

Migrate User Printing to Software Center



A mitigation strategy to help Sysads
that implemented the Print
Nightmare fix (Microsoft KB5005652).

Summary

THE PROBLEM: The "fix" (Microsoft KB5005652) to Print Nightmare removes a standard user's ability to add a printer.

This will likely result in an uptick of "help desk tickets" because 'admin rights' are now needed to map a printer.

THE CURRENT WORKAROUND: The KB article suggests disabling the new registry setting temporarily to allow users to add printers, but this creates a cybersecurity vulnerability.

A NEW SOLUTION: In a Windows Enterprise Network, System Administrators can package software in SCCM and make it available to users in Software Center. This step-by-step guide outlines a method of using Software Center to give users back this functionality while the Print Nightmare fix is enabled!

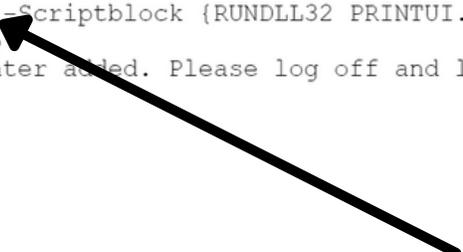
```

#Function for Button click
Function Action-Item {

    $Printername = ""
    $Printername = $textBox1.Text
    #Show Console and hide form
    $consolePtr = [Console.Window]::GetConsoleWindow()
    [Console.Window]::ShowWindow($consolePtr, 4)
    $Form.hide()
    Invoke-Command -Scriptblock {RUNDLL32 PRINTUI.DLL,PrintUIEntry /ga /n\\$Printername} -Verbose
    Start-sleep -s 5
    write-host "Printer added. Please log off and log back in to finish the install"

}

```



This script is built around this command line that adds a printer for all users of a computer. The simplicity should allow you to rewrite this work flow in the language of your choice!

```

#
[void][System.Reflection.Assembly]::Load('System.Drawing, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a')
[void][System.Reflection.Assembly]::Load('System.Windows.Forms, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089')
$Form = New-Object -TypeName System.Windows.Forms.Form
[System.Windows.Forms.Button]$button2 = $null
[System.Windows.Forms.TextBox]$textBox1 = $null
[System.Windows.Forms.ProgressBar]$progressBar1 = $null
[System.Windows.Forms.Label]$label1 = $null
[System.Windows.Forms.Button]$button1 = $null
[System.Windows.Forms.Button]$button2 = $null
$button2 = (New-Object -TypeName System.Windows.Forms.Button)
$textBox1 = (New-Object -TypeName System.Windows.Forms.TextBox)
$progressBar1 = (New-Object -TypeName System.Windows.Forms.ProgressBar)
$label1 = (New-Object -TypeName System.Windows.Forms.Label)
$Form.SuspendLayout()
#
#button2
#
$button2.BackColor = [System.Drawing.Color]::DarkOrange
$button2.Font = (New-Object -TypeName System.Drawing.Font -ArgumentList @(System.String)'Segoe UI',[System.Single]20,[System.Drawing.FontStyle]Normal,[System.Drawing.GraphicsUnit]Point)
$button2.ForeColor = [System.Drawing.SystemColors]::ButtonHighlight
$button2.Location = (New-Object -TypeName System.Drawing.Point -ArgumentList @(System.Int32)334,[System.Int32]333)
$button2.Name = [System.String]'button2'
$button2.Size = (New-Object -TypeName System.Drawing.Size -ArgumentList @(System.Int32)255,[System.Int32]133)
$button2.TabIndex = [System.Int32]3
$button2.Text = [System.String]'Add Printer'
$button2.UseVisualStyleBackColor = $false
$button2.add_Click([Action-Item])
#
#textBox1
#
$textBox1.ForeColor = [System.Drawing.SystemColors]::MenuHighlight
$textBox1.Location = (New-Object -TypeName System.Drawing.Point -ArgumentList @(System.Int32)44,[System.Int32]70)
$textBox1.Name = [System.String]'textBox1'
$textBox1.Size = (New-Object -TypeName System.Drawing.Size -ArgumentList @(System.Int32)212,[System.Int32]29)
$textBox1.TabIndex = [System.Int32]4
$textBox1.add_TextChanged($textBox1_TextChanged)
#
#label1
#
$label1.AutoSize = $true
$label1.ForeColor = [System.Drawing.SystemColors]::ButtonFace
$label1.ForeColor = [System.Drawing.SystemColors]::ButtonFace
$label1.Location = (New-Object -TypeName System.Drawing.Point -ArgumentList @(System.Int32)44,[System.Int32]70)
$label1.Name = [System.String]'label1'
$label1.Size = (New-Object -TypeName System.Drawing.Size -ArgumentList @(System.Int32)212,[System.Int32]29)
$label1.TabIndex = [System.Int32]7
$label1.Text = [System.String]'Enter Printer Name:'
#
#Form
#
$Form.BackColor = [System.Drawing.SystemColors]::ActiveCaptionText
$Form.ClientSize = (New-Object -TypeName System.Drawing.Size -ArgumentList @(System.Int32)255,[System.Int32]133)
$Form.Controls.Add($textBox1)
$Form.Controls.Add($button2)
$Form.Name = [System.String]'PowerShellFormProject3'
$Form.Resumelayout($false)
$Form.PerformLayout()
Add-Member -InputObject $Form -Name base -Value $base -MemberType NoteProperty
Add-Member -InputObject $Form -Name button2 -Value $button2 -MemberType NoteProperty
Add-Member -InputObject $Form -Name textBox1 -Value $textBox1 -MemberType NoteProperty
Add-Member -InputObject $Form -Name label1 -Value $label1 -MemberType NoteProperty
Add-Member -InputObject $Form -Name button1 -Value $button1 -MemberType NoteProperty
$Form.Add_Shown({$Form.Activate()})
[void] $Form.ShowDialog()

```

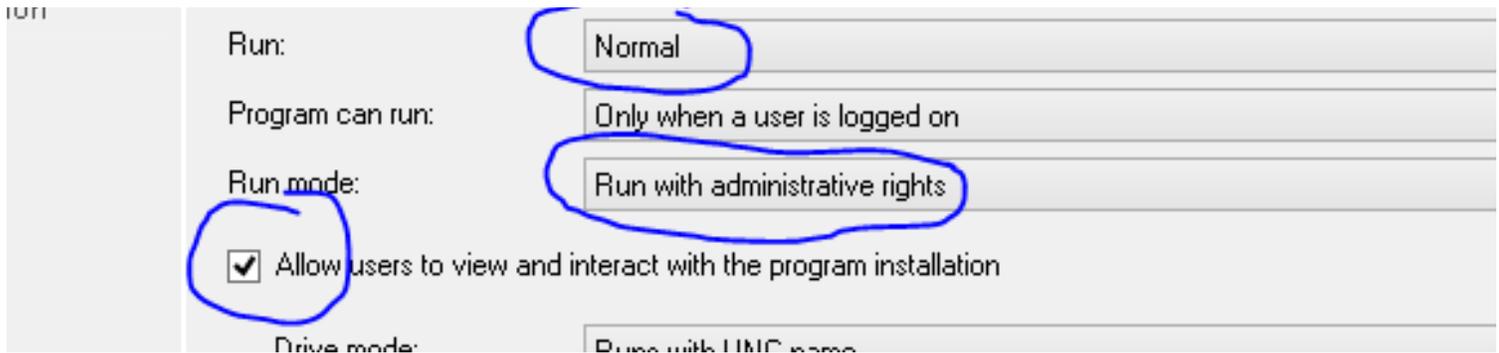
01

SCRIPT BREAKDOWN

Wrapping PowerShell inside of Winforms creates a better user experience than just using a "Read-host -Prompt". This script is set to hide the PowerShell console when the GUI is running--this allows for a simpler command line when running from SCCM/MECM.

```
Powershell.exe -executionpolicy Bypass -file .\install.ps1
```

Sample command line to run a .ps1



02

SCCM/MECM SETTINGS

- ***CREATE A PACKAGE, NOT AN APPLICATION***

If you do not know the difference, please do a little research. This will eliminate the need for a detection method and allows the user to run this app multiple times.

- ***SET THE PROGRAM TO RUN NORMAL, CHECK THE CHECK BOX FOR 'Allow users to view and interact with the program installation'***

This will allow the user to see and interact with the program.

- ***RUNS WITH ADMINISTRATIVE RIGHTS***

This will give the user the rights to add the printer since the registry key fix for PrintNightmare removed this ability from non-admin users.

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19042]
(c) Microsoft Corporation. All rights reserved.
C:\WINDOWS\system32>
```

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19042.1165]
(c) Microsoft Corporation. All rights reserved.
C:\WINDOWS\system32>psexec -s -i "powershell.exe"
```

```
Administrator: C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
S:
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\WINDOWS\system32> whoami
nt authority\system
PS C:\WINDOWS\system32>
```

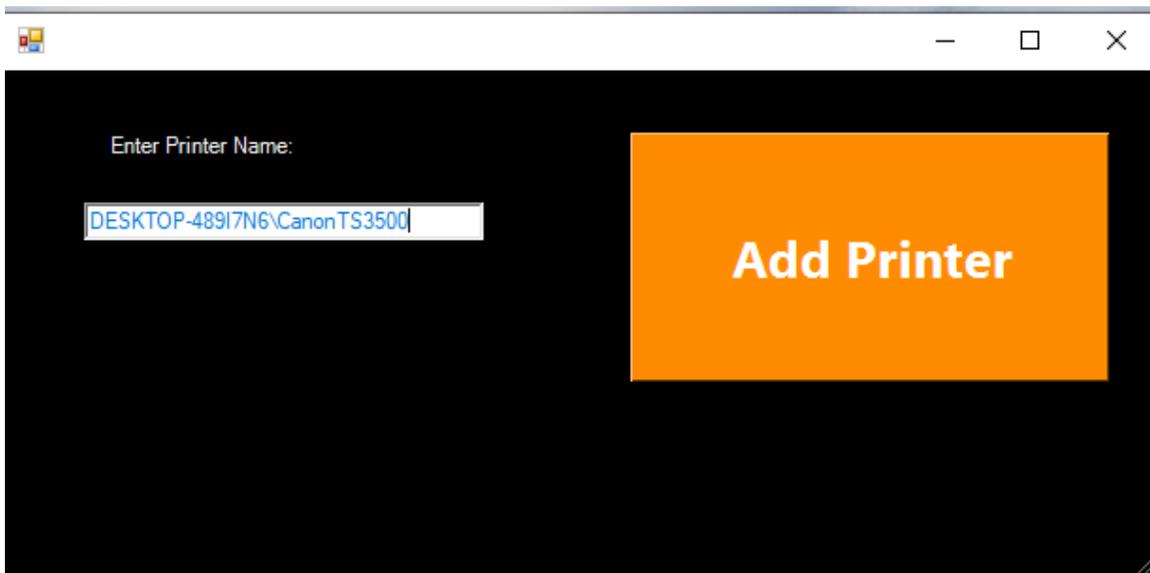
```
Administrator: C:\WINDOWS\System32\WindowsPowerSh
PS C:\temp> get-executionpolicy
Bypass
PS C:\temp>
```

03

PROOF OF CONCEPT

This is a method to test this script before placing it in SCCM/MECM. I have downloaded PSEXEC from the PSTOOLS, placed PSEXEC in my System32 folder, and ran an interactive command prompt in the SYSTEM context. SCCM/MECM uses the SYSTEM context to install as admin. ***NOTE:** To verify that I am running in SYSTEM, I ran a "WHOAMI" in the screenshot above. Also, the last screenshot displays a different directory. Make sure you are in the directory where the script is located AND don't forget to set your execution policy!

```
Administrator: C:\WINDOWS\System32\WindowsPowerShell
PS C:\temp> .\Testscript.ps1
```



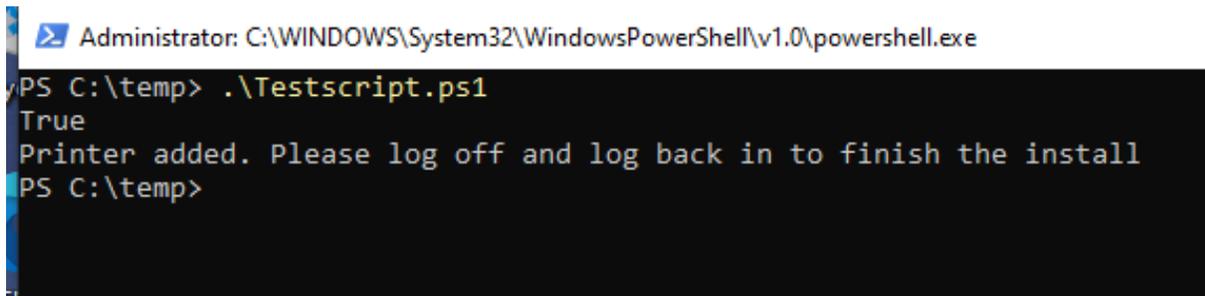
04

RUN THE APP

Run the Script.

Type the printer name in the textbox.

Click the Add Printer Button.



```
Administrator: C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe
PS C:\temp> .\Testscript.ps1
True
Printer added. Please log off and log back in to finish the install
PS C:\temp>
```

In this step, I brought the console back and used a write-host. This can easily be replaced with a message box from the .Net Framework.

05

WATCH THE MAGIC!

Once you see this screen, follow the instructions and log off. ***NOTE:** When placing this in your environment, feel free to force a restart if you want a fully automated solution--just make sure to warn the users!

Printers & scanners

Add printers & scanners

+ Add a printer or scanner

Printers & scanners

Fax

HP ePrint + JetAdvantage

Microsoft Print to PDF

Microsoft XPS Document Writer

OneNote for Windows 10

Add printers & scanners

+ Add a printer or scanner

Printers & scanners

Canon TS3500 series on DESKTOP-489I7N6

Fax

HP ePrint + JetAdvantage

Microsoft Print to PDF

Microsoft XPS Document Writer

OneNote for Windows 10

Before logoff/logon

After logoff/logon

06

LOG ON AND VIEW THE PRINTER!

When you log back in, navigate to "Printers and Scanners", and you will see the Printer! (Give it a moment--it may take a few minutes to finish up). This will add the printer for any user on the computer!

Developed By



Darius Peterson II

Information Technology Research and Development

Key Skills: PowerShell Scripting and Automation, Endpoint Management, Windows OS and Server Administration, Application Packaging (InstallShield, Windows Installer XML, and App Deployment Toolkit), C# (novice)