📘 Implementation Guide: First Login Notification with PowerShell & ServiceUI

🎯 Purpose

To display a **custom notification** to users upon their **first login** to a Windows device — even when the **ESP Account Setup phase is skipped** during Windows Autopilot deployment. This improves user experience and reduces confusion during initial configuration.

🧠 Why This Is Unique

Most Autopilot deployments that skip ESP leave users with no indication that configurations are still being applied. This solution fills that gap — and based on current research, no similar approach has been publicly documented.
It effectively extends the Windows First Login Notification, even when that feature is disabled

🛠️ Requirements

- ✅ PowerShell scripts (provided)
- ✅ ServiceUI.exe (from Microsoft Deployment Toolkit, provided)
- ✅ Banner image (Banner.jpg, dummy – replace from your own)

📁 Folder Structure

Place the following files on your device to convert to an Win32 package:
C:\Temp\

- Install.ps1
- uninstall.ps1
- ShowNotice.ps1
- Banner.jpg
- ServiceUI.exe

🧩 Script Functionality

- **Install.ps1**
    - Creates a **directory** on C:
    - **Copy files** (ShowNotice.ps1, Banner.jpg, ServiceUI.exe)
    - Create **Scheduled Task**
- **Shownotice.ps1** Places a **marker file** in the user's %APPDATA% (Roaming folder)
    - Detects the **logged-in user** via the explorer.exe process
    - Displays a **notification window**
    - Ensures the message is shown **only once per user** (marker file)
- **Uninstall.ps1** script to clean-up the configs

📂 Convert files to intunewin

Start the IntuneWinAppUtil.exe, can be found here:
https://github.com/microsoft/Microsoft-Win32-Content-Prep-Tool

Start converting the files.

See below screenshot for more information about converting the files to an IntuneWin File:

```
C:\Intune Apps\IntuneWinAppUtil.exe
Please specify the source folder: C:\Intune Apps\FirstLoginNotice - V1
Please specify the setup file: install.ps1
Please specify the output folder: C:\Intune Apps\FirstLoginNotice - V1
Do you want to specify catalog folder (Y/N)?n
```

You should now have the following files in the C:\Directory

Name
Banner.jpg
install.intunewin
install.ps1
ServiceUI.exe
ShowNotice.ps1
uninstall.ps1

🧩 Why We Use ServiceUI.exe in This Solution

Standard users do **not have administrative privileges**. Additionally, **Intune policies** can **block access to administrative tools and apps**, including PowerShell ISE, CMD, and other elevated interfaces.

However, this creates a challenge: how do we run a script that needs to interact with the **user session** (e.g., show a notification window), while the script itself is triggered by a **Scheduled Task running as SYSTEM**?

✅ The Solution: ServiceUI.exe

ServiceUI.exe is a utility from the **Microsoft Deployment Toolkit (MDT)** that allows a process running in the **SYSTEM context** to display a **user interface in the currently active user session**.

Use it to launch the PowerShell script that shows the **first login notification**.

- The task launches ServiceUI.exe, targeting the explorer.exe process (which always runs in the user session).
- ServiceUI.exe then launches powershell.exe with the script, allowing the notification window to appear **in the user's desktop environment**, even though the task itself runs with elevated privileges.

🔐 Why This Matters

- Users cannot bypass security restrictions or run PowerShell manually and the script runs with full SYSTEM permissions, but **interacts safely with the user**.

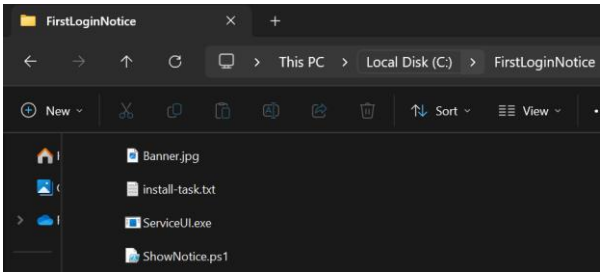
☁️ Upload to Intune

1. Go to **Microsoft Intune Admin Center**
2. Navigate to **Apps > Windows > Add**
3. Choose **App type: Windows app (Win32)**
4. Upload the .intunewin file
5. Configure the app:
    - **Install command**: powershell.exe -ExecutionPolicy Bypass -File .\install.ps1
    - **Uninstall command**: powershell.exe -ExecutionPolicy Bypass -File .\uninstall.ps1
    - **Install behavior**: System
    - **Requirements**: Select your requirement
    - **Device restart behavior:** No specific action
    - **Detection Rule type:** File
        - **Detection rule 1**:
            1. **Path**: C:\FirstLoginNotice
            2. **File or folder**: Banner.jpg
            3. **Detection method:** File or folder exists
            4. **Associated with a 32-bit app on 64-bit clients:** No
        - **Detection rule 2**:
            1. **Path**: C:\FirstLoginNotice
            2. **File or folder**: ServiceUI.exe
            3. **Detection method:** File or folder exists
            4. **Associated with a 32-bit app on 64-bit clients:** No
        - **Detection rule 3**:
            1. **Path**: C:\FirstLoginNotice
            2. **File or folder**: ShowNotice.ps1
            3. **Detection method:** File or folder exists
            4. **Associated with a 32-bit app on 64-bit clients:** No
6. **Assign** the app to a group of devices
7. **Test** and have fun informing your users!

🎯 Result

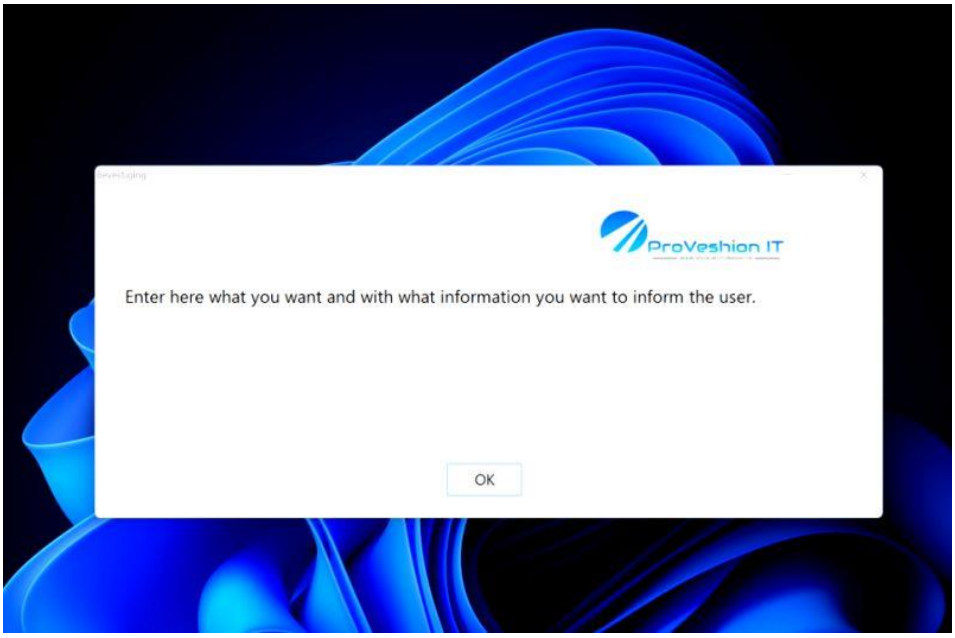
If the app is successfully deployed the device will get the following configuration:

- A new directory on **C:** named **FirstLoginNotice**
  - In this directory there are **four** files placed



  This is done via the **Install.ps1** script
- An scheduled task named **FirstLoginNotice** is created:

  FirstLoginNo... Ready At log on of any user
  - Task is configured to run with the **SYSTEM** account
  - At **log on** of any user
  - Start a program **C:\FirstLoginNotice\ServiceUI.exe**
    - Arguments
      -process:explorer.exe
      C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -ExecutionPolicy Bypass -File **C:\FirstLoginNotice\ShowNotice.ps1**
      This will run the notification script
- After logging in with a new user, the user gets the notification:



  If the user exits the notification the **markerfile** will be placed (Marker.txt)
  - Location: AppData\Roaming**FirstLoginNoticeShown**