**BATCH ID: WiproNGA\_DWS\_B5\_25VID2550**

**NAME:** Shrawani Shyam Balwadkar

**RPS USER ID:** 34932

**DATE**: 11-08-2025

1. **Application Deployment using PowerShell App Deployment Toolkit (PSADT)**

The **PowerShell App Deployment Toolkit (PSADT)** is an open-source framework designed to simplify the packaging and deployment of applications in enterprise environments. It provides a set of predefined functions, a consistent folder structure, and a template script to handle installation, uninstallation, upgrades, and custom actions in a reliable and repeatable way.

**Key Features:**

* **Simplified Scripting** – Pre-built PowerShell functions reduce the amount of custom code required.
* **Consistent Structure** – Standardized folder and script format across all applications.
* **Silent Deployments** – Installs or uninstalls applications without user interaction.
* **Robust Logging** – Automatic logging of every action for troubleshooting and auditing.
* **Error Handling** – Built-in prompts, rollback options, and safe process termination.

**Common Use Cases:**

* Enterprise software rollouts.
* Mass application upgrades.
* Automated uninstallation of outdated software.
* Pre-installation checks and post-installation configurations.

**Advantages over Traditional Scripting:**

* Reduces complexity by providing **Execute-MSI** and **Execute-Process** functions for common installer types.
* Eliminates the need to write repetitive code for logging, checking preconditions, and creating user prompts.
* Works seamlessly with **Microsoft Intune, SCCM, MDT**, and other deployment systems.

**Example:** Installing Microsoft Office 365 silently:

Execute-Process -Path "$dirFiles\setup.exe" -Parameters '/configure Configuration.xml'

**2. Understand Basic Platform Value, Concepts and Usage**

The deployment platform is designed to:

* **Standardize** the process of installing, upgrading, or removing applications.
* **Automate** actions, reducing manual effort.
* **Ensure Reliability** through consistent scripts and error handling.
* **Scale Easily** for large environments with minimal script adjustments.

**Core Concepts:**

* **Template-Based Deployment** – Using a predefined structure for all apps.
* **Silent Execution** – No user prompts during install/uninstall unless specified.
* **Idempotent Actions** – Re-running scripts does not cause duplicate installs or break the setup.

**3. Overview of the Tool**

This tool, often based on **PowerShell App Deployment Toolkit (PSADT)**, provides:

* **Unified Deployment Script** (Deploy-Application.ps1).
* **Support for MSI and EXE** silent installations.
* **Pre and Post Actions** to prepare or clean up systems.
* **Logging and Error Handling** built in.

It is widely used in enterprise IT for deploying software across hundreds or thousands of devices.

**4. Folder Structure**

A standard folder structure ensures organization and easier maintenance:

\DeploymentToolkit\

│ Deploy-Application.ps1 ← Main deployment script

│ Deploy-Application.exe ← Script launcher

│ Deploy-Application.xml ← Toolkit configuration file

│

├── Files\ ← Installer files (MSI/EXE)

├── SupportFiles\ ← Additional files (icons, configs, etc.)

└── Logs\ ← Execution logs

**Purpose:**

* **Files** – Store actual installation media.
* **SupportFiles** – Store dependencies or additional scripts.
* **Logs** – Record script actions for review.

**5. Tool Configuration**

Configuration is handled through:

* **XML Configuration File (Deploy-Application.xml)** – Defines toolkit title, default directories, and logging preferences.
* **Runtime Parameters** – Passed to the EXE launcher (e.g., -DeploymentType Install).
* **Environment Variables** – Set by the toolkit for dynamic path references.

Example:

<ToolkitConfig>

<ToolkitTitle>Office 365 Deployment</ToolkitTitle>

<DefaultInstallDir>C:\Program Files\Office</DefaultInstallDir>

<EnableLogging>True</EnableLogging>

</ToolkitConfig>

**6. Template Script**

The deployment script follows a **standard sequence**:

**6.1 Pre-Installation Phase**

* Notify user (optional).
* Check prerequisites (OS version, disk space).
* Stop running processes.
* Remove older versions.
* Backup configurations if needed.

Example:

Show-InstallationWelcome -CloseApps 'winword,excel' -Silent

**6.2 Installation Phase**

* Install application silently.

Example (MSI):

powershell

CopyEdit

Execute-MSI -Action Install -Path "$dirFiles\MyApp.msi" -Parameters '/qn /norestart'

Example (EXE):

powershell

CopyEdit

Execute-Process -Path "$dirFiles\setup.exe" -Parameters '/quiet /norestart'

**6.3 Post-Installation Phase**

* Create shortcuts.
* Configure registry keys.
* Copy settings/config files.
* Display completion message.

Example:

New-Shortcut -Path "$envPublic\Desktop\MyApp.lnk" -TargetPath "C:\Program Files\MyApp\MyApp.exe"

**7. Predefined Environment Variables Defined within Script and Its Usage**

| **Variable** | **Description** | **Example Usage** |
| --- | --- | --- |
| $dirFiles | Path to installer files | $dirFiles\app.msi |
| $dirSupportFiles | Path to SupportFiles folder | $dirSupportFiles\config.xml |
| $dirLogs | Path to logs folder | Write-Log -Message "Started Install" |
| $configToolkitTitle | Toolkit title from XML | Write-Host $configToolkitTitle |

These variables make scripts **portable** and prevent hardcoding of paths.

**8. Commands for Installation/Uninstallation**

**MSI Installation:**

Execute-MSI -Action Install -Path "$dirFiles\app.msi" -Parameters '/qn'

**MSI Uninstallation:**

Execute-MSI -Action Uninstall -Path '{ProductCode}' -Parameters '/qn'

**EXE Installation:**

Execute-Process -Path "$dirFiles\setup.exe" -Parameters '/quiet'

**EXE Uninstallation:**

Execute-Process -Path "$dirFiles\uninstall.exe" -Parameters '/S'

**9. Execute-MSI – Installation and Uninstallation**

Execute-MSI is a built-in function in the toolkit for MSI files.

**Installation:**

Execute-MSI -Action Install -Path "$dirFiles\myapp.msi" -Parameters '/qn /norestart'

**Uninstallation:**

Execute-MSI -Action Uninstall -Path '{ProductCode}' -Parameters '/qn /norestart'

**Advantages:**

* Simplifies msiexec usage.
* Handles logging automatically.
* Detects if an app is already installed.

**10. Execute-Process**

Execute-Process is used for EXE or custom installers.

**Installation:**

Execute-Process -Path "$dirFiles\setup.exe" -Parameters '/quiet /norestart'

**Uninstallation:**

Execute-Process -Path "$dirFiles\uninstall.exe" -Parameters '/S'

**Options:**

* -Path – Installer executable path.
* -Parameters – Silent/unattended install switches.
* -WaitForMsiExec – Waits for MSI processes before continuing.