# Introduction

I created this PowerShell script in response to a Windows Secrets Lounge (WSL) thread on avoiding UAC prompts. One of the responses was that the solution of creating Scheduled Tasks with Shortcuts was too involved, so I thought I’d make it easier.

The script Create-NoUACTaskShortcut does just that by allowing the user to fill in a dialog box and then automatically create both the Scheduled Task running the program with “Highest Privileges”, i.e. Administrator Rights and then creating a Shortcut to allow the task to be easily run.

There only 2 required values: First a name for the Task and Shortcut and a Task Command, i.e., the program to execute. Of course this is insufficient for many cases so there are optional parameters the user can specify depending on the situation. This short manual lists what each item in the dialog accomplisher and how to use it. I’ve made my best shot at error checking to make sure the provided items are valid but of course in this situation there is no way I can fully check all the combinations of inputs and have to just take some things on faith.

I’ve also tried to provide intelligent defaults where possible, e.g. storing the Shortcut on the Desktop if the user doesn’t specify a location.

If you have any suggestions on the coding of the script please leave them on the Windows Secrets Lounge in the forum Windows Programming) and thread where you found the script. As I am teaching myself PowerShell I welcome all comments to make the code more efficient. I’ll also be most happy to incorporate new features if the code to do so is provided.

You can read the comments in the code, at the end of this document, to see the history and latest changes.

Requirements:

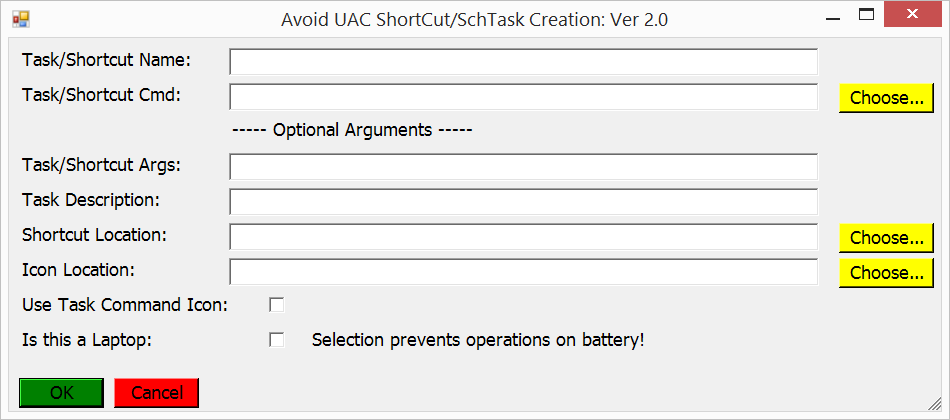
Windows 8.1 or Windows Server 2012 R2

PowerShell 4.0 as Administrator

-- RetiredGeek

# Program Operation:

## User Input Dialog Box:



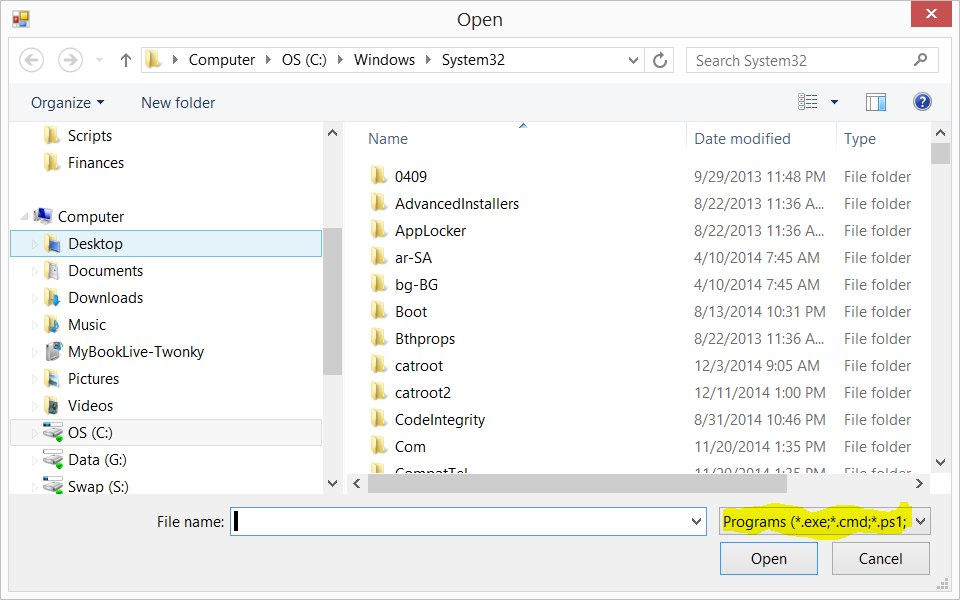
## Dialog Box Arguments:

| Item | Shortcut | Task | Notes |
| --- | --- | --- | --- |
| 1. Task/Shortcut Name: |  |  | This becomes the title for the shortcut and task. |
| 1. Task/Shortcut Cmd: |  |  | This is the command executed by the task and is used for the shortcut icon if the Use Task Command Icon box is checked. You can provide this value via the Choose… button or by cut/paste from an existing shortcut. . The default file filter shows .exe, .cmd, .bat, and .ps1 files only. You can select the All Files filter if necessary. |
| Optional Arguments | | | |
| 1. Task/Shortcut Args: |  |  | If the task command requires any arguments they go here. |
| 1. Task Description: |  |  | Used for the description property of the task. |
| 1. Shortcut Location: |  |  | This will default to the Desktop or you can direct it to a folder. You can type this value or use the Choose… button to browse for the folder where you want to place the shortcut |
| 1. Icon Location: |  |  | If you are using custom icons provide a fully qualified file specification (by typing or use of the Choose… button). You can also access other than the first icon in a file by appending a comma and the relative icon number. Icons are numbered starting with zero. The default file filter shows .exe, .dll, and .ico files only. You can select the All Files filter if necessary. |
| 1. Use Task Command Icon: |  |  | This will use the default icon {icon zero} contained in the Task/ShortcutCmd file. If neither the Icon Location or this option is selected the default windows shortcut icon will be used. |
| 1. Is this a Laptop: |  |  | If selected the task will be set so that you cannot start the task if the machine is on battery power and the task will terminate if the machine switches to battery power! |

**NOTE:** Do NOT use any quotes when filling in the arguments!

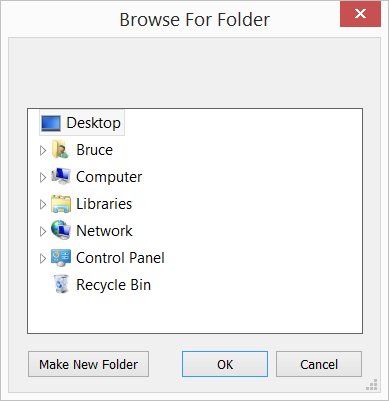
Please Note: In all circumstances that the setting ***Stop On Idle End*** is set to False! Since this program is designed to make tasks for “ON DEMAND” programs I chose to make this a non-choice setting!

## File Browser Dialog:

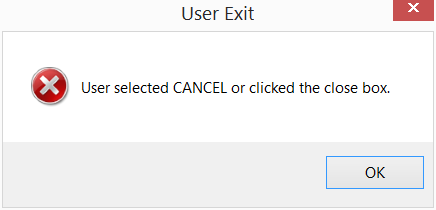


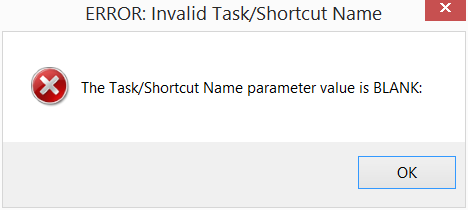
Note: the File name: filter, in yellow, { Programs (\*.exe,\*.cmd,\*.ps1,\*.bat}

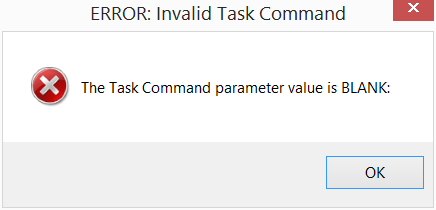
## Folder Browser Dialog:

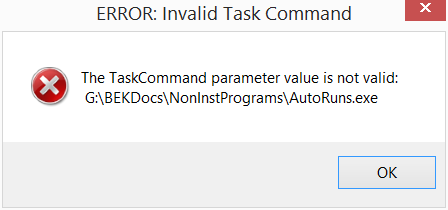


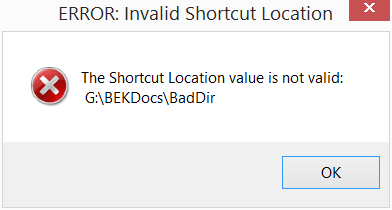
## Messages:

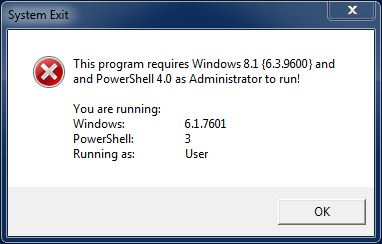
This message is here just in case the user accidentally clicks the wrong place instead of just getting a blank screen.

The required parameter Task/Shortcut Name was left blank.

The required parameter Task Command was left blank.

The provided task Command could not be validated, e.g. in the example AutoRuns.exe doesn’t exist in the directory specified.

The specified drive:\path for the shortcut does not exist.

This message will only be displayed if your system environment does NOT meet the minimum requirements to run the script.

# Setup:

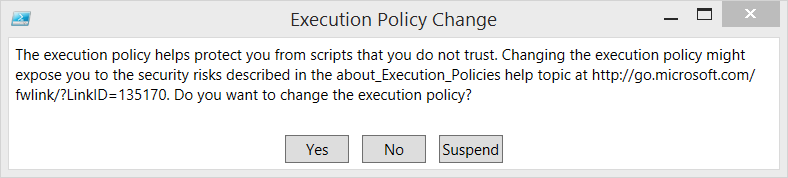
PowerShell comes out of the box with its execution policy set at the highest level (Restricted). For most users who not have the facilities to sign their code you need to adjust this setting by using the Set-ExecutionPolicy cmdlet and check the setting with the Get-ExecutionPolicy cmdlet.

NOTE: You MUST start PowerShell ***as Administrator*** to be able to change the ExecutionPolicy!

PS G:\bekdocs\scripts> Get-ExecutionPolicy

Restricted

PS G:\bekdocs\scripts> Set-ExecutionPolicy RemoteSigned



PS G:\bekdocs\scripts> Get-ExecutionPolicy

RemoteSigned

PS G:\bekdocs\scripts>

Execution Policy Settings:

* Restricted No scripts can be run. Windows PowerShell can be used only in interactive mode.
* AllSigned Only scripts signed by a trusted publisher can be run.
* RemoteSigned Downloaded scripts must be signed by a trusted publisher before they can be run. (*Author’s recommendation for most users*)
* Unrestricted No restrictions; all Windows PowerShell scripts can be run.

Once you have this out of the way you can now run the program and use it to setup a shortcut to run itself with the following parameters:

Task/Shortcut Name: Your Call

Task/Shortcut Command: C:\Windows\system32\WindowsPowerShell\v1.0\PowerSh ell.exe

Task/Shortcut Args: d:\path\Create-NoUACTaskShortcut.ps1  
Where: d:\path is the location of the ps1 file.

To use an icon from Windows:

Icon Location: C:\Windows\System32\SHELL32.dll,n  
Where: n is the icon number.

Or just check the Use Task/Shortcut Command Icon: option.

This program also needs Administrator Privileges to run. This means that you can’t run the program by Right-Clicking on it in File Explorer and selecting “Run with PowerShell”.

To run the 1st time:

1. Ssearch for : PowerShell.exe.
2. Right-Click: PowerShell.exe.
3. Select: Run as administrator.
4. In the PS window type: *d:\path*\Create-NoUACTaskShortcut.ps1
5. If the menu does not appear check the taskbar to see if the PS icon is doubled (2 windows). Hover your mouse over the icons and click on the one with the menu.
6. Now you can setup the shortcut above.

Script Code:

<#

+----------------------------------------------------------------------+

| Program : Create-NoUACTaskShortcut.ps1 |

| Programmer : RetiredGeek (WSL) aka: ComputerMentor |

| Created : 24 Nov 2014 |

| Updated : 12 Dec 2014 |

| Description : Prompts the user with a dialog for the data required |

| to create a Shortcut and Scheduled task for a program |

| that normally requires Elevated Privileges prompt. |

| Requirements : Windows 8.1 - PowerShell 4.0 |

| Last Updated : |

| Current Vers : 2.0 |

| 11/26/2014 : Added check for Windows & PowerShell minimum reqs. |

| 11/27/2014 : Added suggestions by Cliff.H on WSL, Thanks Cliff |

| 12/06/2014 : Added Menu support for new Task Creation options |

| See comments this date in Create-NoUACTask |

| 12/11/2014 : Functionalized form creation activities. |

| 12/12/2014 : Created Browser buttons for filename/folder selection.|

+----------------------------------------------------------------------+

#>

Function Create-NoUACTask {

<#+----------------------------------------------------------------------+

| Function : Create-NoUACTask |

| Description : Create Task to get around UAC Prompt |

| Programmer : RetiredGeek (WSL) aka: ComputerMentor |

| Created : 21 Nov 2014 |

| Last Updated : 06 Dec 2014 |

| Current Vers : 1.2 |

| 12/06/14 : Added laptop settings to allow on battery operations. |

| Set Stop on Idle end to FALSE |

+----------------------------------------------------------------------+

#>

param (

[Parameter(Mandatory=$true)]

[String]$TaskName,

[Parameter(Mandatory=$true)]

[String]$TaskCommand,

[Parameter(Mandatory=$false)]

[String]$TaskArg,

[Parameter(Mandatory=$false)]

[String]$TaskDesc="No Description Supplied",

[Parameter(Mandatory=$false)]

[Boolean]$Laptop=$false

)

$TaskSet = New-ScheduledTaskSettingsSet -Compatibility Win8

$TaskSet.DisallowStartIfOnBatteries = $Laptop

$TaskSet.StopIfGoingOnBatteries = $Laptop

$TaskSet.IdleSettings.StopOnIdleEnd = $false

$TaskAction = New-ScheduledTaskAction -Execute `"$TaskCommand`" `

-Argument `"$TaskArg`"

Register-ScheduledTask -Action $TaskAction `

-TaskName "$TaskName" `

-User "$env:USERDOMAIN\$env:username" `

-RunLevel Highest `

-Description "$TaskDesc" `

-Settings $TaskSet

} #End Function Create-NoUACTask

Function Create-NoUACShortCut {

<#

+-------------------------------------------------------------------------+

| Function : Create-NoUACTask |

| Programmer : RetiredGeek (WSL) aka: ComputerMentor |

| Created : 24 Nov 2014 |

| Last Updated : 28 Nov 2014 |

| Current Vers : 1.3 |

| |

|Where: |

| $TaskName - is the Name of the Task/Shortcut to create |

| $TaskCommand - is the Command Executed by the Task |

| $DestinationPath - is the location to store the created shortcut |

| ----- Use only ONE of the Following ---- |

| $ShortcutIcon - is the path & number of the icon to use. |

| ex: "C:\MyIcon.ico,0" - optional |

| $UseTaskCommandIcon - will select the first icon in the TaskCommand |

| file. |

+-------------------------------------------------------------------------+

#>

param ( [Parameter(Mandatory=$true)]

[String]$TaskName,

[Parameter(Mandatory=$false)]

[String]$TaskCommand,

[Parameter(Mandatory=$false)]

[String]$DestinationPath="$env:UserProfile\Desktop",

[Parameter(Mandatory=$false)]

[String]$ShortcutIcon="NO",

[switch]$UseTaskCommandIcon

)

$WshShell = New-Object -comObject WScript.Shell

$Shortcut = $WshShell.CreateShortcut("$DestinationPath\$TaskName.lnk")

$Shortcut.TargetPath = "C:\Windows\System32\schtasks.exe"

$Shortcut.Arguments = "`/run `/TN " + "`"" + $TaskName +"`""

$Shortcut.WindowStyle = 7

If ($ShortcutIcon -ne 'NO') {

$Shortcut.IconLocation = $ShortcutIcon

}

Else {

If ($UseTaskCommandIcon.IsPresent) {

$Shortcut.IconLocation = $TaskCommand

}

}

$Shortcut.Save()

} #End Function Create-NoUACShortCut

Function New-Font {

<#+----------------------------------------------------------------------+

| Function : New-Font |

| Description : Create a new font object |

| Programmer : RetiredGeek (WSL) aka: ComputerMentor |

| Created : 11 Dec 2014 |

| Last Updated : |

| Current Vers : 1.0 |

+----------------------------------------------------------------------+

#>

param(

[Parameter(Mandatory=$true)]

[String]$FontName,

[Parameter(Mandatory=$false)]

[String]$FontStyle='Regular',

[Parameter(Mandatory=$false)]

[Single]$FontSize=12

)

New-Object System.Drawing.Font("$FontName",$FontSize, `

[System.Drawing.FontStyle]::$FontStyle)

} #End New-Font

Function New-Button {

<#+----------------------------------------------------------------------+

| Function : New-Button |

| Description : Create a new Button object |

| Programmer : RetiredGeek (WSL) aka: ComputerMentor |

| Created : 11 Dec 2014 |

| Last Updated : |

| Current Vers : 1.0 |

+----------------------------------------------------------------------+

#>

param(

[Parameter(Mandatory=$true)]

[Int]$BtnLocX, #Distance from left edge of form

[Parameter(Mandatory=$true)]

[Int]$BtnLocY, #Distance from top of form

[Parameter(Mandatory=$true)]

[Int]$BtnWidth,

[Parameter(Mandatory=$true)]

[Int]$BtnHeight,

[Parameter(Mandatory=$true)]

[String]$BtnText,

[Parameter(Mandatory=$true)]

$BtnFont,

[Parameter(Mandatory=$true)]

[String]$BtnBackColor,

[Parameter(Mandatory=$false)]

[String]$Result ="None" # Yes No Cancel Ok

)

$Button = New-Object System.Windows.Forms.Button

$Button.Location = New-Object System.Drawing.Size($BtnLocX,$BtnLocY)

$Button.Size = New-Object System.Drawing.Size($BtnWidth,$BtnHeight)

$Button.Text = "$BtnText"

$Button.BackColor = "$BtnBackColor"

$Button.Font = $BtnFont

If ($Result -ne "None") {

$Button.DialogResult = [System.Windows.Forms.DialogResult]::$Result

}

$Button

} #End New-Button

Function New-Label {

<#+----------------------------------------------------------------------+

| Function : New-Label |

| Description : Create a new Label object |

| Programmer : RetiredGeek (WSL) aka: ComputerMentor |

| Created : 11 Dec 2014 |

| Last Updated : |

| Current Vers : 1.0 |

+----------------------------------------------------------------------+

#>

param(

[Parameter(Mandatory=$true)]

[Int]$LabelLocX, #Distance from left edge of form

[Parameter(Mandatory=$true)]

[Int]$LabelLocY, #Distance from top of form

[Parameter(Mandatory=$true)]

[Int]$LabelWidth,

[Parameter(Mandatory=$true)]

[Int]$LabelHeight,

[Parameter(Mandatory=$true)]

[String]$LabelText,

[Parameter(Mandatory=$true)]

$LabelFont

)

$objLabel = New-Object System.Windows.Forms.Label

$objLabel.Location = New-Object System.Drawing.Size($LabelLocX,$LabelLocY)

$objLabel.Size = New-Object System.Drawing.Size($LabelWidth,$LabelHeight)

$objLabel.Text = "$LabelText"

$objLabel.Font = $LabelFont

$objLabel

} #End New-Label

Function New-TextBox {

<#+----------------------------------------------------------------------+

| Function : New-TextBox |

| Description : Create a new TextBox object |

| Programmer : RetiredGeek (WSL) aka: ComputerMentor |

| Created : 11 Dec 2014 |

| Last Updated : |

| Current Vers : 1.0 |

+----------------------------------------------------------------------+

#>

param(

[Parameter(Mandatory=$true)]

[Int]$TextBoxLocX, #Distance from left edge of form

[Parameter(Mandatory=$true)]

[Int]$TextBoxLocY, #Distance from top of form

[Parameter(Mandatory=$true)]

[Int]$TextBoxWidth,

[Parameter(Mandatory=$true)]

[Int]$TextBoxHeight,

[Parameter(Mandatory=$true)]

$TextBoxFont,

[Parameter(Mandatory=$false)]

[String]$TextBoxText #Initialization Value

)

$objTextBox = New-Object System.Windows.Forms.TextBox

$objTextBox.Location = `

New-Object System.Drawing.Size($TextBoxLocX,$TextBoxLocY)

$objTextBox.Size = `

New-Object System.Drawing.Size($TextBoxWidth,$TextBoxHeight)

$objTextBox.Font = $TextBoxFont

If ($TextBoxText -ne "") {

$objTextBox.Text = "$TextBoxText"

}

$objTextBox

} #End New-TextBox

Function New-CheckBox {

<#+----------------------------------------------------------------------+

| Function : New-CheckBox |

| Description : Create a new ChecktBox object |

| Programmer : RetiredGeek (WSL) aka: ComputerMentor |

| Created : 11 Dec 2014 |

| Last Updated : |

| Current Vers : 1.0 |

+----------------------------------------------------------------------+

#>

param(

[Parameter(Mandatory=$true)]

[Int]$CheckBoxLocX, #Distance from left edge of form

[Parameter(Mandatory=$true)]

[Int]$CheckBoxLocY, #Distance from top of form

[Parameter(Mandatory=$true)]

[Int]$CheckBoxWidth,

[Parameter(Mandatory=$true)]

[Int]$CheckBoxHeight,

[Parameter(Mandatory=$true)]

$CheckBoxFont

)

$ObjCheckBox = New-Object System.Windows.Forms.CheckBox

$ObjCheckBox.Location = New-Object `

System.Drawing.Size($CheckBoxLocX,$CheckBoxLocY)

$ObjCheckBox.Size = New-Object `

System.Drawing.Size($CheckBoxWidth,$CheckBoxHeight)

$ObjCheckBox.Font = $lbfont

$ObjCheckBox

}

# ------------------- Main ------------------------

Add-Type -AssemblyName System.Windows.Forms #PS V3+

Add-Type -AssemblyName System.Drawing

$ErrorActionPreference = 'Stop'

$MsgBox = [Windows.Forms.MessageBox]

$Buttons = [Windows.Forms.MessageBoxButtons]

$MBIcons = [Windows.Forms.MessageBoxIcon]

<# \*\*\*\* Check Operating Requirements \*\*\*

\*\*\*\* Windows 8.1 \*\*\*

\*\*\*\* PowerShell 4.0 as Admin \*\*\*

#>

$CurOS = Get-CimInstance Win32\_OperatingSystem

$FullVersion = $CurOS.Version

$OSVerItems = $CurOs.Version.split('.')

$OSVer = $OSVerItems[0]\*10 + $OSVerItems[1]

$CurPSVer = $psversiontable.psversion.Major

If (-NOT ([Security.Principal.WindowsPrincipal] `

[Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole(`

[Security.Principal.WindowsBuiltInRole] "Administrator"))

{$PSPriv = "User"} Else {$PSPriv = "Administrator"}

If ($OSVer -lt 63 -or $CurPSVer -lt 4 -or $PSPriv -eq "User") {

#\*\*\* Now strings incorporate ALL spaces thus no identing!

$Message = `

@("This program requires Windows 8.1 {6.3.9600} and

and PowerShell 4.0 as Administrator to run!

You are running:

Windows:`t`t$FullVersion

PowerShell:`t$CurPSVer

Running as:`t$($PSPriv)")

[void]$MsgBox::Show("$Message","System Exit", `

$Buttons::OK, $MBIcons::Stop)

Exit

}

# \*\*\* End Check Operating Requirements \*\*\*

# \*\*\* Setup File/Folder Browsers \*\*\*

$FileBrowser = New-Object Windows.Forms.OpenFileDialog

$FolderBrowser = New-Object Windows.Forms.FolderBrowserDialog

$FolderBrowser.RootFolder = [System.Environment+SpecialFolder]'Desktop'

# Font styles are: Regular, Bold, Italic, Underline, Strikeout

$Font = New-Font -FontName "Tahoma" -FontStyle "Regular" -FontSize 11

$LBFont = New-Font -FontName "Courier New" -FontSize 11 -FontStyle "Regular"

#Form Object Location/Width/Height sizes

$LNSpacing = 35

$TNameLoc = 10

$TCmdLoc = $TNameLoc+$LNSpacing

$OptArgLblLoc = $TCmdLoc+$LNSpacing

$TArgsLoc = $OptArgLblLoc+$LNSpacing

$TDescLoc = $TArgsLoc+$LNSpacing

$SCutLoc = $TDescLoc+$LNSpacing

$IconLoc = $SCutLoc+$LNSpacing

$UseTIconLoc = $IconLoc+$LNSpacing

$LTLoc = $UseTIconLoc+$LNSpacing

$OkCanBtnLoc = 340

$StdBtnWidth = 85

$LblWidth = 210

$TextBoxWIdth = 590

$OKButton = New-Button -BtnLocX 10 -BtnLocY $OkCanBtnLoc `

-BtnWidth $StdBtnWidth -BtnHeight 30 `

-BtnText "OK" -BtnFont $Font `

-BtnBackColor "Green" -Result "OK"

$CancelButton = New-Button -BtnLocX 105 -BtnLocY $OkCanBtnLoc `

-BtnWidth $StdBtnWidth -BtnHeight 30 `

-BtnText "Cancel" -BtnFont $Font `

-BtnBackColor "Red" -Result "Cancel"

#File Browser Buttons:

$TCmdBrBtn = New-Button -BtnLocX 830 -BtnLocY $TCmdLoc `

-BtnWidth ($StdBtnWidth+10) -BtnHeight 30 `

-BtnText "Choose..." -BtnFont $Font `

-BtnBackColor "Yellow"

$TCmdBrBtn.Add\_Click({$FileBrowser.Filter = `

"Programs|\*.exe;\*.cmd;\*.ps1;\*.bat|All Files|\*.\*";

$FileBrowser.ShowDialog();`

$TextBoxCmd.Text = $FileBrowser.Filename;

$FileBrowser.FileName = $null;

$FileBrowser.Filter = ""})

$IconBrBtn = New-Button -BtnLocX 830 -BtnLocY $IconLoc `

-BtnWidth ($StdBtnWidth+10) -BtnHeight 30 `

-BtnText "Choose..." -BtnFont $Font `

-BtnBackColor "Yellow"

$IconBrBtn.Add\_Click({$FileBrowser.Filter = `

"Icon Files|\*.exe;\*.dll;\*.ico|All Files|\*.\*";

$FIleBrowser.ShowDialog();`

$TextBoxIC.Text = $FileBrowser.Filename;

$FileBrowser.FileName = $null;

$FileBrowser.Filter = ""})

#Folder Browser Buttons:

$SCutBrBtn = New-Button -BtnLocX 830 -BtnLocY $SCutLoc `

-BtnWidth ($StdBtnWidth+10) -BtnHeight 30 `

-BtnText "Choose..." -BtnFont $Font `

-BtnBackColor "Yellow"

$SCutBrBtn.Add\_Click({$FolderBrowser.ShowDialog();`

$TextBoxDP.Text = $FolderBrowser.SelectedPath;

$FolderBrowser.SelectedPath = ""})

#Labels:

$lblTName = New-Label -LabelLocX 10 -LabelLocY $TNameLoc `

-LabelWidth $LblWidth -LabelHeight 25 `

-LabelText "Task/Shortcut Name:" `

-LabelFont $Font

$lblTCmd = New-Label -LabelLocX 10 -LabelLocY $TCmdLoc `

-LabelWidth $LblWidth -LabelHeight 25 `

-LabelText "Task/Shortcut Cmd:" `

-LabelFont $Font

$lblOptArgs = New-Label `

-LabelLocX 220 -LabelLocY $OptArgLblLoc `

-LabelWidth 300 -LabelHeight 25 `

-LabelText "----- Optional Arguments -----" `

-LabelFont $Font

$lblTArgs = New-Label -LabelLocX 10 -LabelLocY $TArgsLoc `

-LabelWidth $LblWidth -LabelHeight 25 `

-LabelText "Task/Shortcut Args:" `

-LabelFont $Font

$lblTDesc = New-Label -LabelLocX 10 -LabelLocY $TDescLoc `

-LabelWidth $LblWidth -LabelHeight 25 `

-LabelText "Task Description:" `

-LabelFont $Font

$lblSCutDest = New-Label -LabelLocX 10 -LabelLocY $SCutLoc `

-LabelWidth $LblWidth -LabelHeight 25 `

-LabelText "Shortcut Location:" `

-LabelFont $Font

$lblIconLoc = New-Label -LabelLocX 10 -LabelLocY $IconLoc `

-LabelWidth $LblWidth -LabelHeight 25 `

-LabelText "Icon Location:" `

-LabelFont $Font

$lblUseTIC = New-Label -LabelLocX 10 -LabelLocY $UseTIconLoc `

-LabelWidth ($LblWidth+40) -LabelHeight 25 `

-LabelText "Use Task Command Icon:" `

-LabelFont $Font

$lblLaptop = New-Label -LabelLocX 10 -LabelLocY $LTLoc `

-LabelWidth ($LblWidth+40) -LabelHeight 25 `

-LabelText "Is this a Laptop:" `

-LabelFont $Font

$lblLaptopInst = New-Label `

-LabelLocX 300 -LabelLocY $LTLoc `

-LabelWidth ($LblWidth+200) -LabelHeight 25 `

-LabelText "Selection prevents operations on battery!" `

-LabelFont $Font

#Text Boxes:

$TextBoxTN = New-TextBox -TextBoxLocX 220 -TextBoxLocY $TNameLoc `

-TextBoxWidth $TextBoxWIdth -TextBoxHeight 25 `

-TextBoxFont $LBFont

$TextBoxCmd = New-TextBox -TextBoxLocX 220 -TextBoxLocY $TCmdLoc `

-TextBoxWidth $TextBoxWIdth -TextBoxHeight 25 `

-TextBoxFont $LBFont

$TextBoxArg = New-TextBox -TextBoxLocX 220 -TextBoxLocY $TArgsLoc `

-TextBoxWidth $TextBoxWIdth -TextBoxHeight 25 `

-TextBoxFont $LBFont

$TextBoxTD = New-TextBox -TextBoxLocX 220 -TextBoxLocY $TDescLoc `

-TextBoxWidth $TextBoxWIdth -TextBoxHeight 25 `

-TextBoxFont $LBFont

$TextBoxDP = New-TextBox -TextBoxLocX 220 -TextBoxLocY $SCutLoc `

-TextBoxWidth $TextBoxWIdth -TextBoxHeight 25 `

-TextBoxFont $LBFont

$TextBoxIC = New-TextBox -TextBoxLocX 220 -TextBoxLocY $IconLoc `

-TextBoxWidth $TextBoxWIdth -TextBoxHeight 25 `

-TextBoxFont $LBFont

#CheckBoxes:

$CheckBoxUseTC = New-CheckBox -CheckBoxLocX 260 -CheckBoxLocY $UseTIconLoc `

-CheckBoxWidth 25 -CheckBoxHeight 25 `

-CheckBoxFont $LBFont

$CheckBoxLaptop = New-CheckBox -CheckBoxLocX 260 -CheckBoxLocY $LTLoc `

-CheckBoxWidth 25 -CheckBoxHeight 25 `

-CheckBoxFont $LBFont

#Form Setup:

$objForm = New-Object System.Windows.Forms.Form

$objForm.Text = "Avoid UAC ShortCut/SchTask Creation:"

$objForm.Size = New-Object System.Drawing.Size(950,420)

$objForm.StartPosition = "CenterScreen"

$objForm.AcceptButton = $OKButton

$objForm.CancelButton = $CancelButton

$objForm.Controls.AddRange( `

@($OKButton,$CancelButton, `

$lblTName,$lblTCmd,$lblTArgs,$lblOptArgs, `

$lblTDesc,$lblSCutDest,$lblIconLoc,$lblUseTIC, `

$lblLaptop,$lblLaptopInst,

$TextBoxTN,$TextBoxCmd,$TextBoxArg, `

$TextBoxTD,$TextBoxDP,$TextBoxIC, `

$CheckBOxUseTC,$CheckBoxLaptop, `

$TCmdBrBtn,$IconBrBtn,$SCutBrBtn))

$objForm.Topmost = $True

$dialogResult = $objForm.ShowDialog()

#Process Form Contents:

if ($dialogResult -eq [System.Windows.Forms.DialogResult]::OK)

{

$TaskName = $TextBoxTN.Text

$TaskCommand = $TextBoxCmd.Text

$TaskArg =$TextBoxArg.Text

$TaskDesc = $TextBoxTD.Text

$DestinationPath = $TextBoxDP.Text

$ShortcutIcon = $TextBoxIC.Text

$UseTaskCommandIcon = $objcbusetc.checked

$LapTop = $objLaptop.Checked

$FoundError = $false

If ($TaskName -eq "") {

$Message = "The Task/Shortcut Name parameter value is BLANK:"

$MsgBox::Show("$Message","ERROR: Invalid Task/Shortcut Name", `

$Buttons::OK, $MBIcons::Stop)

$FoundError = $true

}

If ($TaskCommand -eq "") {

$Message = "The Task Command parameter value is BLANK:"

$MsgBox::Show("$Message","ERROR: Invalid Task Command", `

$Buttons::OK, $MBIcons::Stop)

$FoundError = $true

}

Else {

If (!(Test-Path -Path $TaskCommand)) {

$Message = "The TaskCommand parameter value is not valid:"

$MsgBox::Show("$Message `n $TaskCommand","ERROR: Invalid Task Command", `

$Buttons::OK, $MBIcons::Stop)

$FoundError = $true

} # End If (!(Test-Path -Path $TaskCommand))

}

If ($DestinationPath -ne "") {

If (!(Test-Path -Path $DestinationPath)) {

$Message = "The Shortcut Location value is not valid:"

$MsgBox::Show("$Message `n $DestinationPath","ERROR: Invalid Shortcut Location", `

$Buttons::OK, $MBIcons::Stop)

$FoundError = $true

} # End If (!(Test-Path -Path $TaskCommand))

}

ELse {

$DestinationPath = "$env:UserProfile\Desktop"

}

If ($FoundError) { EXIT }

#------ Error Checks PASSED Create Task & Shortcut ---

If ($TaskDesc -eq "") {

Create-NoUACTask -TaskCommand "$TaskCommand" `

-TaskArg $TaskArg `

-TaskName $TaskName `

-LapTop $LapTop

}

Else {

Create-NoUACTask -TaskCommand "$TaskCommand" `

-TaskArg $TaskArg `

-TaskName $TaskName `

-TaskDesc "$TaskDesc"`

-LapTop $LapTop

}

If ($UseTaskCommandIcon) {

Create-NoUACShortCut -TaskName $TaskName `

-DestinationPath $DestinationPath `

-TaskCommand "$TaskCommand" `

-UseTaskCommandIcon

}

Else {

If ($ShortcutIcon -ne "") {

Create-NoUACShortCut -TaskName $TaskName `

-DestinationPath $DestinationPath `

-ShortcutIcon $ShortcutIcon

}

Else {

Create-NoUACShortCut -TaskName $TaskName `

-DestinationPath $DestinationPath

}

}

} #End if ($dialogResult -eq [System.Windows.Forms.DialogResult]::OK)

Else

{

$Message = "User selected CANCEL or clicked the close box."

$MsgBox::Show("$Message","User Exit", `

$Buttons::OK, $MBIcons::Stop)

}

$objForm.dispose()