## Practice Activity 6

Name: Sajiya

Mail id: [sajiyadl888@gmail.com](mailto:sajiyadl888@gmail.com)

User id: 34764

**ASSIGNMENT**

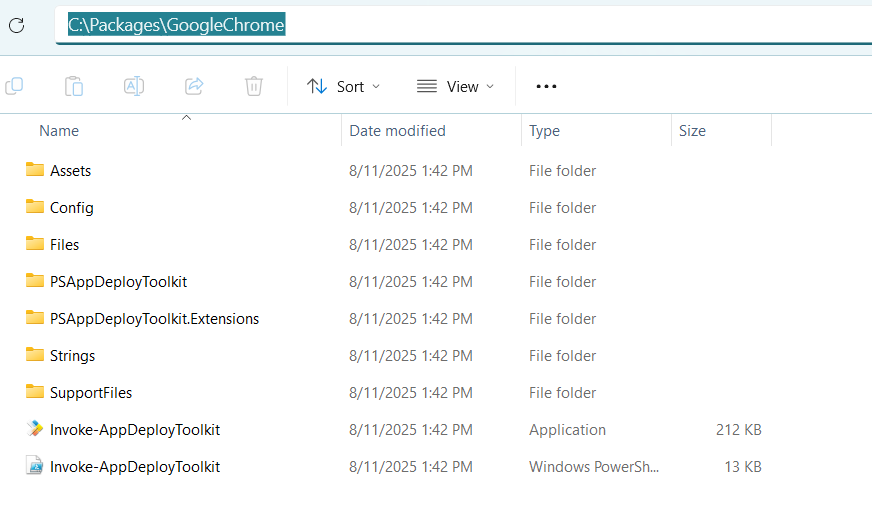
**Topic:** (i) Google Chrome deployment with PSADT

**Step 1: Download PSADT**

Link here [PSADT\_ZIP](https://github.com/PSAppDeployToolkit/PSAppDeployToolkit/releases/tag/4.1.0) (It is link of github)

Download lastest version from assests (PSAppDeployToolkit\_Template\_v4.zip)

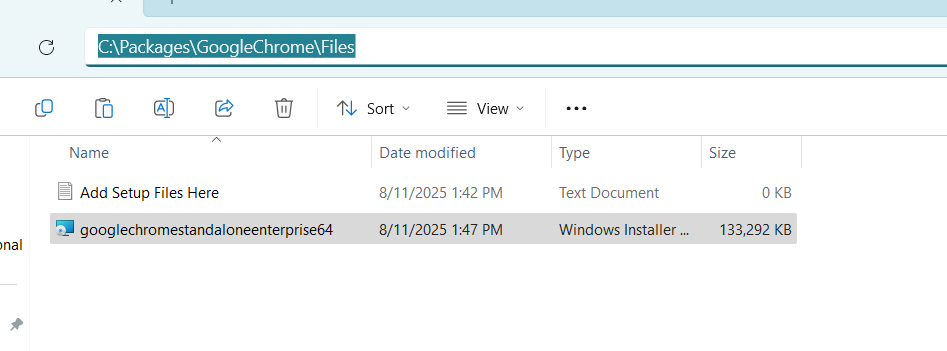
Extract the ZIP to a working folder (e.g., C:\Packages\GoogleChrome).



**Step 2: Download Chrome msi**

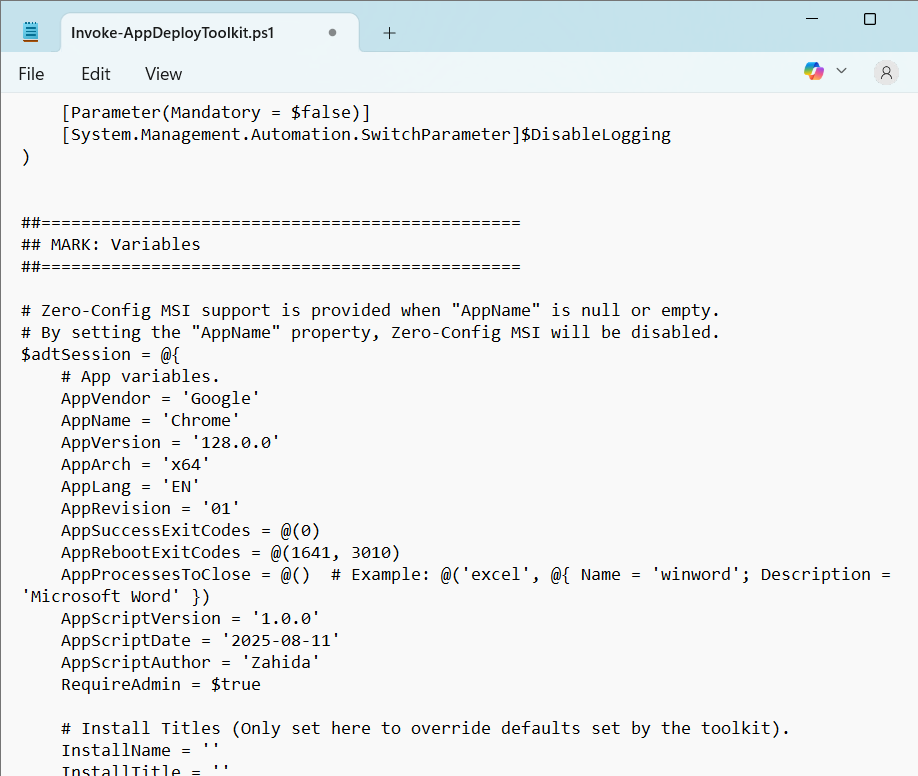
Link here [Chrome.MSI](https://chromeenterprise.google/download/)

Place the downloaded msi file into… ‘C:\Packages\ChromePSADT\Files’



**Step 3: Edit Invoke-AppDeploymentToolkit.ps1**

1. Scroll to Variables and edit as shown in screenshot

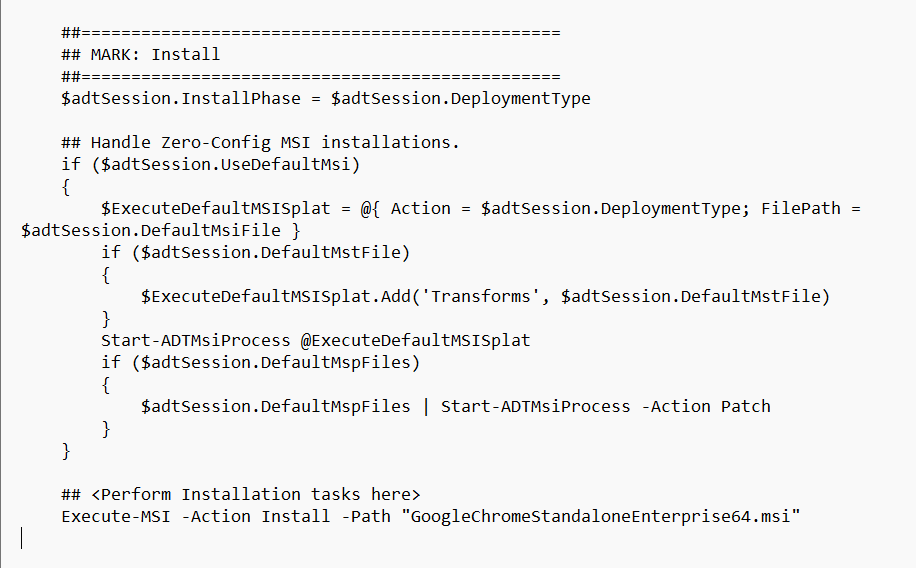


1. Scroll to installation section

Replace: # <Perform Installation tasks here>

With

Execute-MSI -Action Install -Path "GoogleChromeStandaloneEnterprise64.msi"

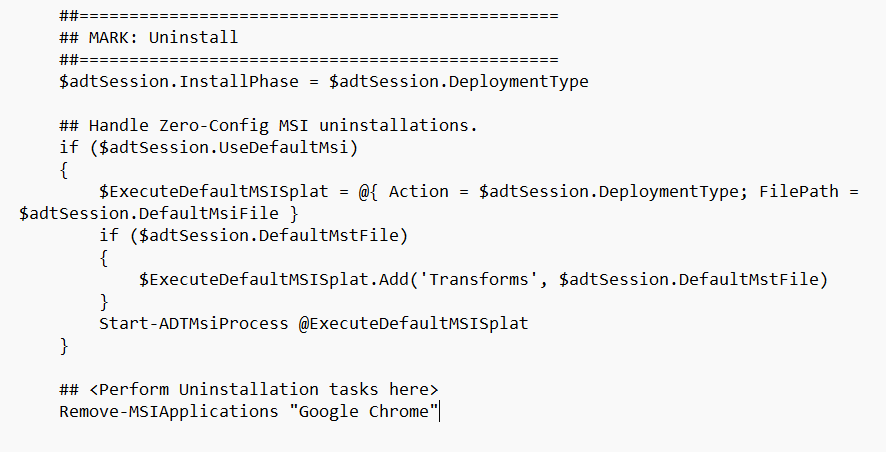


1. Scroll to uninstallation section

Replace: # <Perform Uninstallation tasks here>

With

Remove-MSIApplications "Google Chrome"



Save the file.

**Step 4: Test locally before uploading it to Intune**

1. Download PsExec

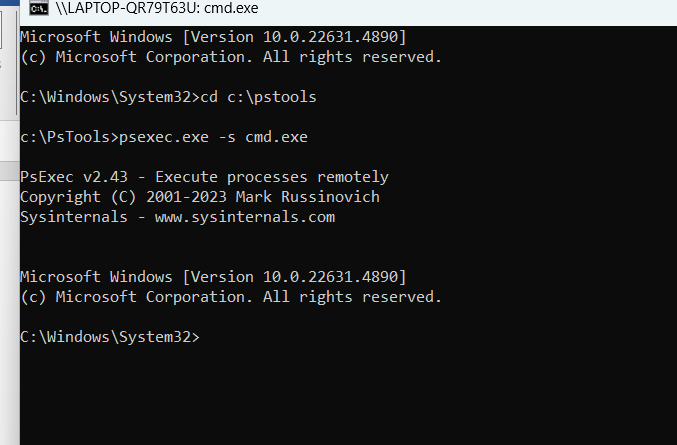
Link of [PsTools](https://learn.microsoft.com/en-us/sysinternals/downloads/psexec) (will download all PsTools)

Extract it to C:\PsTools.

1. Open cmd and run in system context

cd C:\PsTools

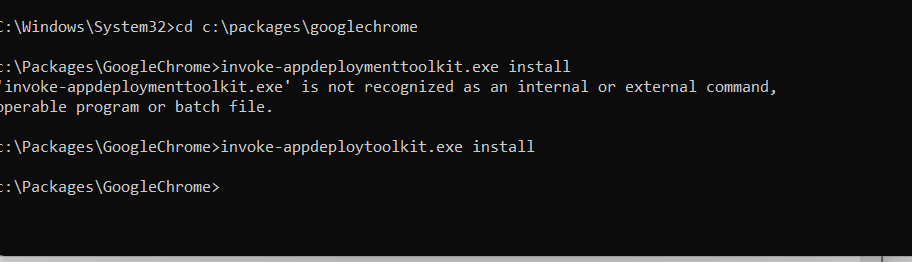
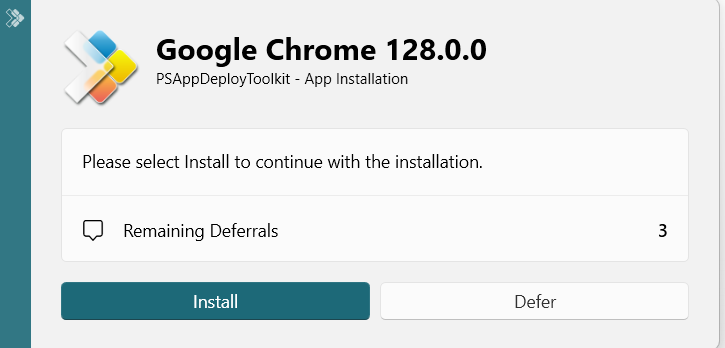
PsExec.exe -s cmd.exe

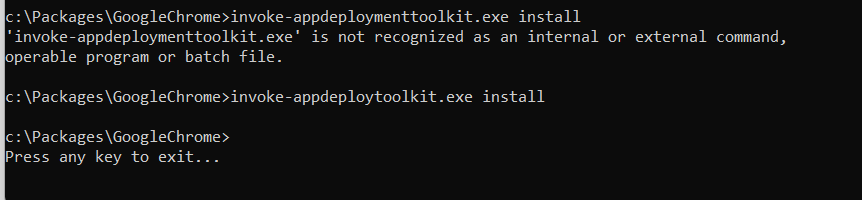


1. Install Chrome

cd C:\Packages\GoogleChrome

Deploy-Application.exe Install



1. Uninstall Chrome

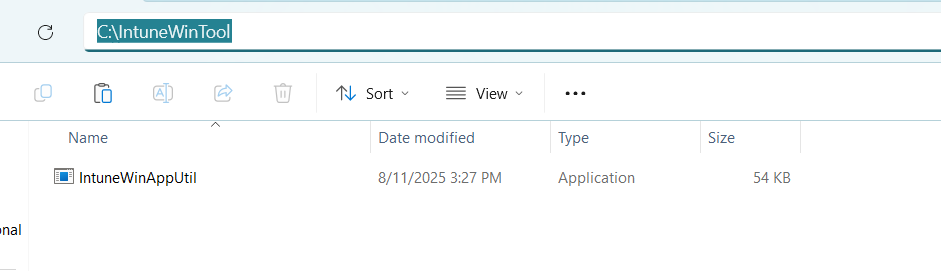
Invoke-AppDeployToolkit.exe Uninstall

**Step: 5 Prepare for Intune**

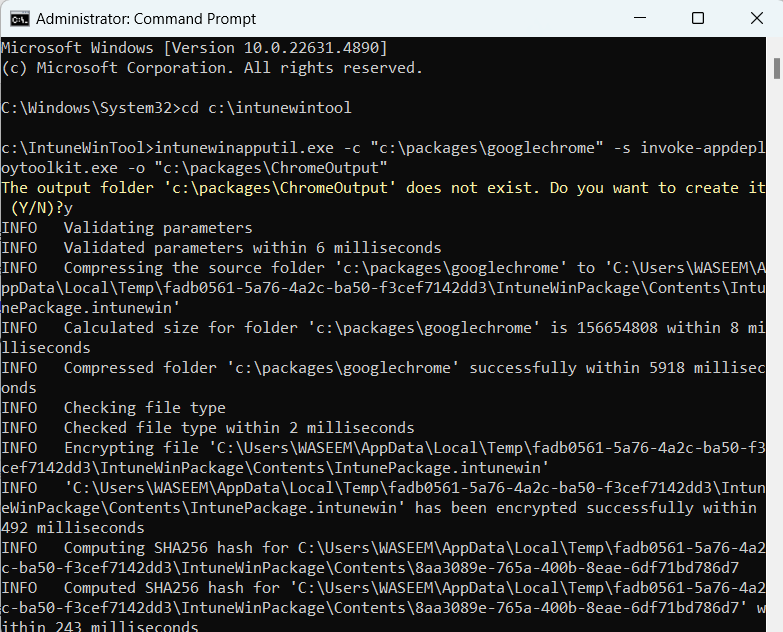
1. Download win32 content prep tool

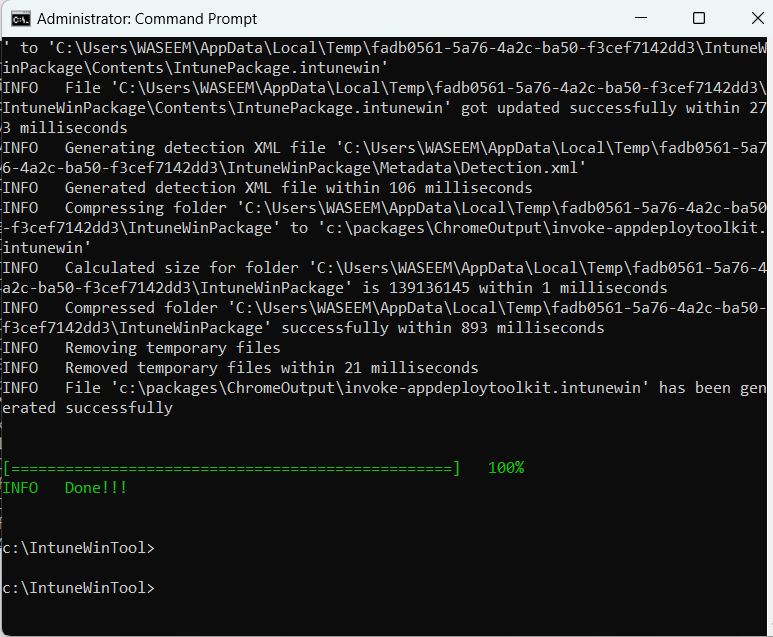
Link [Win32ContentPrepTool\_ZIP](https://github.com/microsoft/Microsoft-Win32-Content-Prep-Tool/releases/tag/v1.8.6)

Extract to C:\IntuneWinTool.



1. Wrap the App





**Step 6: Upload to Intune**

Go to **Intune Admin Center → Apps → Windows apps → Add → Win32 App**.

Select your .intunewin file.

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

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

**Detection Rule**:

* + MSI product code from:

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall

Assign to required device group.

**Step 7: Logs**

After install/uninstall:

makefile

C:\Windows\Logs\Software

Open log with **CMTrace** for easy reading.

**Topic:** (ii) Summary (Template Script in PSADT)

1. Understanding PSAppDeployToolkit and Templates:  
2. Using the Template Script (Deploy-Application.ps1):  
3. Creating a New Deployment Using Templates:  
4. Additional Resources:  
5. Customizing the Template Script:

The **PowerShell Application Deployment Toolkit (PSAppDeployToolkit)** streamlines enterprise app deployments using customizable template scripts like Deploy-Application.ps1. These templates contain predefined functions, UI elements, and a consistent structure for both installs and uninstalls, split into **Pre-Install**, **Install**, and **Post-Install** phases.

You can create new deployments from built-in or downloadable templates (v3 or v4) using the New-ADTTemplate command or ZIP files from GitHub. The Deploy-Application.ps1 script is edited to match app-specific requirements—adjusting parameters, adding commands, or modifying the UI—making it a foundation for both simple and complex deployment scenarios.

**Topic:** (iii) Summary (MSI/MSP logging in PSADT)

In **PSAppDeployToolkit (PSADT)**, we use the Execute-MSI and Execute-MSP functions to install, patch, or uninstall MSI and MSP files. These functions also let us set up logging, which helps track what happens during deployment and makes troubleshooting easier.

We can customize logging using these main options:

* **LogName**: Name of the log file (adds .log if missing).
* **LogPath**: Folder where the log file will be saved. If not set, logs go to the default PSADT log folder.
* **LogVerbosity**: How detailed the log is (like verbose, info, or errors only).

For example:

* To install with logging:

powershell

Execute-MSI -Action Install -Path "C:\MyApps\MyPackage.msi" -LogName "MyPackageInstallLog" -LogVerbosity Verbose

* To patch:

powershell

Execute-MSI -Action Patch -Path "C:\MyApps\MyPatch.msp" -LogName "MyPatchLog"

* To uninstall:

powershell

Execute-MSI -Action Uninstall -Path "[PRODUCTCODE]" -LogName "MyPackageUninstallLog"

You can change where logs are saved by setting $configToolkitLogDir in the AppDeployToolkitConfig.xml file or inside the script itself.

It’s important to pick a good log location (maybe a shared folder) and choose the right log level — verbose logs everything, error logs only failures. You can also decide if deployment continues or stops on errors.

Using these settings, PSADT helps keep good records of MSI/MSP installs, which makes fixing problems and monitoring deployments easier.

**Topic:** (iv) Summary (Commands for installation and uninstallation in PSADT)

For installation

Execute-MSI -Action Install -Path "C:\path\to\your.msi" -DeployMode Silent

For uninstallation

Execute-MSI -Action Uninstall -Path "{PRODUCT-GUID}"

**Other helpful commands**:

* Get-RunningProcesses to find processes that need to close before install/uninstall.
* Show-WelcomePrompt to display messages to users.
* Remove-MSIApplications, Unblock-ADTAppExecution, and Unregister-ADTDll for other deployment tasks.

**Topic:** (v) Summary (Tool configuration in PSADT)

**PowerShell App Deployment Toolkit (PSAppDeployToolkit)** is a framework that makes deploying applications easier and customizable. You control many settings by editing the **AppDeployToolkitConfig.xml** file, which lets you set things like default log locations, UI banners, icons, and logging options.

**How to configure it:**

1. Download and unzip PSAppDeployToolkit.
2. Find the AppDeployToolkitConfig.xml in the Toolkit folder.
3. Open and edit this XML file to customize:
   * General toolkit settings (admin rights, temp folders, logging).
   * UI elements like banners, logos, and icons.
   * MSI install options and logging paths.
   * UI behavior like notifications and timeouts.
   * Language messages for the interface.
4. Save the file — changes apply to all PSADT scripts.

**Benefits:**

* Centralized settings for all deployments.
* Consistent look and behavior across apps.
* Easier customization and less repetitive coding.