BATCH INFORMATION:

* Name: Sohini Sri Lenka
* User ID: 34747
* Batch name: WiproNGA\_DWS\_B5\_25VID2550
* Batch ID: 25VID2550
* Assignment date: 12/8/25

OVERVIEW OF THE TOOL PSADT:

PSADT is a ready-made PowerShell tool that helps you install or remove software on Windows computers in a neat and professional way.  
It’s like a smart assistant for IT admins that does all the hard parts of software deployment.

**Key Features (What it can do)**

* Install or uninstall apps easily (EXE, MSI, etc.).
* Show pop-up messages to the user (e.g., “Please save your work”).
* Close running apps before installing.
* Let users delay the install if they are busy.
* Show a progress bar during the install.
* Log everything so you know what happened.
* Check system requirements before starting.

**3. Benefits (Why it’s useful)**

* Saves time — no need to write big scripts from scratch.
* Looks professional — It clean messages, progress bars, same style for all apps.
* Avoids problems — It checks for space, OS, running apps before installing.
* User-friendly — It doesn’t surprise the user, gives them control.
* Works with many tools — Intune, SCCM, or standalone.

**4. How it works (Simple steps)**

1. Download PSADT and put your app installer in its “Files” folder.
2. Edit one script (Deploy-Application.ps1) to tell it what to install.
3. Run it (manually or via Intune/SCCM).
4. PSADT will show messages, check things, close apps, install the software, and log everything.
5. You get a clear success or failure result.

FOLDER STRUCTURE IN PSADT:

The PowerShell App Deployment Toolkit (PSADT) uses a structured folder hierarchy to organize deployment packages and scripts. This structure allows for easy management and customization of deployments.

**Key folders and their purpose:**

**Toolkit/**: This folder contains the core PSADT module files, including the manifest, script, and language strings.

* **Examples/**: This folder holds example deployment scripts for various applications, serving as templates for new deployments. [1, 1]
* **Files/**: This is where you place the application's installation files, such as .msi, .exe, or .appx files.
* **SupportFiles/**: This folder is used to store supporting files required for the deployment, like configuration files, scripts, or certificates.
* **Deploy-Application.ps1**: This is the main PowerShell script that orchestrates the deployment process. It's where you customize the script for your specific application.
* **Deploy-Application.exe**: This is a wrapper for the Deploy-Application.ps1 script. It ensures that the correct PowerShell execution policy is set.

**Customization and Version Control:**

* **Branding:** You can customize the PSADT with branding elements like banners and icons.
* **Templates:** You can create a template deployment folder that can be reused for future deployments.
* **Version Control:** It's recommended to use a version control system (like Git) to track changes to your deployment scripts and configurations.

**Key Considerations:**

* **Modular Structure:** PSADT v4 introduces a more modular structure with dedicated folders for different types of files and settings, making it easier to manage and customize.
* **File and Folder Structure:** The deployment structure is designed to be flexible and user-friendly, allowing for customization and reuse.
* **Customization:** You can customize the PSADT's UI, branding, and deployment scripts to suit your specific needs.

 TOOL CONFIGURATION IN PSADT:

The PowerShell App Deployment Toolkit (PSAppDeployToolkit) is a scripting framework that simplifies application deployment. It allows for customization and control over the deployment process, including pre-installation checks, UI customization, and logging options

**Configuration Process:**

1. **Download and Extract:** Download the PSAppDeployToolkit and extract the zip file.
2. **Locate Configuration File:** Navigate to the Toolkit\AppDeployToolkit directory and find the AppDeployToolkitConfig.xml file.
3. **Edit Configuration:** Open the AppDeployToolkitConfig.xml file and customize it according to your needs.

* **Toolkit Options:** Configure general toolkit settings like required administrative rights, temporary file paths, and logging options.
* **Banner, Logo & Icon Options:** Customize the UI banner, logos, and icons.
* **MSI Options:** Configure logging options, log paths, and installation parameters for MSI installations.
* **UI Options:** Set general UI options like balloon notifications, script timeouts, and exit codes.
* **UI Messages:** Edit or add languages for the user interface.

1. **Save and Apply:** Save the changes to the AppDeployToolkitConfig.xml file. These changes will be applied to all scripts created using PSAppDeployToolkit.

**Benefits of Using AppDeployToolkitConfig.xml:**

Centralized Configuration, Consistency , Customization , Simplified Scripting.

**Understanding PSAppDeployToolkit and Templates:**

* PSADT is a ready-to-use PowerShell toolkit for installing, updating, or removing software.
* It comes with template scripts pre-written code with placeholders.
* You just fill in your app details instead of writing everything from scratch.

**2. Using the Template Script (Deploy-Application.ps1)**

* The file Deploy-Application.ps1 is your main script.
* It already has sections for:
  + Pre-Install (things to do before installing, like closing apps)
  + Install (commands to install your app)
  + Post-Install (things to do after installation, like showing a message)
* You simply uncomment and edit the lines to match your app.

**3. Creating a New Deployment Using Templates**

1. Copy the PSADT folder as a new project folder.
2. Put your installer in the Files folder.
3. Edit the Deploy-Application.ps1 file to add your install/uninstall commands.
4. Test it on a sample machine.
5. Deploy using Intune, SCCM, or manually.

**5. Customizing the Template Script**

* You can change pop-up messages, add your company logo, or modify the colors.
* You can add extra functions to run tasks before/after install.
* You can adjust system checks (like OS version or free disk space).
* You can also disable or enable deferral options for the user

MSI/MSP LOGGING IN PSADT:

In PowerShell App Deployment Toolkit (PSADT), MSI (Windows Installer) and MSP (patch) file logging can be configured through various parameters within the Execute-MSI and Execute-MSP functions. These functions handle the complexities of MSI/MSP deployment, including logging, error handling, and user interaction.

Here's a breakdown of how to enable and configure MSI/MSP logging in PSADT:

**1. Logging Options:**

* -LogName: Specifies the name of the log file to be created. If the name does not end with .log, it will be automatically appended. [4, 4]
* -LogPath: Specifies the path where the log file will be created. If not specified, it defaults to the standard PSADT log directory. [4, 4, 5, 5, 6, 7]
* -LogVerbosity: Controls the level of detail in the log file (e.g., verbose, informational, error). [4, 4, 8, 8, 9]

**2. Using Execute-MSI for MSI Installation:**

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

**3. Using Execute-MSI for MSP Patching:**

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

**4. Using Execute-MSI for MSI Uninstallation:**

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

**5. Customizing Log File Location:**

* Configuring $configToolkitLogDir: You can set the $configToolkitLogDir variable in the AppDeployToolkitConfig.xml file to specify a custom location for all logs.
* Modifying Deploy-Application.ps1: You can also set $configToolkitLogDir early in the Deploy-Application.ps1 script, or modify AppDeployToolkitMain.ps1 to set the variable.

**6. Important Considerations:**

* **Log File Location:** The default log directory is typically under the PSADT working directory, so you may want to configure a more suitable location, such as a network share or a dedicated log folder.
* **Log Level:** Use -LogVerbosity to control the amount of detail in the log. Verbose provides extensive information, while Error only logs errors.
* **Error Handling:** The Execute-MSI and Execute-MSP functions can be configured to continue on errors (e.g., -ContinueOnError $true) or fail on errors (e.g., -ContinueOnError $false)

 By using these parameters and techniques, you can effectively log MSI and MSP installations and uninstallations within PSADT, enabling you to troubleshoot issues and monitor the deployment process.