Introduction

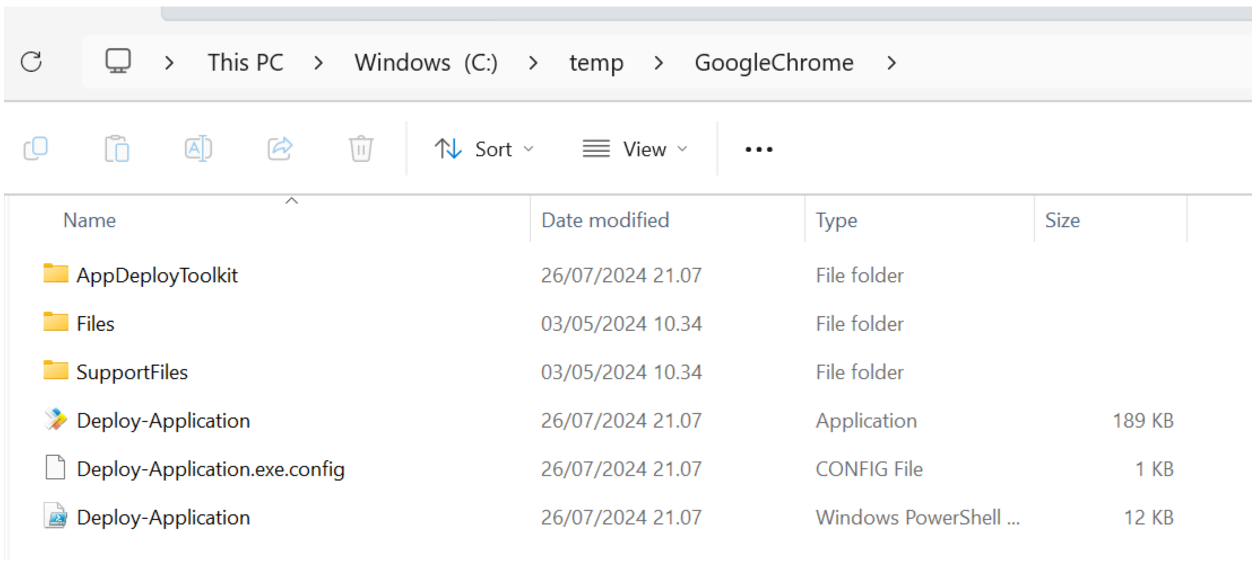
Application packaging in Intune is fun! It can be done in many different ways, but I prefer to use PSADT when I deploy apps.

If you are experienced in app deployment, you might have heard about PSADT (PSAppDeployToolkit) and even used it. In this blog post, we will take a look at deploying apps with PSADT.

Structure of the PSADT Folder

It's not a requirement to use PSADT in order to deploy Win32 apps; that shows how it can be done without.

1. The first step is to head to the PSADT website and click on download. This will redirect you to the Github page, where you can download the latest version.
2. When it's downloaded, I extract it, and normally I rename the toolkit folder to the application I will deploy. In this case, I will rename it to Google Chrome.



Name	Date modified	Type	Size
AppDeployToolkit	26/07/2024 21.07	File folder	
Files	03/05/2024 10.34	File folder	
SupportFiles	03/05/2024 10.34	File folder	
Deploy-Application	26/07/2024 21.07	Application	189 KB
Deploy-Application.exe.config	26/07/2024 21.07	CONFIG File	1 KB
Deploy-Application	26/07/2024 21.07	Windows PowerShell ...	12 KB

3. Now, we have a folder full of files, but what do we need to do with all those files? If you want a more in-depth explanation, I would recommend you read the official docs.

The most important thing you have to know as of now is:

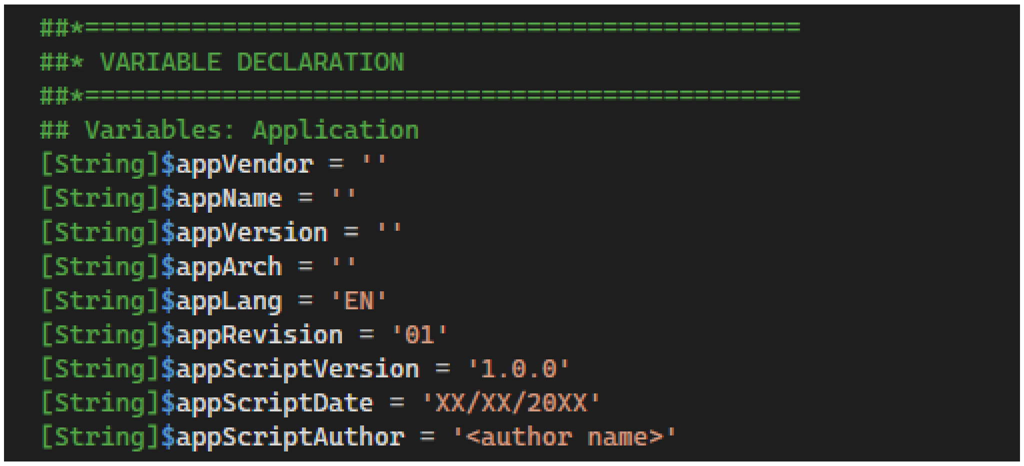
- **Files**
  - The files folder is where you will place your installation files; as an example, it could be the msi file for Google Chrome.
- **SupportFiles**
  - Next, the folder named supportfiles is where you will normally place any additional files for your deployment. After Chrome has been installed, we would like to copy bookmarks.
- **Deploy-Application.exe**

  - Deploy-Application.exe can be used when you are ready to deploy the application. Basically, the .exe file will call the .ps1 in order to perform the actions specified.
- **Deploy-Application.ps1**
  - The Deploy-Application.ps1 file is where you will spend the most of your time. This is where we will do the magic and specify what action we would like to take (e.g. installation or uninstallation)

Structure of the Deploy-Application.ps1

Open the Deploy-Application.ps1 file, and by first glance, you might think there is a LOT of information in here. What if I told you, this is actually pretty straight forward.

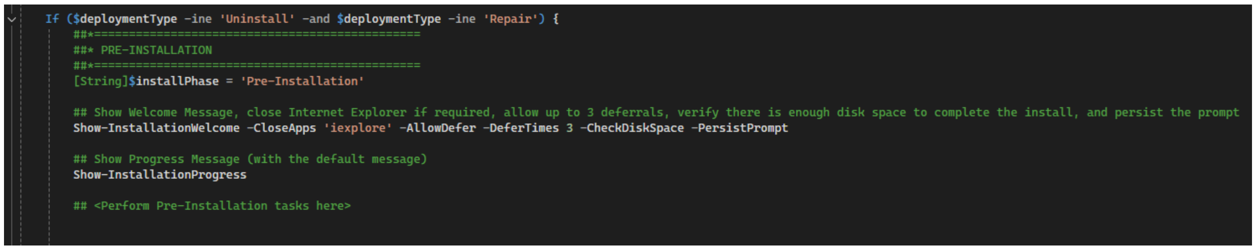
1. Scroll down until "variable declaration". Here you can input some useful information, such as application vendor, name, version, etc.



```
##*===============================================
##* VARIABLE DECLARATION
##*===============================================
## Variables: Application
[String]$appVendor = ''
[String]$appName = ''
[String]$appVersion = ''
[String]$appArch = ''
[String]$appLang = 'EN'
[String]$appRevision = '01'
[String]$appScriptVersion = '1.0.0'
[String]$appScriptDate = 'XX/XX/20XX'
[String]$appScriptAuthor = '<author name>'
```

Normally, I fill out the application vendor, name and version in order to help the next person looking at this package. Lastly, I will fill out the script date and script author.

2. From here, scroll down to where you can see pre-installation.

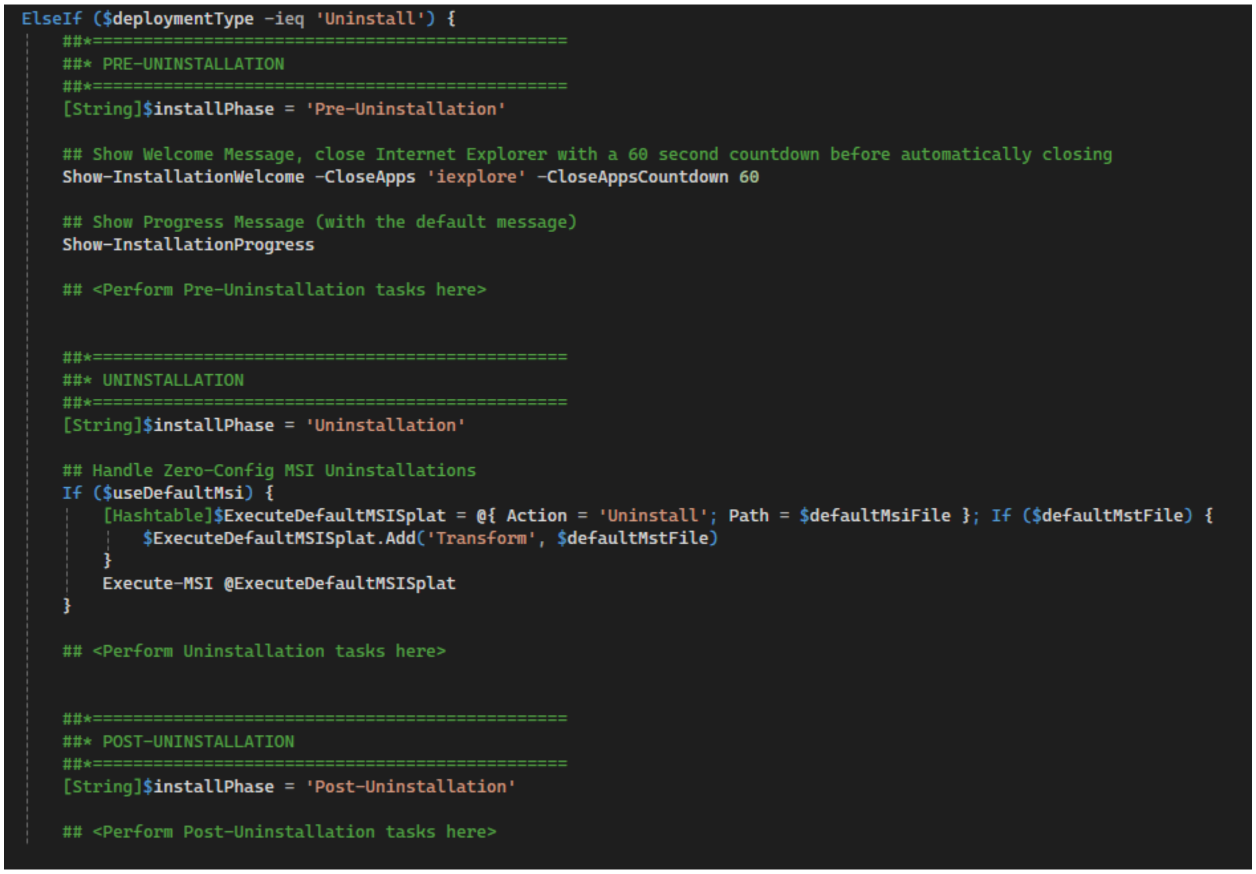


```
If ($deploymentType -ine 'Uninstall' -and $deploymentType -ine 'Repair') {
    ##*===============================================
    ##* PRE-INSTALLATION
    ##*===============================================
    [String]$installPhase = 'Pre-Installation'

    ## Show Welcome Message, close Internet Explorer if required, allow up to 3 deferrals, verify there is enough disk space to complete the install, and persist the prompt
    Show-InstallationWelcome -CloseApps 'iexplore' -AllowDefer -DeferTimes 3 -CheckDiskSpace -PersistPrompt

    ## Show Progress Message (with the default message)
    Show-InstallationProgress

    ## <Perform Pre-Installation tasks here>
```

If you scroll further down, you can see there are different sections such as pre-installation, installation, post-installation, and so on.

As an example, if you would like to remove older versions of Chrome before installing the newest, you can do that in the pre-installation section. If you would like to copy a shortcut of Chrome after installation, this can be done in the post-installation section.

For uninstallation, the same follows: pre-uninstallation, uninstallation, and post-uninstallation.



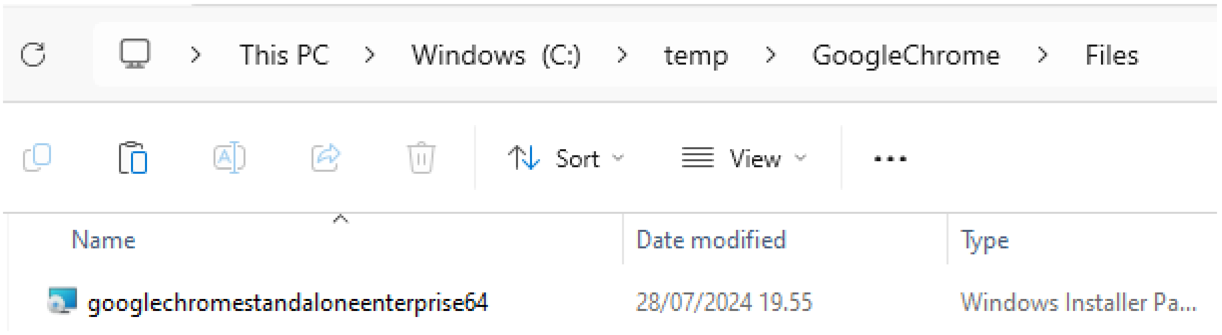
```
ElseIf ($deploymentType -ieq 'Uninstall') {
    ##*===============================================
    ##* PRE-UNINSTALLATION
    ##*===============================================
    [String]$installPhase = 'Pre-Uninstallation'

    ## Show Welcome Message, close Internet Explorer with a 60 second countdown before automatically closing
    Show-InstallationWelcome -CloseApps 'iexplore' -CloseAppsCountdown 60

    ## Show Progress Message (with the default message)
    Show-InstallationProgress

    ## <Perform Pre-Uninstallation tasks here>


    ##*===============================================
    ##* UNINSTALLATION
    ##*===============================================
    [String]$installPhase = 'Uninstallation'

    ## Handle Zero-Config MSI Uninstallations
    If ($useDefaultMsi) {
        [Hashtable]$ExecuteDefaultMSISplat = @{ Action = 'Uninstall'; Path = $defaultMsiFile }; If ($defaultMstFile) {
            $ExecuteDefaultMSISplat.Add('Transform', $defaultMstFile)
        }
        Execute-MSI @ExecuteDefaultMSISplat
    }

    ## <Perform Uninstallation tasks here>


    ##*===============================================
    ##* POST-UNINSTALLATION
    ##*===============================================
    [String]$installPhase = 'Post-Uninstallation'

    ## <Perform Post-Uninstallation tasks here>
```

Lastly, there are sections called repair. The same follows for those three sections.

Deploy Chrome with PSADT

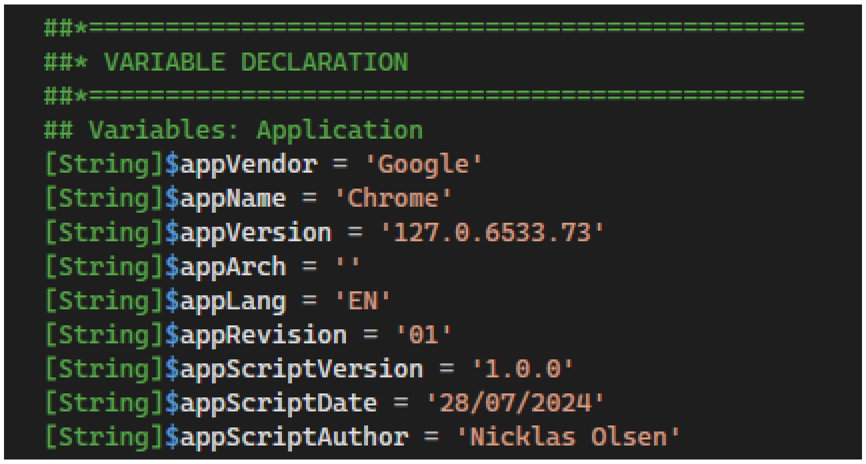
Now, we have taken a brief look at the folder and the Deploy-Application.ps1 structure of PSADT. Next, we need to do some actual work and prepare for our Google Chrome deployment.

1. First of all, we need to get the MSI file for Chrome. It can be downloaded from here. When that's downloaded, we will place our installation file in the files folder.



Name	Date modified	Type
googlechromestandaloneenterprise64	28/07/2024 19.55	Windows Installer Pa...

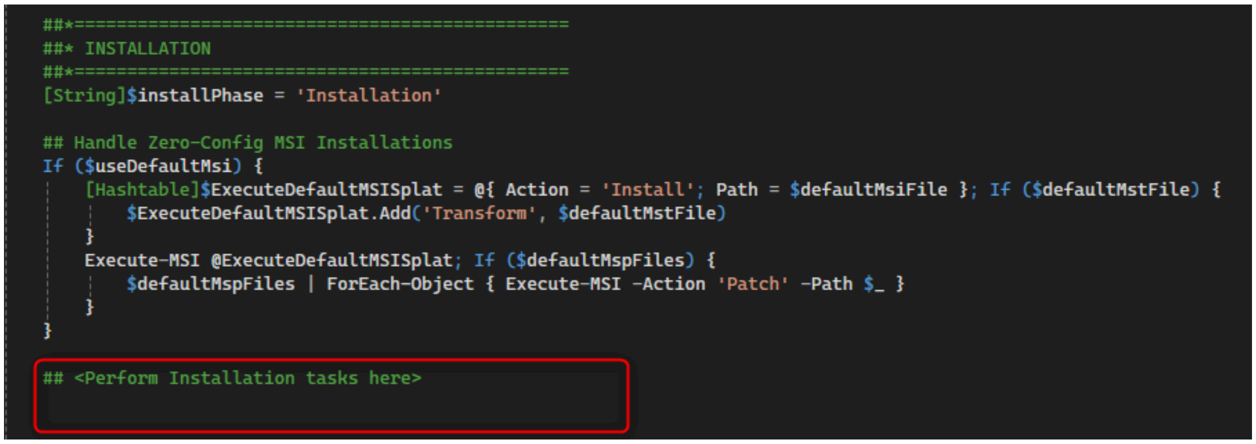
2. Head into Deploy-Application.ps1 and fill out the information regarding the application and the author. Here is an example of how I do it.



```
##*===============================================
##* VARIABLE DECLARATION
##*===============================================
## Variables: Application
[String]$appVendor = 'Google'
[String]$appName = 'Chrome'
[String]$appVersion = '127.0.6533.73'
[String]$appArch = ''
[String]$appLang = 'EN'
[String]$appRevision = '01'
[String]$appScriptVersion = '1.0.0'
[String]$appScriptDate = '28/07/2024'
[String]$appScriptAuthor = 'Nicklas Olsen'
```

3. Once that's done, scroll down to the installation section. This is where the magic will happen. You've probably noticed the

perform installation tasks here, that is where we will write our code.



```
##*===============================================
##* INSTALLATION
##*===============================================
[String]$installPhase = 'Installation'

## Handle Zero-Config MSI Installations
If ($useDefaultMsi) {
    [Hashtable]$ExecuteDefaultMSISplat = @{ Action = 'Install'; Path = $defaultMsiFile }; If ($defaultMstFile) {
        $ExecuteDefaultMSISplat.Add('Transform', $defaultMstFile)
    }
    Execute-MSI @ExecuteDefaultMSISplat; If ($defaultMspFiles) {
        $defaultMspFiles | ForEach-Object { Execute-MSI -Action 'Patch' -Path $_ }
    }
}

## <Perform Installation tasks here>
```

4. The team behind PSADT has created some really great functions to make it easy for us. In the docs, look for the Execute-MSI function.

We can re-use the example from the docs, so the installation task for Chrome can look like this:

```
## <Perform Installation tasks here>
Execute-MSI -Action 'Install' -Path 'googlechromestandaloneenterprise64.msi'
```

By default, the Execute-MSI will look in the files folder, so I can actually just write the name of the MSI.

5. Now, we have the installation tasks in order. But what about the uninstallation? We need that! Scroll down to the uninstallation section.

You've probably guessed it, but there is a function for this. It's called Remove-MSIApplications. It will look through the registry editor and see if there is a match on "Google Chrome".

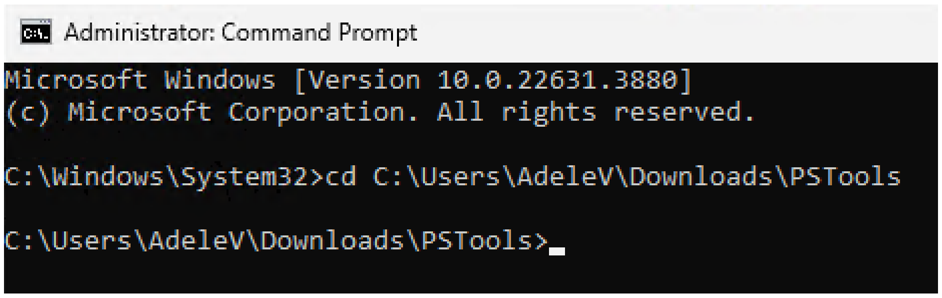
```
## <Perform Uninstallation tasks here>
Remove-MSIApplications -Name "Google Chrome"
```

6. Now, we are finished, and it's time for testing!

We need to make sure it works!

When deploying applications from Intune in SYSTEM context, it can sometimes cause some issues. Before I upload my application to Intune, I test it locally on a VM and make sure it works.

1. We need to grab and download PsExec, as we will be using this for testing. Make sure to extract the folder after it's finished downloading.
2. Open CMD as admin and change directory into where the PsExec file is located.

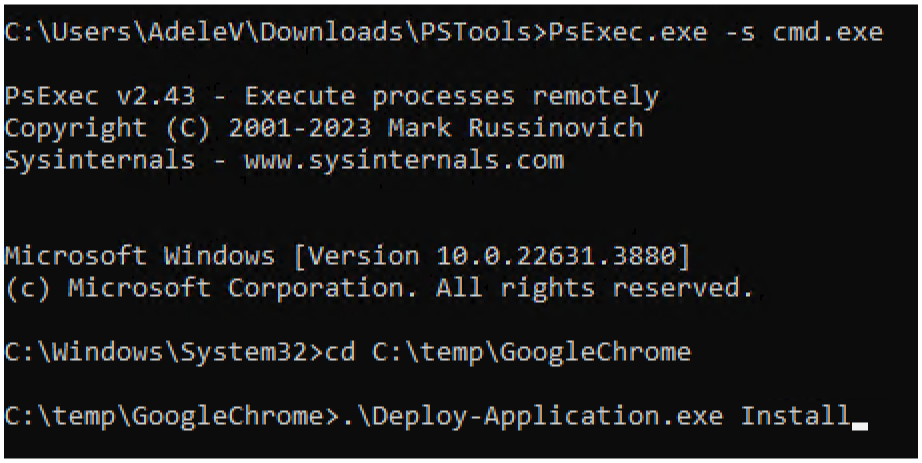


```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Users\AdeleV\Downloads\PSTools

C:\Users\AdeleV\Downloads\PSTools>
```

3. In order to kick off PsExec, I use the following command:

```
PsExec.exe -s cmd.exe
```

4. When it's finished loading, change directory into your PSADT folder and specify .\Deploy-Application.exe Install



```
C:\Users\AdeleV\Downloads\PSTools>PsExec.exe -s cmd.exe

PsExec v2.43 - Execute processes remotely
Copyright (C) 2001-2023 Mark Russinovich
Sysinternals - www.sysinternals.com


Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\temp\GoogleChrome

C:\temp\GoogleChrome>.\Deploy-Application.exe Install
```

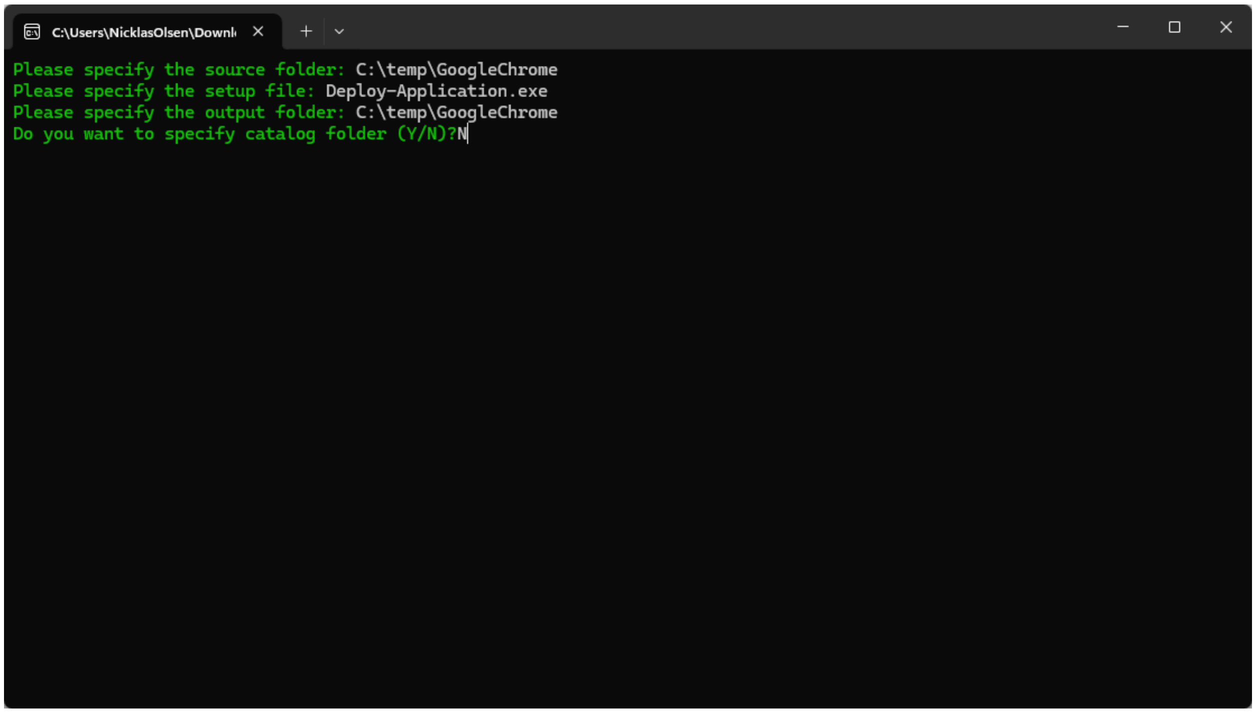
If the installation is successful, it will return "Exit 0" in the console.

5. Lastly, if you would like to test the uninstallation, just make sure to specify uninstall instead of install.

Upload the App to Intune

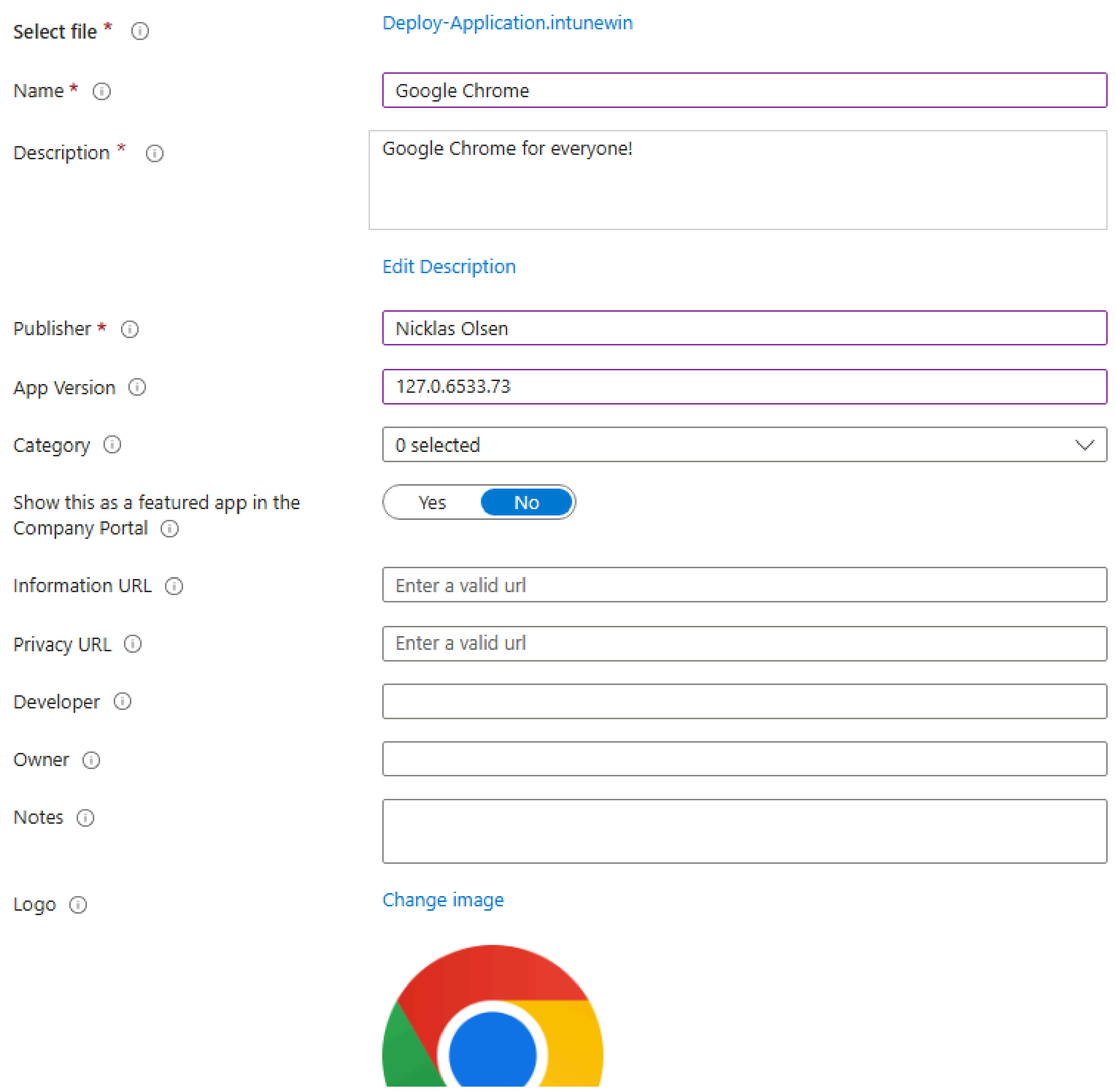
The app has been tested, and we are now ready to upload it to Intune. We will upload it as a Win32 app.

1. Wrap Deploy-Application.exe with the Win32 content prep tool.



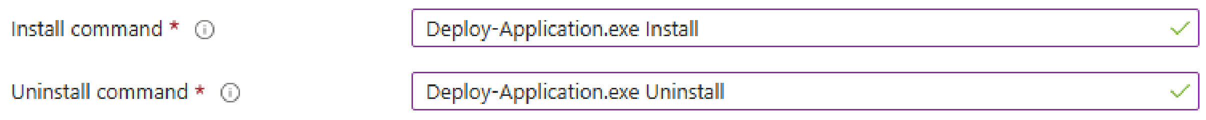
```
Please specify the source folder: C:\temp\GoogleChrome
Please specify the setup file: Deploy-Application.exe
Please specify the output folder: C:\temp\GoogleChrome
Do you want to specify catalog folder (Y/N)?N
```

2. Fill out the app information and continue to the next page.



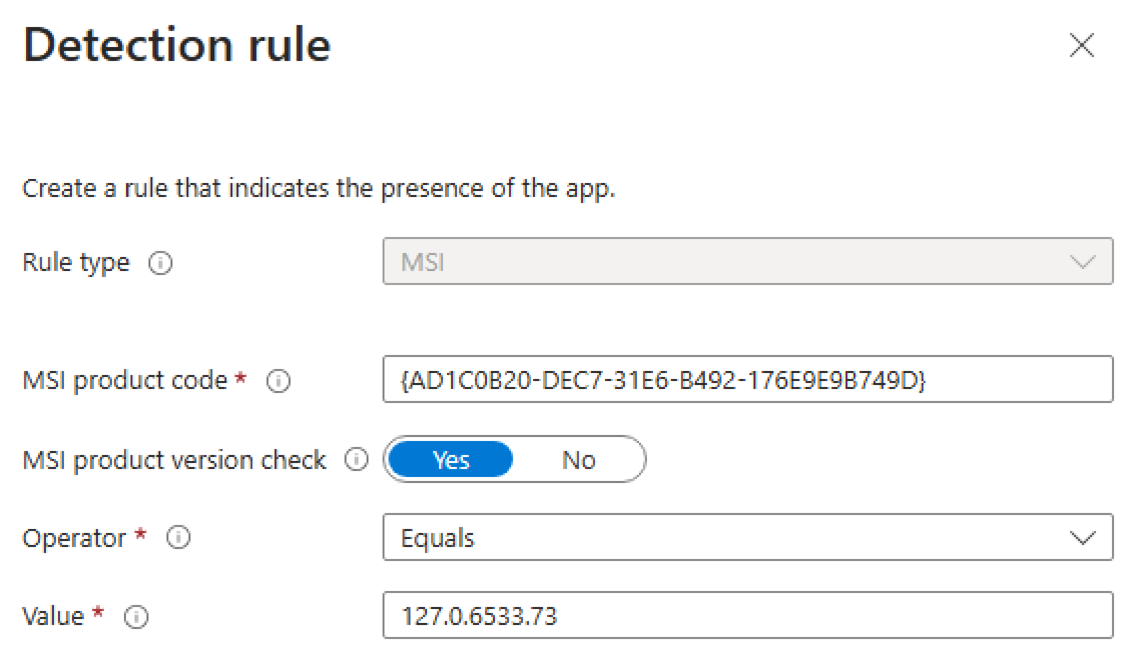
Select file *	Deploy-Application.intunewin
Name *	Google Chrome
Description *	Google Chrome for everyone!
	Edit Description
Publisher *	Nicklas Olsen
App Version	127.0.6533.73
Category	0 selected
Show this as a featured app in the Company Portal	Yes [✓ No]
Information URL	Enter a valid url
Privacy URL	Enter a valid url
Developer	
Owner	
Notes	
Logo	Change image

3. The installation and uninstallation commands are very easy. We will be calling the .exe file and the action.



Install command *	Deploy-Application.exe Install
Uninstall command *	Deploy-Application.exe Uninstall

4. I will not go into details with the requirements, so I will just jump to detection rules. For the detection rule, I will check the product code of the MSI file and verify if it equals the specific version.



Detection rule

Create a rule that indicates the presence of the app.

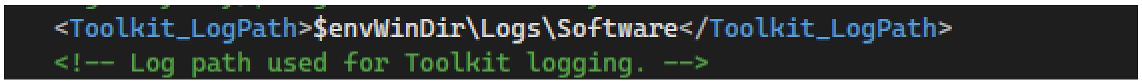
Rule type	MSI
MSI product code *	{AD1C0B20-DEC7-31E6-B492-176E9E9B749D}
MSI product version check	[✓ Yes] No
Operator *	Equals
Value *	127.0.6533.73

For this example, you can find the product code and the version in the following registry path: HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall

5. Lastly, I will make my assignments and create the application.

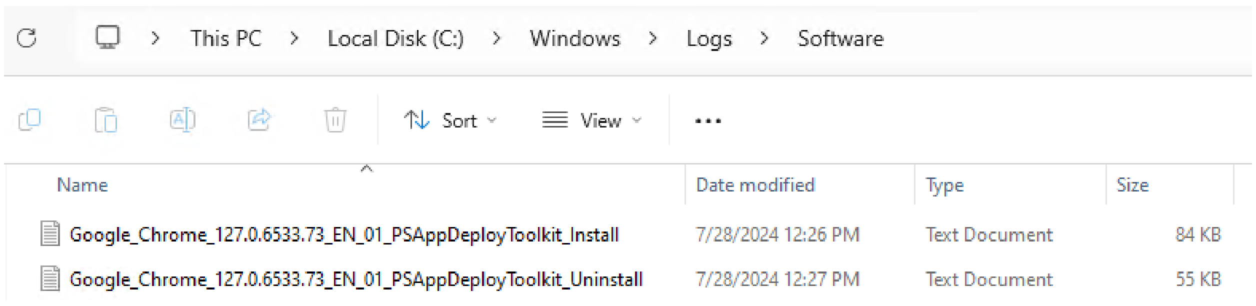
Logging

When you are deploying software with PSADT, the standard logpath will be in "C:\Windows\Logs\Software". This can be modified from the AppDeployToolkitConfig.xml file under the AppDeployToolkit folder.



```
    <Toolkit_LogPath>$envWinDir\Logs\Software</Toolkit_LogPath>
    <!-- Log path used for Toolkit logging. -->
```

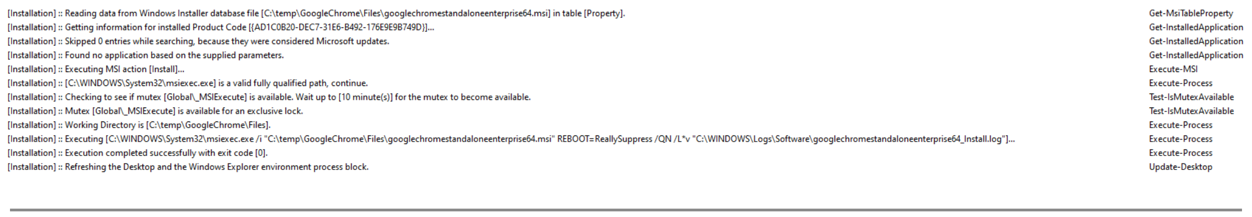
When you've performed an action, such as installing or uninstalling, a log file will be created.



Name	Date modified	Type	Size
Google_Chrome_127.0.6533.73_EN_01_PSAppDeployToolkit_Install	7/28/2024 12:26 PM	Text Document	84 KB
Google_Chrome_127.0.6533.73_EN_01_PSAppDeployToolkit_Uninstall	7/28/2024 12:27 PM	Text Document	55 KB

Always, when reading logs, I will recommend using CMTrace because it will make the logs much easier to read.

With a quick glance at the logs, we can see a lot of useful information. In this case, we would like to see what happened in the installation phase. Pretty good, right?



```
[Installation] :: Reading data from Windows Installer database file [C:\temp\GoogleChrome\Files\googlechromestandaloneenterprise64.msi] in table [Property].    Get-MsiTableProperty
[Installation] :: Getting information for installed Product Code [{AD1C0B20-DEC7-31E6-B492-176E9E9B749D}]...    Get-InstalledApplication
[Installation] :: Skipped 0 entries while searching, because they were considered Microsoft updates.    Get-InstalledApplication
[Installation] :: Found no application based on the supplied parameters.    Get-InstalledApplication
[Installation] :: Executing MSI action [Install]...    Execute-MSI
[Installation] :: [C:\WINDOWS\System32\msiexec.exe] is a valid fully qualified path, continue.    Execute-Process
[Installation] :: Checking to see if mutex [Global\_MSIExecute] is available. Wait up to [10 minute(s)] for the mutex to become available.    Test-IsMutexAvailable
[Installation] :: Mutex [Global\_MSIExecute] is available for an exclusive lock.    Test-IsMutexAvailable
[Installation] :: Working Directory is [C:\temp\GoogleChrome\Files].    Execute-Process
[Installation] :: Executing [C:\WINDOWS\System32\msiexec.exe /i "C:\temp\GoogleChrome\Files\googlechromestandaloneenterprise64.msi" REBOOT=ReallySuppress /QN /L*v "C:\WINDOWS\Logs\Software\googlechromestandaloneenterprise64_Install.log"]...    Execute-Process
[Installation] :: Execution completed successfully with exit code [0].    Execute-Process
[Installation] :: Refreshing the Desktop and the Windows Explorer environment process block.    Update-Desktop
```

Conclusion

If you haven't used PSADT before, I would recommend you take a look at it. Thanks for reading this blog, I hope it was useful for you!