml.IO.ControlFeature.Add_SelectionChanged(
        {
            $Index = $Ctrl.Xaml.IO.ControlFeature.SelectedIndex
            If ($Index -ne -1)
            {
                $Object = ForEach ($Property in "Index","FeatureName","State","Path","Online",
                "WinPath","SysDrivePath","RestartNeeded","LogPath","ScratchDirectory","LogLevel")
                {
                    [PSNoteProperty]::New($Property,$Ctrl.Xaml.IO.ControlFeature.SelectedItem.$Property)
                }

                $Ctrl.Reset($Ctrl.Xaml.IO.ControlFeatureExtension,$Object)
```

```
            }
        })

        $Ctrl.Xaml.IO.ControlFeatureFilter.Add_TextChanged(
        {
            Start-Sleep -Milliseconds 50
            $Property = $Ctrl.Xaml.IO.ControlFeatureProperty.SelectedItem.Content
            $Text     = $Ctrl.Xaml.IO.ControlFeatureFilter.Text
            $List     = Switch -Regex ($Text)
            {
                ""
                {
                    $Ctrl.Control.Feature | ? $Property -match $Text
                }
                Default
                {
                    $Ctrl.Control.Feature
                }
            }

            $Ctrl.Reset($Ctrl.Xaml.IO.ControlFeature,$List)
        })

        # [AppX]
        $Ctrl.Reset($Ctrl.Xaml.IO.ControlAppX,$Ctrl.Control.AppX)

        $Ctrl.Xaml.IO.ControlAppX.Add_SelectionChanged(
        {
            $Index = $Ctrl.Xaml.IO.ControlAppX.SelectedIndex
            $Ctrl.Xaml.IO.ControlAppXExtension.Items.Clear()
            If ($Index -ne -1)
            {
                $Object = ForEach ($Property in "PackageName","DisplayName","PublisherID","Version",
                "Architecture","ResourceID","InstallLocation","RestartNeeded","LogPath","LogLevel")
                {
                    [PSNoteProperty]::New($Property,$Ctrl.Xaml.IO.ControlAppX.SelectedItem.$Property)
                }

                $Ctrl.Reset($Ctrl.Xaml.IO.ControlAppXExtension,$Object)
            }
        })

        $Ctrl.Xaml.IO.ControlAppXFilter.Add_TextChanged(
        {
            Start-Sleep -Milliseconds 50
            $Property = $Ctrl.Xaml.IO.ControlAppXProperty.SelectedItem.Content
            $Text     = $Ctrl.Xaml.IO.ControlAppXFilter.Text
            $List     = Switch -Regex ($Text)
            {
                ""
                {
                    $Ctrl.Control.AppX | ? $Property -match $Text
                }
                Default
                {
                    $Ctrl.Control.AppX
                }
            }

            $Ctrl.Reset($Ctrl.Xaml.IO.ControlAppX,$List)
        })
    }
}
```

_____/ Class [ViperBombController]

Output /_____\

/_____

The function isn't quite complete yet, however, I'll cover the output of the function as it stands.

In its current state, all of those functions up above are wrapped within a function like so...

```
<#
.SYNOPSIS
.DESCRIPTION
.LINK
.NOTES
    ____                                                                                                    ________
   //--\\_____________________________________________________________________________________________________//--\\__//
   \\__//-----------------------------------------------------------------------------------------------------\\__//--\\
   //--\\__[ [FightingEntropy()][2022.11.0] ]_____________________________________________________________//--\\__//
   \\__//-------------------------------------------------------------------------------------------    ---\\
   //---                                                                                                    //
   \\                                                                                                     \\
   //        FileName    : Get-ViperBomb.ps1                                                              //
   \\        Solution    : [FightingEntropy()][2022.11.0]                                                 \\
   //        Purpose     : For managing system details, Windows services, and controls                    //
   \\        Author      : Michael C. Cook Sr.                                                            \\
   //        Contact     : @mcc85s                                                                        //
   \\        Primary     : @mcc85s                                                                        \\
   //        Created     : 2022-10-10                                                                     //
   \\        Modified    : 2022-12-02                                                                     \\
   //        Demo        : N/A                                                                            //
   \\        Version     : 0.0.0 - () - Finalized functional version 1.                                   \\
   //        TODO        : AKA "System Control Extension Utility"                                         //
   \\                                                                                                     \\
   //                                                                                                  ___//
   \\___                                                                                          ___//--\\
   //--\\_____________________________________________________________________________________________//---  ___//
   \\__//-----------------------------------------------------------------------------------------------\\__//---
    ---\\__[ 12-02-2022 13:09:03    ]_____________________________________________________________________//---
        -------------------------------------------------------------------------------------------------
.Example
#>
Function Get-ViperBomb
{
    # <All of those classes above go right here...>

    [ViperBombController]::New()
}
```

Now, normally I would have additional stuff for the class `[ViperBombController]::New()` to do. However, I have to explain (how/why) the function wrapper is ALMOST cosmetic, to show the (input/output) of the code-behind.

```
# [Import the module, in order to override the function Get-ViperBomb]
Import-Module FightingEntropy

# [Run all of that code up above, to override the default function]
# [I'll add a resource link for people to use, to follow along]

# [Cast the function and its' (output/methods/properties) to variable $Ctrl]
$Ctrl = Get-ViperBomb

# [To launch the GUI, these commands stage the event handlers, and invoke it]
$Ctrl.StageXamlEvent()
$Ctrl.Xaml.Invoke()
```

For now, ignore the last (2) actions in the box above, unless you want to see the (GUI/graphical user interface). Since I want to talk about what the code-behind is doing, I want the variable `$Ctrl` to be accessible, AND, to expose all of the (properties + methods) of the `[ViperBombController]` (controller/factory) class... that's what I'll have to do, in order for the (input/output) below, to make any sense.

I'm going to actually drop a resource link right here and now, to a VERY useful article by a dude with a pretty cool sense of humor, and understanding of reality, as well as `[PowerShell]` classes and stuff.

```
[Michael Willis @ xainey] ~    [https://xainey.github.io/2016/powershell-classes-and-concepts/]
[+] Developer, Student, and Automation Enthusiast.
[+] Certified in Puncraft and the Wibbly Wobbly.
[+] Believes in DevOps, Alligators, and Aristotle.
```

This dude goes into extreme depth, making comparisons between `[PowerShell]`, and JavaScript/Java, C#/C++, and etc. I highly suggest reading that article, in order to understand why I've capitalized on the material.

There are plenty of other sources to consider, however, that is extremely useful and it gets right to the point.

To continue, follow the resource link, or the updated function I'll post at some point in this document.
Or, literally copy+paste all of the above classes into the function wrapper above.

```
PS Prompt:\> $Ctrl = Get-ViperBomb
[00:00:00.0600031] (State: 0/Status: Loading [~] [FightingEntropy(π)][System Control Extension Utility])
[00:00:01.2361728] (State: 0/Status: Loading [~] AppX)
[00:00:01.2898663] (State: 0/Status: Gathering [~] System/Details)
[00:00:05.2876800] (State: 0/Status: Gathering [~] OS/Hotfix Details)
[00:00:05.9527697] (State: 0/Status: Gathering [~] Default/Config)
[00:00:06.1007702] (State: 0/Status: Gathering [~] System Control)
[00:00:14.5692946] (State: 0/Status: Gathering [~] Xaml Interface)
PS Prompt:\> $Ctrl.Console.Finalize()
PS Prompt:\>
```

The above command `$Ctrl.Console.Finalize()` is not necessary, but it will stop the console timer.

```
PS Prompt:\> $Ctrl

Console   : 00:00:20.8280029
Module    : Get-FEModule.Main
System    :
Xaml      : Get-ViperBomb.XamlWindow
Caption   : Microsoft Windows 10.0.19045
Platform  : Win32NT
PSVersion : 5.1.19041.1682
Type      : Win32_Client
HotFix    : Get-ViperBomb.HotFixList
Config    : Get-ViperBomb.ViperBombConfig
Service   :
Control   : Get-ViperBomb.SystemControl

PS Prompt:\>
```

So, here is the main portion of the object. Some of the information is showing a little differently than I would like...? However, everything that says `Get-FEModule` or `Get-ViperBomb` are effectively function wrappers that have underlying base classes as well as additional subproperties/methods, typically anyway.

Let's examine the `$Ctrl.Console` property, as it shows information related to the stuff printed out to the console up above. The `$Ctrl.Console` property is actually the `[ViperBombStatusBank]` object, and it is actually a smaller class factory within a larger class factory. That is effectively the same exact way that C# works.

```
PS Prompt:\> $Ctrl.Console

Start  : 12/2/2022 1:29:21 PM
End    : 12/2/2022 1:29:42 PM
Span   : 00:00:20.8280029
Status : [00:00:20.8280029] (State: 100/Status: Complete [+] (12/2/2022 1:29:42 PM), Total: (00:00:20.8280029))
Output : {[00:00:00.0199996] (State: 0/Status: Running [~] (12/2/2022 1:29:21 PM))... }

PS Prompt:\>
```

I am actually truncating some of the information returned from the function, so that it fits within the box.
The `$Ctrl.Console.Start` box is when the function was started... and `$Ctrl.Console.End` is when I used the `$Ctrl.Console.Finalize()` method.

The `$Ctrl.Console.Span` property returns a `[Timespan]` `[String]`. (That means it can be deserialized...)
The `$Ctrl.Console.Status` property returns a `[ViperBombStatus]` object.
The `$Ctrl.Console.Output` property reutrns an array of `[ViperBombStatus]` objects.

```
PS Prompt:\> $Ctrl.Console.Start

Name  Time                 Set
----  ----                 ---
Start 12/2/2022 1:29:21 PM   1

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.Console.End
```

```
Name Time                Set
---- ----                ---
End  12/2/2022 1:29:42 PM   1

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.Console.Span
00:00:20.8280029

PS Prompt:\> [Timespan]$Ctrl.Console.Span

Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 20
Milliseconds      : 828
Ticks             : 208280029
TotalDays         : 0.00024106484837963
TotalHours        : 0.00578555636111111
TotalMinutes      : 0.347133381666667
TotalSeconds      : 20.8280029
TotalMilliseconds : 20828.0029

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.Console.Status

Index Elapsed          State Status
----- -------          ----- ------
    8 00:00:20.8280029   100 Complete [+] (12/2/2022 1:29:42 PM), Total: (00:00:20.8280029)

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.Console.Output

Index Elapsed          State Status
----- -------          ----- ------
    0 00:00:00.0199996     0 Running [~] (12/2/2022 1:29:21 PM)
    1 00:00:00.0600031     0 Loading [~] [FightingEntropy(π)][System Control Extension Utility]
    2 00:00:01.2361728     0 Loading [~] AppX
    3 00:00:01.2898663     0 Gathering [~] System/Details
    4 00:00:05.2876800     0 Gathering [~] OS/Hotfix Details
    5 00:00:05.9527697     0 Gathering [~] Default/Config
    6 00:00:06.1007702     0 Gathering [~] System Control
    7 00:00:14.5692946     0 Gathering [~] Xaml Interface
    8 00:00:20.8280029   100 Complete [+] (12/2/2022 1:29:42 PM), Total: (00:00:20.8280029)

PS Prompt:\>
```

Those are all of the subproperties of the `$Ctrl.Console` property, and what I can say right here and now, is that there is a NETWORK of properties and methods that this little `$Ctrl` variable has.

In particular, the properties that have subproperties and methods...
`$Ctrl.Module`, `$Ctrl.System`, `$Ctrl.Xaml`, `$Ctrl.HotFix`, `$Ctrl.Config`, `$Ctrl.Service`, `$Ctrl.Control`.

I mean, look at it this way...
If you're looking to kick ass and chew bubble gum...?

```
----------------------------
| This is the way to do it |
----------------------------
```

AND... you'll REALLY be able to show Duke Nukem how to get some stuff done, even if you're all outta gum.

If I go into depth regarding all of these properties...?
This is going to be an incredibly long document.

As for the `$Ctrl.Module` proeprty, some elements from it are necessary.

```
PS Prompt:\> $Ctrl.Module

Source      : https://www.github.com/mcc85s/FightingEntropy
Name        : [FightingEntropy(π)]
Description : Beginning the fight against ID theft and cybercrime
```

```
Author      : Michael C. Cook Sr.
Company     : Secure Digits Plus LLC
Copyright   : (c) 2022 (mcc85s/mcc85sx/sdp). All rights reserved.
Guid        : 0b36cfa4-dfad-4863-9171-f8afe65769cf
Date        : 11/7/2022 4:01:21 PM
Version     : 2022.11.0
OS          : <FightingEntropy.Module.OS>
Root        : <FightingEntropy.Module.Root>
Manifest    : <FightingEntropy.Module.Manifest>
Registry    : <FightingEntropy.Module.RegistryKey>

PS Prompt:\>
```

I've covered some of those subproperties in another document...

Date	Url
10/31/22	[https://github.com/mcc85s/FightingEntropy/blob/main/Docs/2022_1031-(FightingEntropy).pdf]

I've expanded upon those above properties in that document listed above.
The module itself has access to ALL of the files, functions, and registry settings that allow certain components to be more readily available upon demand. There is SOME merit to the phrase:

```
-----------------------------------------------------------------------------------------
| Smart Guy : Better to HAVE a tool, and NOT NEED it, than to NEED it, and NOT HAVE it. |
-----------------------------------------------------------------------------------------
```

The idea behind the module AND this utility, is to have subcomponents that are only pulled as needed.
Think of it like this.
Resources have a level of SCARCITY, so like, a guy who just so happens to own a tractor trailer full of stuff...?
He's not gonna drive his truck to a corner store that's like, 100 feet away. He'll say to himself:

```
-------------------------------------------------------------------------------------------------------
| Truck driver : It would be pretty ridiculous, for me to drive my truck 100 feet down the street... |
-------------------------------------------------------------------------------------------------------
```

However, when additional features or information are necessary, then they'll be ready to pull into memory.
The end result is that the module is rather lightweight, easy to (understand/use), and powerful.

I won't cover the `$Ctrl.Module` property, so if you really want to understand what's happening there...?
Scope out the above PDF link.

I WILL cover the other properties to some extent. `$Ctrl.System` is actually another function in the module, one that I wrote specifically for the `[Event Log Utility]` that I've talked about in other documents.

```
PS Prompt:\> $Ctrl.System

Snapshot        : L420-X64
BiosInformation : LENOVO | Ver 1.00PARTTBL
OperatingSystem : Microsoft Windows 10 Pro 10.0.19045-19045
ComputerSystem  : LENOVO | 7829AU6
Processor       : Processor(s)[1]
Disk            : Disk(s)[1]
Network         : Network(s)[1]
LogProviders    : Log Providers[492]

PS Prompt:\>
```

These properties are all things that are able to be seen in the (GUI/graphical user interface).
They're all in `[String]` format, but are able to be expanded. The `$Ctrl.System.Snapshot` property isn't used, here.

Effectively, I'm utilizing a way to create custom functions in the module `[FightingEntropy(π)]`, and then immediately utilizing (GUI/program) elements that, in many other instances, have to be installed as an executable. However, all of this is able to be accessed via the command line, and it is pretty self-contained, and these elements can be easily accessed from additional functions, and stuff like that.

So, lets cover the various properties seen above, as they are critical to the `[System]` tab in the GUI.
In the [Square Brackets], is the INDEX for the screenshot earlier in the document. I will also cover the additional tabs from the GUI screenshots up above.

```
[01]: [System] tab, OS/[PowerShell] Info + Bios Info subtab
```

```
PS Prompt:\> $Ctrl.Module.OS

Caption    : Microsoft Windows 10.0.19045
```

```
Platform  : Win32NT
PSVersion : 5.1.19041.1682
Type      : Win32_Client
Output    : {(51) <FightingEntropy.Module.OSPropertySet[Environment]>,
             (65) <FightingEntropy.Module.OSPropertySet[Variable]>,
             (10) <FightingEntropy.Module.OSPropertySet[Host]>,
             (10) <FightingEntropy.Module.OSPropertySet[PowerShell]>}

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.System.BiosInformation

Name            : Ver 1.00PARTTBL
Manufacturer    : LENOVO
SerialNumber    : LR0FXTA
Version         : LENOVO - 1230
ReleaseDate     : 09/15/2013 20:00:00
SmBiosPresent   : True
SmBiosVersion   : 8GET46WW (1.23 )
SmBiosMajor     : 2
SmBiosMinor     : 6
SystemBiosMajor : 1
SystemBiosMinor : 35

PS Prompt:\>
```

```
[02]: [System] tab, Operating System subtab
```

```
PS Prompt:\> $Ctrl.System.OperatingSystem

Caption  : Microsoft Windows 10 Pro
Version  : 10.0.19045
Build    : 19045
Serial   : 00330-50000-00000-AAOEM
Language : 1033
Product  : 256
Type     : 18

PS Prompt:\>
```

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

Source   Description     HotFixID  InstalledBy         InstalledOn
------   -----------     --------  -----------         -----------
L420-X64 Update          KB2693643                     09/16/2021
L420-X64 Update          KB5020613 NT AUTHORITY\SYSTEM 11/17/2022
L420-X64 Update          KB4562830 NT AUTHORITY\SYSTEM 04/02/2021
L420-X64 Security Update KB4570334                     11/18/2020
L420-X64 Update          KB4577586 NT AUTHORITY\SYSTEM 04/02/2021
L420-X64 Security Update KB4580325                     11/19/2020
L420-X64 Security Update KB4586864                     11/19/2020
L420-X64 Update          KB4589212 NT AUTHORITY\SYSTEM 04/02/2021
L420-X64 Update          KB5000736 NT AUTHORITY\SYSTEM 05/30/2021
L420-X64 Update          KB5003791 NT AUTHORITY\SYSTEM 12/17/2021
L420-X64 Security Update KB5012170 NT AUTHORITY\SYSTEM 09/10/2022
L420-X64 Update          KB5015684 NT AUTHORITY\SYSTEM 10/25/2022
L420-X64 Security Update KB5019959 NT AUTHORITY\SYSTEM 11/17/2022
L420-X64 Update          KB5006753 NT AUTHORITY\SYSTEM 11/11/2021
L420-X64 Update          KB5007273 NT AUTHORITY\SYSTEM 12/17/2021
L420-X64 Update          KB5016705 NT AUTHORITY\SYSTEM 09/06/2022
L420-X64 Update          KB5018506 NT AUTHORITY\SYSTEM 11/17/2022
L420-X64 Security Update KB5005699 NT AUTHORITY\SYSTEM 09/14/2021
L420-X64 Update          KB5007273 NT AUTHORITY\SYSTEM 12/17/2021
L420-X64 Update          KB5016705 NT AUTHORITY\SYSTEM 09/06/2022

PS Prompt:\>
```

```
[03]: [System] tab, Computer System subtab
```

```
PS Prompt:\> $Ctrl.System.ComputerSystem

Manufacturer : LENOVO
Model        : 7829AU6
```

```
Product      : 7829AU6
Serial       : 1ZKG92222GY
Memory       : 3.84 GB
Architecture : x64
UUID         : D4059A81-4AEE-11CB-8D7D-E9D79C51C524
Chassis      : Laptop
BiosUefi     : BIOS
AssetTag     : No Asset Information

PS Prompt:\>
```

[04]: [System] tab, Processor(s) subtab

```
PS Prompt:\> $Ctrl.System.Processor

Name         Count Output
----         ----- ------
Processor(s)     1 {Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz}

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.System.Processor.Output

Manufacturer : Intel
Name         : Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz
Caption      : Intel64 Family 6 Model 42 Stepping 7
Cores        : 2
Used         : 2
Logical      : 4
Threads      : 4
ProcessorId  : BFEBFBFF000206A7
DeviceId     : CPU0
Speed        : 2501

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.System.Processor.Output | Format-Table

Manufacturer Name                                  Caption                             Cores Used Logical
------------ ----                                  -------                             ----- ---- -------
Intel        Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz Intel64 Family 6 Model 42 Stepping 7     2    2       4

PS Prompt:\>
```

[05]: [System] tab, Disks subtab

```
PS Prompt:\> $Ctrl.System.Disk

Name    Count Output
----    ----- ------
Disk(s)     1 {Crucial_CT1050MX300SSD4(0)}

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.System.Disk.Output

Index             : 0
Disk              : \\.\PHYSICALDRIVE0
Model             : Crucial_CT1050MX300SSD4
Serial            : 170715DE96DD
PartitionStyle    : 1
ProvisioningType  : 2
OperationalStatus : 53264
HealthStatus      : 0
BusType           : 11
UniqueId          : 500A071515DE96DD
Location          : Integrated : Bus 0 : Device 31 : Function 2 : Adapter 0 : Port 0
Partition         : (3) [Disk #0, Partition #0/50.00 MB],
                        [Disk #0, Partition #1/977.55 GB],
                        [Disk #0, Partition #2/499.00 MB]
Volume            : (1) [C:977.55 GB]

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.System.Disk.Output | Format-Table (Edited)

Index Disk               Model                  Serial       P.Style OpStatus HStatus B.Type UniqueId
----- ----               -----                  ------       ------- -------- ------- ------ --------
    0 \\.\PHYSICALDRIVE0 Crucial_CT1050MX300SSD4 170715DE96DD 1       53264    0       11     00A071515DE96DD

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.System.Disk.Output[0].Partition[0].Output | Format-Table

Type                   Name                  Size      Boot Primary Disk Partition
----                   ----                  ----      ---- ------- ---- ---------
Installable File System Disk #0, Partition #0 50.00 MB     1       1    0         0
Installable File System Disk #0, Partition #1 977.55 GB    0       1    0         1
Unknown                Disk #0, Partition #2 499.00 MB    0       1    0         2

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.System.Disk.Output[0].Volume[0].Output | Format-Table (Edited)

DriveID Description      Filesystem Partition                        Freespace Used      Size
------- -----------      ---------- ---------                        --------- ----      ----
C:      Local Fixed Disk NTFS       [Disk #0, Partition #1/977.55 GB] 526.57 GB 450.98 GB 977.55 GB

PS Prompt:\>
```

```
[06]: [System] tab, Network subtab
```

```
PS Prompt:\> $Ctrl.System.Network.Output

Name       : Remote NDIS based Internet Sharing Device
Index      : 17
IPAddress  : 192.168.42.146
SubnetMask : 255.255.255.0
Gateway    : 192.168.42.129
DnsServer  : 192.168.42.129
DhcpServer : 192.168.42.129
MacAddress : BE:D4:54:4A:83:21

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.System.Network.Output | Format-Table (Edited)

Name      Index IPAddress      SubnetMask    Gateway        DnsServer      DhcpServer     MacAddress
----      ----- ---------      ----------    -------        ---------      ----------     ----------
RNDIS ISD    17 192.168.42.146 255.255.255.0 192.168.42.129 192.168.42.129 192.168.42.129 BE:D4:54:4A:83:21

PS Prompt:\>
```

```
[07]: [Service] tab, Configuration subtab
    ------------------------------------------------------------------------------------------
    | Note: In order to have access to the stuff seen in the GUI, you have to effectively run the |
    | method/command that clicking the “Get” button accesses, by performing the following...      |
    ------------------------------------------------------------------------------------------
```

```
PS Prompt:\> $Ctrl.Service = $Ctrl.GetServices()
[00:00:20.8280029] (State: 0/Status: Gathering [~] System/Services)
PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.Service

Slot Sub                                Config                        Output
---- ---                                ------                        ------
   0 Get-ViperBomb.ServiceSubcontroller {AJRouter, ALG, AppHostSvc...} {AarSvc_8ab7c, AJRouter, ALG...}

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.Service.Sub.StartMode.Output

Index Type        Description
----- ----        -----------
    0 Skip        The service is skipped
    1 Disabled    The service is totally disabled
    2 Manual      The service requires a manual start
```

```
    3 Auto        The service automatically starts
    4 AutoDelayed The service automatically starts, but is delayed

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.Service.Sub.State.Output

Index Type    Description
----- ----    -----------
    0 Running The service is currently running
    1 Stopped The service is NOT currently running

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.Service.Config[0..2]

Name       Profile
----       -------
AJRouter   {2, 2, 2, 2...}
ALG        {2, 2, 2, 2...}
AppHostSvc {3, 0, 3, 0...}

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.Service.Output[0]

Index            : 0
Name             : AarSvc_8ab7c
Scope            : 0
Profile          : {2, 2, 2, 2...}
StartMode        : Manual
Target           : Manual
State            : Running
DelayedAutoStart : 0
Status           : OK
DisplayName      : Agent Activation Runtime_8ab7c
PathName         : C:\Windows\system32\svchost.exe -k AarSvcGroup -p
Description      : Runtime for activating conversational agent applications

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.Service.Output[0..2] | Format-Table

Index Name         Scope Profile         StartMode Target State   DelayedAutoStart Status DisplayName
----- ----         ----- -------         --------- ------ -----   ---------------- ------ -----------
    0 AarSvc_8ab7c     0 {2, 2, 2, 2...} Manual    Manual Running                0 OK     Agent Activation...
    1 AJRouter         1 {2, 2, 2, 2...} Manual    Manual Stopped                0 OK     AllJoyn Router S...
    2 ALG              1 {2, 2, 2, 2...} Manual    Manual Stopped                0 OK     Application Laye...

PS Prompt:\>
```

```
[08]: [Service] tab, Preferences subtab

None of this is wired up fpr functionality right now.
However... some of the properties below ARE wired up.
```

```
PS Prompt:\> $Ctrl.Config

BypassBuild      : 0
BypassEdition    : 0
BypassLaptop     : 0
DisplayActive    : 1
DisplayInactive  : 1
DisplaySkipped   : 1
MiscSimulate     : 0
MiscXbox         : 1
MiscChange       : 0
MiscStopDisabled : 0
DevErrors        : 0
DevLog           : 0
DevConsole       : 0
DevReport        : 0
LogServiceLabel  : Service.log
LogScriptLabel   : Script.log
```

```
RegLabel         : Backup.reg
CsvLabel         : Backup.csv
Filter           : {XblAuthManager, XblGameSave, XboxNetApiSvc, XboxGipSvc...}
Profile          : {HomeMax, HomeMin, ProMax, ProMin...}

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.Config.Filter

Type   Name
----   ----
Xbox   XblAuthManager
Xbox   XblGameSave
Xbox   XboxNetApiSvc
Xbox   XboxGipSvc
Xbox   xbgm
NetTCP NetMsmqActivator
NetTCP NetPipeActivator
NetTCP NetTcpActivator
Skip   BcastDVRUserService_8ab7c
Skip   DevicePickerUserSvc_8ab7c
Skip   DevicesFlowUserSvc_8ab7c
Skip   PimIndexMaintenanceSvc_8ab7c
Skip   PrintWorkflowUserSvc_8ab7c
Skip   UnistoreSvc_8ab7c
Skip   UserDataSvc_8ab7c
Skip   WpnUserService_8ab7c
Skip   AppXSVC
Skip   BrokerInfrastructure
Skip   ClipSVC
Skip   CoreMessagingRegistrar
Skip   DcomLaunch
Skip   EntAppSvc
Skip   gpsvc
Skip   LSM
Skip   MpsSvc
Skip   msiserver
Skip   NgcCtnrSvc
Skip   NgcSvc
Skip   RpcEptMapper
Skip   RpcSs
Skip   Schedule
Skip   SecurityHealthService
Skip   sppsvc
Skip   StateRepository
Skip   SystemEventsBroker
Skip   tiledatamodelsvc
Skip   WdNisSvc
Skip   WinDefend

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.Config.Profile

Index Type              Description
----- ----              -----------
    0 HomeMax           Windows (10|11) Home, Default/Maximum
    1 HomeMin           Windows (10|11) Home, Default/Minimum
    2 ProMax            Windows (10|11) Pro, Default/Maximum
    3 ProMin            Windows (10|11) Pro, Default/Minimum
    4 DesktopSafeMax    Desktop (General), Safe Maximum
    5 DesktopSafeMin    Desktop (General), Safe Minimum
    6 DesktopTweakedMax Desktop (General), Tweaked Maximum
    7 DesktopTweakedMin Desktop (General), Tweaked Minimum
    8 LaptopSafeMax     Laptop (General), Safe Maximum
    9 LaptopSafeMin     Laptop (General), Safe Minimum

PS Prompt:\>
```

```
[09]: [Service] tab, About subtab
All of the information in this particular tab is statically written into the chunk of XAML for the time being.
It WILL be reimplemented, via these classes at SOME point.
```

```
[10]: [Control] tab, Preferences subtab
None of the options in this particular tab are wired up.
```

```
[11]: [Control] tab, Settings subtab

Note: This particular tab has some wicked useful flexibility where the ComboBox allows the selection of certain
properties or sources.

So, if you want to make changes to ALL of the options...?
Then, the option for ALL subitems is there.

If you aren't sure what particular section or name is related to a particular key phrase...?
You can filter all of the options by their description.

This effectively works a lot like the native <Control Panel>.
```

```
PS Prompt:\> $Ctrl.Control | Format-List

Output  : {Telemetry, WifiSense, SmartScreen, LocationTracking...}
Feature : {AppServerClient, Client-DeviceLockdown, Client-EmbeddedBootExp, Client-EmbeddedLogon...}
AppX    : {Microsoft.549981C3F5F10_4.2204.13303.0_neutral_~_8wekyb3d8bbwe,
           Microsoft.BingWeather_4.53.43112.0_neutral_~_8wekyb3d8bbwe,
           Microsoft.DesktopAppInstaller_2022.927.3.0_neutral_~_8wekyb3d8bbwe,
           Microsoft.GetHelp_10.2208.2551.0_neutral_~_8wekyb3d8bbwe...}

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.Control.Output[0..2] | Format-Table (Edited)

Source  Name        DisplayName Value Description         Options                 Output
------  ----        ----------- ----- -----------         -------                 ------
Privacy Telemetry   Telemetry       1 Various location... {Skip, Enable*, Disable} {AllowTelemetry, All...}
Privacy WifiSense   Wi-Fi Sense     1 Lets devices mor... {Skip, Enable*, Disable} {Value, Value, AutoC...}
Privacy SmartScreen SmartScreen     1 Cloud-based anti... {Skip, Enable*, Disable} {SmartScreenEnabled,...}

PS Prompt:\>
```

```
[12]: [Control] tab, Features subtab

This touches on some aspects of the server end stuff in the function Get-FEDCPromo, and Install-IISServer

While those functions create their own internal properties related to Windows features, there's the possibility
that I may EXTEND this particular feature, so that it is its' own external function.

Who knows.
```

```
PS Prompt:\> $Ctrl.Control.Feature

Index FeatureName                            State
----- -----------                            -----
    0 AppServerClient                        Disabled
    1 Client-DeviceLockdown                  Disabled
    2 Client-EmbeddedBootExp                 Disabled
    3 Client-EmbeddedLogon                   Disabled
    4 Client-EmbeddedShellLauncher           Disabled
    5 ClientForNFS-Infrastructure            Disabled
    6 Client-KeyboardFilter                  Disabled
    7 Client-ProjFS                          Disabled
    8 Client-UnifiedWriteFilter              Disabled
    9 Containers                             Disabled
   10 Containers-DisposableClientVM          Disabled
   11 DataCenterBridging                     Disabled
   12 DirectoryServices-ADAM-Client          Disabled
   13 DirectPlay                             Disabled
   14 HostGuardian                           Disabled
   15 HypervisorPlatform                     Disabled
   16 IIS-ApplicationDevelopment             Disabled
   17 IIS-ApplicationInit                    Disabled
   18 IIS-ASP                                Disabled
   19 IIS-ASPNET                             Disabled
```

```
 20 IIS-ASPNET45                                Disabled
 21 IIS-BasicAuthentication                     Disabled
 22 IIS-CertProvider                            Disabled
 23 IIS-CGI                                     Disabled
 24 IIS-ClientCertificateMappingAuthentication  Disabled
 25 IIS-CommonHttpFeatures                      Disabled
 26 IIS-CustomLogging                           Disabled
 27 IIS-DefaultDocument                         Disabled
 28 IIS-DigestAuthentication                    Disabled
 29 IIS-DirectoryBrowsing                       Disabled
 30 IIS-FTPExtensibility                        Disabled
 31 IIS-FTPServer                               Disabled
 32 IIS-FTPSvc                                  Disabled
 33 IIS-HealthAndDiagnostics                    Disabled
 34 IIS-HostableWebCore                         Disabled
 35 IIS-HttpCompressionDynamic                  Disabled
 36 IIS-HttpCompressionStatic                   Disabled
 37 IIS-HttpErrors                              Disabled
 38 IIS-HttpLogging                             Disabled
 39 IIS-HttpRedirect                            Disabled
 40 IIS-HttpTracing                             Disabled
 41 IIS-IIS6ManagementCompatibility             Disabled
 42 IIS-IISCertificateMappingAuthentication     Disabled
 43 IIS-IPSecurity                              Disabled
 44 IIS-ISAPIExtensions                         Disabled
 45 IIS-ISAPIFilter                             Disabled
 46 IIS-LegacyScripts                           Disabled
 47 IIS-LegacySnapIn                            Disabled
 48 IIS-LoggingLibraries                        Disabled
 49 IIS-ManagementConsole                       Disabled
 50 IIS-ManagementScriptingTools                Disabled
 51 IIS-ManagementService                       Disabled
 52 IIS-Metabase                                Disabled
 53 IIS-NetFxExtensibility                      Disabled
 54 IIS-NetFxExtensibility45                    Disabled
 55 IIS-ODBCLogging                             Disabled
 56 IIS-Performance                             Disabled
 57 IIS-RequestFiltering                        Disabled
 58 IIS-RequestMonitor                          Disabled
 59 IIS-Security                                Disabled
 60 IIS-ServerSideIncludes                      Disabled
 61 IIS-StaticContent                           Disabled
 62 IIS-URLAuthorization                        Disabled
 63 IIS-WebDAV                                  Disabled
 64 IIS-WebServer                               Disabled
 65 IIS-WebServerManagementTools                Disabled
 66 IIS-WebServerRole                           Disabled
 67 IIS-WebSockets                              Disabled
 68 IIS-WindowsAuthentication                   Disabled
 69 IIS-WMICompatibility                        Disabled
 70 Internet-Explorer-Optional-amd64            Enabled
 71 LegacyComponents                            Disabled
 72 MediaPlayback                               Enabled
 73 Microsoft-Hyper-V                           Enabled
 74 Microsoft-Hyper-V-All                       Enabled
 75 Microsoft-Hyper-V-Hypervisor                Enabled
 76 Microsoft-Hyper-V-Management-Clients        Enabled
 77 Microsoft-Hyper-V-Management-PowerShell     Enabled
 78 Microsoft-Hyper-V-Services                  Enabled
 79 Microsoft-Hyper-V-Tools-All                 Enabled
 80 MicrosoftWindowsPowerShellV2                Enabled
 81 MicrosoftWindowsPowerShellV2Root            Enabled
 82 Microsoft-Windows-Subsystem-Linux           Enabled
 83 MSMQ-ADIntegration                          Disabled
 84 MSMQ-Container                              Disabled
 85 MSMQ-DCOMProxy                              Disabled
 86 MSMQ-HTTP                                   Disabled
 87 MSMQ-Multicast                              Disabled
 88 MSMQ-Server                                 Disabled
 89 MSMQ-Triggers                               Disabled
 90 MSRDC-Infrastructure                        Enabled
 91 MultiPoint-Connector                        Disabled
```

```
  92 MultiPoint-Connector-Services               Disabled
  93 MultiPoint-Tools                            Disabled
  94 NetFx3                                      DisabledWithPayloadRemoved
  95 NetFx4-AdvSrvs                              Enabled
  96 NetFx4Extended-ASPNET45                     Enabled
  97 NFS-Administration                          Disabled
  98 Printing-Foundation-Features                Enabled
  99 Printing-Foundation-InternetPrinting-Client Enabled
 100 Printing-Foundation-LPDPrintService         Disabled
 101 Printing-Foundation-LPRPortMonitor          Disabled
 102 Printing-PrintToPDFServices-Features        Enabled
 103 Printing-XPSServices-Features               Enabled
 104 SearchEngine-Client-Package                 Enabled
 105 ServicesForNFS-ClientOnly                   Disabled
 106 SimpleTCP                                   Disabled
 107 SMB1Protocol                                Disabled
 108 SMB1Protocol-Client                         Disabled
 109 SMB1Protocol-Deprecation                    Disabled
 110 SMB1Protocol-Server                         Disabled
 111 SmbDirect                                   Enabled
 112 TelnetClient                                Disabled
 113 TFTP                                        Disabled
 114 TIFFIFilter                                 Disabled
 115 VirtualMachinePlatform                      Disabled
 116 WAS-ConfigurationAPI                        Disabled
 117 WAS-NetFxEnvironment                        Disabled
 118 WAS-ProcessModel                            Disabled
 119 WAS-WindowsActivationService                Disabled
 120 WCF-HTTP-Activation                         Disabled
 121 WCF-HTTP-Activation45                       Disabled
 122 WCF-MSMQ-Activation45                       Disabled
 123 WCF-NonHTTP-Activation                      Disabled
 124 WCF-Pipe-Activation45                       Disabled
 125 WCF-Services45                              Enabled
 126 WCF-TCP-Activation45                        Disabled
 127 WCF-TCP-PortSharing45                       Enabled
 128 Windows-Defender-ApplicationGuard           Disabled
 129 Windows-Defender-Default-Definitions        Disabled
 130 Windows-Identity-Foundation                 Disabled
 131 WindowsMediaPlayer                          Enabled
 132 WorkFolders-Client                          Enabled

PS Prompt:\>
```

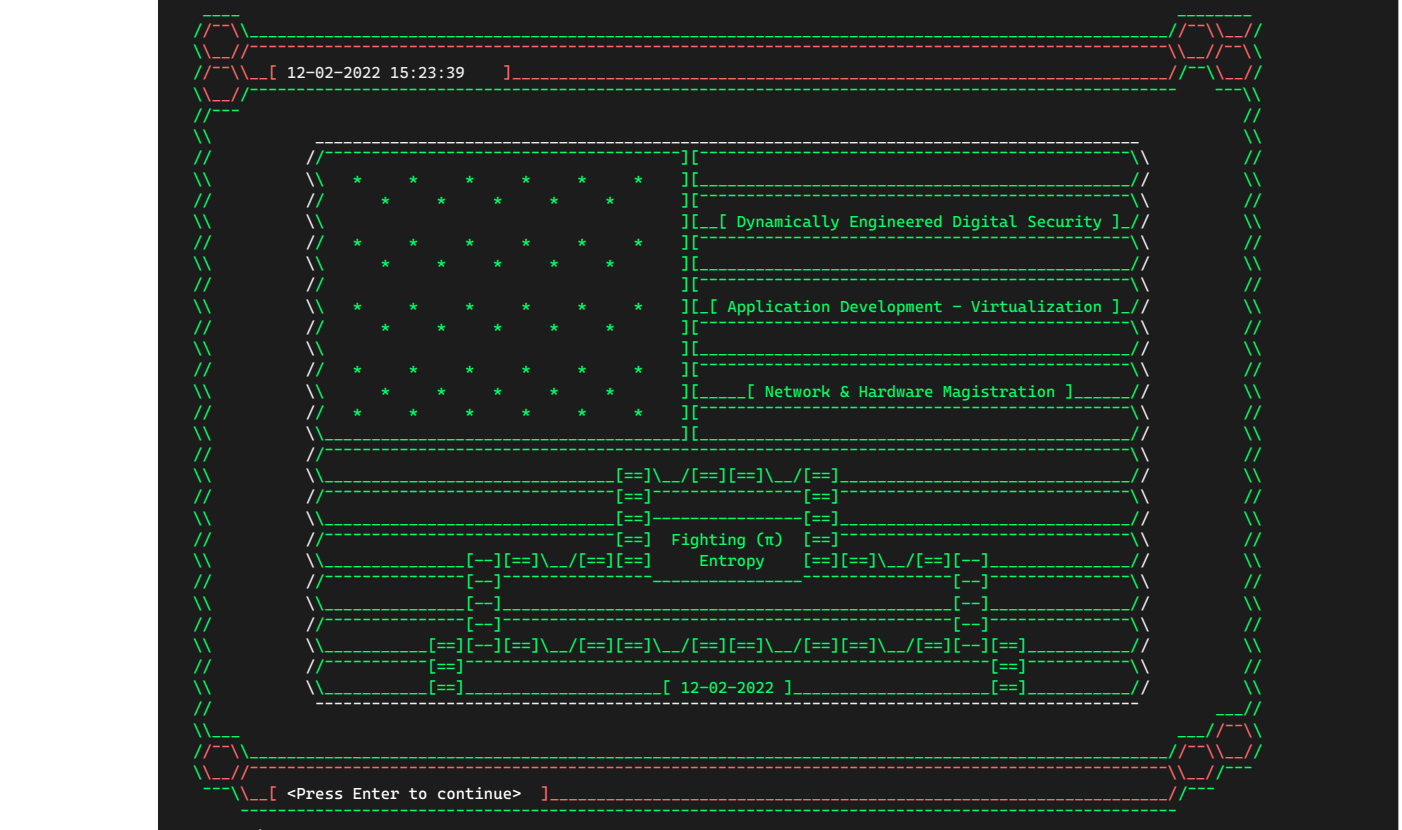
```
[13]: [Control] tab, AppX subtab
PS Prompt:\> $Ctrl.Control.AppX[0] (Edited)

Index            : 0
Profile          : 0
Version          : 4.2204.13303.0
PackageName      : Microsoft.549981C3F5F10_4.2204.13303.0_neutral_~_8wekyb3d8bbwe
DisplayName      : Microsoft.549981C3F5F10
PublisherID      : 8wekyb3d8bbwe
MajorVersion     : 4
MinorVersion     : 2204
Build            : 13303
Revision         : 0
Architecture     : 11
ResourceID       : ~
InstallLocation  : C:\Program Files\WindowsApps\Microsoft.549981C3F5F10_4.2204.13303.0_neutral_~_8wekyb3d8b...
Regions          :
Path             :
Online           : 1
WinPath          :
SysDrivePath     :
RestartNeeded    : 0
LogPath          : C:\Windows\Logs\DISM\dism.log
ScratchDirectory :
LogLevel         : WarningsInfo
Slot             : -1
```

```
PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.Control.AppX[0..2] | Format-Table (Edited)

Index Pro. Version        PackageName       DisplayName                  PublisherID   Major Minor Build Rev
----- ---- -------        -----------       -----------                  -----------   ----- ----- ----- ---
    0    0 4.2204.13303.0 Microsoft.5499... Microsoft.549981C3F5F10      8wekyb3d8bbwe     4  2204 13303   0
    1    0 4.53.43112.0   Microsoft.Bing... Microsoft.BingWeather        8wekyb3d8bbwe     4    53 43112   0
    2    0 2022.927.3.0   Microsoft.Desk... Microsoft.DesktopAppInstaller 8wekyb3d8bbwe  2022   927     3   0

PS Prompt:\>
```

```
[14]: [FightingEntropy(π)] Flag

This is a custom feature of the module, just sort of adding some extra flair to the presentation.

Normally, this command would stylize the font below with COLOR in the CONSOLE.

In this instance, I'm manually changing the colors, and, further to that point, I would have to use tables in
order to stylize this with the proper background colors.

MAYBE in a future demonstration or lesson plan, I will actually do that. Not today, I'm afraid.
```

```
PS Prompt:\> Write-Theme -Flag
```




```
PS Prompt:\>
```

There is a property that I haven't covered, and it dictates all of the interaction or (input/output) with the Xaml object, $Ctrl.Xaml.

```
PS Prompt:\> $Ctrl.Xaml

Names     : {Window, Border, OS, BiosInformation...}
Types     : {Window, OS, BiosInformation, BiosInformationExtension...}
Node      : System.Xml.XmlNodeReader
IO        : System.Windows.Window
Exception :

PS Prompt:\>
```

So, what is seen here, is basically:

```
-------------------------------------------------------------------------------------------
| 1) the names of the controls,                                                           |
| 2) the (types/extended properties) for each control that [PowerShell] interacts with,   |
| 3) the [System.Xml.XmlNodeReader] object, and                                           |
| 4) the [System.Windows.Window] object                                                   |
-------------------------------------------------------------------------------------------
```

SOME of this stuff can be seen in this video that has an associated lesson plan...

[https://www.youtube.com/watch?v=NK4NuQrraCI]

[https://github.com/mcc85sx/FightingEntropy/blob/master/Documentation/2021_0128-A_Deep_Dive.pdf]

However, I've expanded upon what this is able to do.
When [Bill Gates], cofounder and former CEO of the [Microsoft Corporation] pioneered an industry that capitalized on what a little Altair 8800 could do when combined with well-written code...?

He was one of the founding fathers for a thing that became a term, [Operating System]
You wouldn't call something that doesn't WORK, an [Operating System], cause then itd be a [Non-operating System].
Anyway, he developed an array of [Operating System[]], each with their own unique NAME, VERSION, and CAPABILITIES.

The last [Operating System] that this guy released was back on October 25th, 2001.
A day that will live on, in expanded productivity.
That's cause... that's what Windows XP stands for, eXpanded Productivity.

Anyway, that's what the above thing does, that I'm about to go over.
It EXPANDS upon the original thing I covered in the above video... for expanded productivity.

```
PS Prompt:\> $Ctrl.Xaml.Names
Window
Border
OS
BiosInformation
BiosInformationExtension
OperatingSystem
HotFix
ComputerSystem
ComputerSystemExtension
Processor
ProcessorExtension
Disk
DiskExtension
DiskPartition
DiskVolume
Network
NetworkExtension
ServiceSlot
ServiceDisplayly
ServicePropertyApply
ServiceFiltercls
Service
ServiceExtension
ServiceGet
ServiceSet
ServiceBypassBuild
ServiceBypassEdition
ServiceBypassLaptop
ServiceDisplayActive
ServiceDisplayInactive
ServiceDisplaySkipped
ServiceMiscSimulate
ServiceMiscXbox
ServiceMiscChange
ServiceMiscStopDisabled
ServiceLogServiceSwitch
ServiceLogServiceFile
ServiceLogServiceBrowse
ServiceLogScriptSwitch
ServiceLogScriptFile
ServiceLogScriptBrowse
ServiceRegSwitch
```

```
ServiceRegFile
ServiceRegBrowse
ServiceCsvSwitch
ServiceCsvFile
ServiceCsvBrowse
ServiceDevErrors
ServiceDevLog
ServiceDevConsole
ServiceDevReport
ControlGlobalRestorePoint
ControlGlobalRestart
ControlGlobalShowSkipped
ControlGlobalVersionCheck
ControlGlobalInternetCheck
ControlBackupSave
ControlBackupLoad
ControlBackupWinDefault
ControlBackupResetDefault
ControlSlot
ControlProperty
ControlFilter
ControlOutput
ControlOutputExtension
ControlOutputApply
ControlOutputDontApply
ControlFeatureProperty
ControlFeatureFilter
ControlFeature
ControlFeatureExtension
ControlFeatureApply
ControlFeatureDontApply
ControlAppXProperty
ControlAppXFilter
ControlAppX
ControlAppXExtension
ControlAppXApply
ControlAppXDontApply
PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.Xaml.Types
Index Name                        Type     Control
----- ----                        ----     -------
    0 Window                      Window   System.Windows.Window
    1 OS                          DataGrid System.Windows.Controls.DataGrid Items.Count:0
    2 BiosInformation             DataGrid System.Windows.Controls.DataGrid Items.Count:0
    3 BiosInformationExtension    DataGrid System.Windows.Controls.DataGrid Items.Count:0
    4 OperatingSystem             DataGrid System.Windows.Controls.DataGrid Items.Count:0
    5 HotFix                      DataGrid System.Windows.Controls.DataGrid Items.Count:0
    6 ComputerSystem              DataGrid System.Windows.Controls.DataGrid Items.Count:0
    7 ComputerSystemExtension     DataGrid System.Windows.Controls.DataGrid Items.Count:0
    8 Processor                   DataGrid System.Windows.Controls.DataGrid Items.Count:0
    9 ProcessorExtension          DataGrid System.Windows.Controls.DataGrid Items.Count:0
   10 Disk                        DataGrid System.Windows.Controls.DataGrid Items.Count:0
   11 DiskExtension               DataGrid System.Windows.Controls.DataGrid Items.Count:0
   12 DiskPartition               DataGrid System.Windows.Controls.DataGrid Items.Count:0
   13 DiskVolume                  DataGrid System.Windows.Controls.DataGrid Items.Count:0
   14 Network                     DataGrid System.Windows.Controls.DataGrid Items.Count:0
   15 NetworkExtension            DataGrid System.Windows.Controls.DataGrid Items.Count:0
   16 ServiceSlot                 ComboBox System.Windows.Controls.ComboBox Items.Count:0
   17 ServiceDisplay              DataGrid System.Windows.Controls.DataGrid Items.Count:0
   18 ServiceProperty             ComboBox System.Windows.Controls.ComboBox Items.Count:2
   19 ServiceFilter               TextBox  System.Windows.Controls.TextBox
   20 Service                     DataGrid System.Windows.Controls.DataGrid Items.Count:0
   21 ServiceExtension            DataGrid System.Windows.Controls.DataGrid Items.Count:0
   22 ServiceGet                  Button   System.Windows.Controls.Button: Get
   23 ServiceSet                  Button   System.Windows.Controls.Button: Apply
   24 ServiceBypassBuild          CheckBox System.Windows.Controls.CheckBox Content:Skip Build/Version Check IsChecked:False
   25 ServiceBypassEdition        ComboBox System.Windows.Controls.ComboBox Items.Count:3
   26 ServiceBypassLaptop         CheckBox System.Windows.Controls.CheckBox Content:Enable Laptop Tweaks IsChecked:False
   27 ServiceDisplayActive        CheckBox System.Windows.Controls.CheckBox Content:Active IsChecked:False
   28 ServiceDisplayInactive      CheckBox System.Windows.Controls.CheckBox Content:Inactive IsChecked:False
   29 ServiceDisplaySkipped       CheckBox System.Windows.Controls.CheckBox Content:Skipped IsChecked:False
   30 ServiceMiscSimulate         CheckBox System.Windows.Controls.CheckBox Content:Simulate Changes [Dry Run] IsChecked:False
   31 ServiceMiscXbox             CheckBox System.Windows.Controls.CheckBox Content:Skip All Xbox Services IsChecked:False
   32 ServiceMiscChange           CheckBox System.Windows.Controls.CheckBox Content:Allow Change of Service State IsChecked:False
   33 ServiceMiscStopDisabled     CheckBox System.Windows.Controls.CheckBox Content:Stop Disabled Services IsChecked:False
   34 ServiceLogServiceSwitch     CheckBox System.Windows.Controls.CheckBox Content:Services IsChecked:False
   35 ServiceLogServiceFile       TextBox  System.Windows.Controls.TextBox
```

```
   36 ServiceLogServiceBrowse    Button   System.Windows.Controls.Button: Browse
   37 ServiceLogScriptSwitch     CheckBox System.Windows.Controls.CheckBox Content:Script IsChecked:False
   38 ServiceLogScriptFile       TextBox  System.Windows.Controls.TextBox
   39 ServiceLogScriptBrowse     Button   System.Windows.Controls.Button: Browse
   40 ServiceRegSwitch           CheckBox System.Windows.Controls.CheckBox Content:*.reg IsChecked:False
   41 ServiceRegFile             TextBox  System.Windows.Controls.TextBox
   42 ServiceRegBrowse           Button   System.Windows.Controls.Button: Browse
   43 ServiceCsvSwitch           CheckBox System.Windows.Controls.CheckBox Content:*.csv IsChecked:False
   44 ServiceCsvFile             TextBox  System.Windows.Controls.TextBox
   45 ServiceCsvBrowse           Button   System.Windows.Controls.Button: Browse
   46 ServiceDevErrors           CheckBox System.Windows.Controls.CheckBox Content:Diagnostic Output [On Error] IsChecked:False
   47 ServiceDevLog              CheckBox System.Windows.Controls.CheckBox Content:Enable Development Logging IsChecked:False
   48 ServiceDevConsole          CheckBox System.Windows.Controls.CheckBox Content:Enable Console IsChecked:False
   49 ServiceDevReport           CheckBox System.Windows.Controls.CheckBox Content:Enable Diagnostic IsChecked:False
   50 ControlGlobalRestorePoint  CheckBox System.Windows.Controls.CheckBox Content:Create Restore Point IsChecked:False
   51 ControlGlobalRestart       CheckBox System.Windows.Controls.CheckBox Content:Restart When Done IsChecked:False
   52 ControlGlobalShowSkipped   CheckBox System.Windows.Controls.CheckBox Content:Show Skipped Items IsChecked:False
   53 ControlGlobalVersionCheck  CheckBox System.Windows.Controls.CheckBox Content:Check for Update IsChecked:False
   54 ControlGlobalInternetCheck CheckBox System.Windows.Controls.CheckBox Content:Skip Internet Check IsChecked:False
   55 ControlBackupSave          Button   System.Windows.Controls.Button: Save Settings
   56 ControlBackupLoad          Button   System.Windows.Controls.Button: Load Settings
   57 ControlBackupWinDefault    Button   System.Windows.Controls.Button: Windows Default
   58 ControlBackupResetDefault  Button   System.Windows.Controls.Button: Reset All Items
   59 ControlSlot                ComboBox System.Windows.Controls.ComboBox Items.Count:14
   60 ControlProperty            ComboBox System.Windows.Controls.ComboBox Items.Count:2
   61 ControlFilter              TextBox  System.Windows.Controls.TextBox
   62 ControlOutput              DataGrid System.Windows.Controls.DataGrid Items.Count:0
   63 ControlOutputExtension     DataGrid System.Windows.Controls.DataGrid Items.Count:0
   64 ControlOutputApply         Button   System.Windows.Controls.Button: Apply
   65 ControlOutputDontApply     Button   System.Windows.Controls.Button: Do not apply...
   66 ControlFeatureProperty     ComboBox System.Windows.Controls.ComboBox Items.Count:2
   67 ControlFeatureFilter       TextBox  System.Windows.Controls.TextBox
   68 ControlFeature             DataGrid System.Windows.Controls.DataGrid Items.Count:0
   69 ControlFeatureExtension    DataGrid System.Windows.Controls.DataGrid Items.Count:0
   70 ControlFeatureApply        Button   System.Windows.Controls.Button: Apply
   71 ControlFeatureDontApply    Button   System.Windows.Controls.Button: Do not apply...
   72 ControlAppXProperty        ComboBox System.Windows.Controls.ComboBox Items.Count:4
   73 ControlAppXFilter          TextBox  System.Windows.Controls.TextBox
   74 ControlAppX                DataGrid System.Windows.Controls.DataGrid Items.Count:0
   75 ControlAppXExtension       DataGrid System.Windows.Controls.DataGrid Items.Count:0
   76 ControlAppXApply           Button   System.Windows.Controls.Button: Apply
   77 ControlAppXDontApply       Button   System.Windows.Controls.Button: Do not apply...

PS Prompt:\>
```

I won't cover what the `[System.Xml.XmlNodeReader]` object does, nor the `[System.Windows.Window]` object, because THOSE (properties/classes) are REALLY dense objects with a LOT of internal properties and methods that relate to the `[Windows Presentation Framework]`.

If the day comes where I have to go over all of that stuff...?
Maybe I'll have a coffee mug in my hand that says `[Microsoft Corporation]` or something.

Conclusion

There is something I would like to add, regarding the ongoing war between the titans of technology.
I keep having this dialogue in my mind, that is sorta covered in the movie with Ashton Kutcher...
A conversation between Steve Jobs, and Bill Gates...

Jobs : Dude, Macintosh is gonna blow everybody out of the water, and there's nothin' you can even do about it...
Gates : Yeah right, dude.
        scoffs You don't get it, Steve...
Jobs : What don't I get, Bill...?
Gates : You don't need to be the best at everything, to give people what they want.
Jobs : Well, that's what I LIVE for...
<to be continued>

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