PowerShell App Deployment Toolkit (PSADT) - Structured Summary

1. Overview of PSAppDeployToolkit

The **PSAppDeployToolkit (PSADT)** is a **PowerShell-based framework** designed to simplify and standardize **enterprise application deployment**. It offers a robust structure, user interface elements, and pre-built functions to streamline scripting tasks and improve deployment reliability.

Platform Value

- **Simplifies Scripting**: Automates deployment scripting to reduce manual complexity.
- **Consistent User Interface**: Ensures a standardized, professional user experience.
- **Boosts Success Rates**: Reduces errors and enhances deployment reliability.
- **Open-Source**: Freely available and community-driven for continuous improvement.

Core Concepts

- **ADTSession Object**: Stores deployment details (e.g., user input, status).
- **Pre-defined Functions**: Handles common tasks like app closure, validations, and installations.
- **Exit Codes**: Standardized codes indicate deployment outcomes for troubleshooting.
- **Deployment Structure**: Organized folder layout for maintainability.
- **Deployment Modes**: Supports interactive, silent, and non-interactive installations.

Usage

- **Application Deployment**: Wraps around setup files like MSI for enhanced automation.
- **Custom Scripting**: Write logic for tasks such as file management and user settings.
- **Function Use**: Access built-in functions for efficient task execution.
- **Script Execution**: Run via PowerShell or Deploy-Application.exe.
- **Session Management**: Use ADTSession for advanced deployment control and error handling.

Conclusion: PSADT is a **powerful, flexible, and user-friendly toolkit** for automating software deployments in enterprise environments, offering a more structured and reliable alternative to traditional PowerShell scripting.

2. Folder Structure

The **PowerShell App Deployment Toolkit (PSADT)** uses a well-defined **folder structure** to maintain clarity, organization, and consistency in deployment packages. Here's a breakdown of the key components:

Key Folders and Files

- **Toolkit**:
  Contains the core PSAppDeployToolkit framework, including all essential scripts and functions.
- **Examples**:
  Provides sample deployments (e.g., Adobe Reader, Microsoft Office) to guide script development.
- **Files**:
  Stores the **main installation files** (e.g., MSI or EXE installers) for the application being deployed.
- **SupportFiles** *(optional)*:
  Used for supplementary content like config files, custom scripts, or helper utilities needed during deployment.
- **Strings**:
  Holds **localized UI message files**, allowing multilingual support for deployment prompts and notifications.
- **Images**:
  Contains branding or UI elements like **logos and icons** used during the deployment process.
- **Configuration**:
  Includes the primary configuration file (Deploy-Configuration.psd1) and other relevant config settings.

Key Executable and Script Files

- **Deploy-Application.exe**:
  The **main executable** used to initiate the deployment process.
- **Deploy-Application.ps1**:
  The **PowerShell script** that contains all the deployment logic—this is where customizations are made.

3. Template Script: Deploy-Application.ps1

The **PowerShell Application Deployment Toolkit (PSAppDeployToolkit)** provides **template scripts**—most notably `Deploy-Application.ps1`—to help IT professionals create consistent and efficient deployment packages for enterprise applications.

---

1. Toolkit and Template Overview

- **PSAppDeployToolkit** simplifies complex PowerShell scripting for software deployment.
- It includes **pre-defined functions** and **UI elements** for common deployment tasks.
- Template scripts serve as a **customizable starting point** for deployments.

---

2. Deploy-Application.ps1 Script

- Core template used for both **installing and uninstalling** applications.
- Located in: `C:\AdminStudio Shared\PowerShellTemplate`
- Editable in **PowerShell ISE** for customization.
- Handles two main deployment types:
  - **Install**: Broken into `Pre-Install`, `Install`, and `Post-Install` phases.
  - **Uninstall**
- Relies on **AppDeployToolkitMain.ps1** for core logic and function execution.

---

3. Creating New Deployments

- Use internal commands to generate new deployment templates:
  - **Version 3 (legacy)**:

```
New-ADTTemplate -Destination C:\Temp\MyAppDeployment -Name
"MyOldAppDeployment" -Version 3
```

  - **Version 4 (current)**:

```
New-ADTTemplate -Destination C:\Temp\MyAppDeployment -Name
"MyAppDeployment"
```

---

4. Additional Resources

- Templates like `PSAppDeployToolkit_Template_v3.zip` and `v4.zip` are available on the **GitHub Releases** page.
- These can be downloaded and extracted locally for use.

---

5. Customizing Deploy-Application.ps1

- Modify script parameters and logic to suit specific deployment needs.
- Customize UI prompts and workflow.
- Acts as a **foundation** for building advanced, enterprise-grade deployment scenarios.

4. Configuration Using AppDeployToolkitConfig.xml

The **PowerShell App Deployment Toolkit (PSAppDeployToolkit)** enables centralized configuration and customization of application deployments through the **AppDeployToolkitConfig.xml** file. This allows for streamlined, consistent, and user-friendly deployment processes.

Configuration Steps

1. **Download and Extract**
   Download the toolkit from the official site or GitHub and extract the ZIP package.
2. **Locate Configuration File**
   Go to: Toolkit\AppDeployToolkit\AppDeployToolkitConfig.xml.
3. **Edit the XML File**
   Customize settings in a text or XML editor based on your deployment needs:
   - **Toolkit Options**: Admin rights, temp paths, logging behaviors.
   - **Banner, Logo & Icon**: Customize UI visuals for branding.
   - **MSI Options**: Set logging parameters and default install options for MSI files.
   - **UI Options**: Define balloon tips, timeout settings, and script exit codes.
   - **UI Messages**: Configure language-specific messages.
4. **Save and Apply**
   Saving the file updates configurations for **all deployments using the toolkit**.

Benefits of Configuring AppDeployToolkitConfig.xml

- **Centralized Control**: One file to configure multiple deployments.
- **Consistency**: Ensures uniform look, feel, and behavior across all deployments.

 Customization: Tailor the toolkit to match organizational standards and user expectations.

 Simplified Scripting: Reduces the need to repeat settings in every deployment script.

PowerShell App Deployment Toolkit (PSAppDeployToolkit / PSADT)

The **PowerShell App Deployment Toolkit (PSADT)** is an open-source scripting framework designed to simplify, standardize, and automate **software deployment** using **PowerShell**. It is widely used by IT professionals for consistent, reliable, and customizable installations and uninstallations across systems.

🔧 Key Features

- **Pre-built functions and UI prompts** for common deployment tasks
- **Centralized configuration** via AppDeployToolkitConfig.xml
- **Support for complex installation logic** using simple scripting
- **Custom cmdlets** like Set-RegistryKey, Execute-MSI, and more
- **UI enhancements**: Toast notifications, banners, and progress boxes
- **Error handling, logging, and rollback options**

📁 Folder Structure

- **Toolkit** – Contains core PSADT framework scripts.
- **Deploy-Application.ps1** – Main customizable deployment script.
- **Deploy-Application.exe** – Wrapper for running the main script.
- **Files** – Store MSI/MSP and other installation files.
- **SupportFiles** – Auxiliary scripts, configs, or additional files.
- **AppDeployToolkitConfig.xml** – Global configuration settings (e.g., MSI options, UI messages, banners).
- **Images, Strings, Configuration** – Used for UI branding and localization.

✍ Scripting Example – Registry Key Handling

You can handle registry manipulation easily with built-in cmdlets:

powershell
CopyEdit
Set-RegistryKey -Key 'HKCU:\Software\MySoftware\Scripts' -Name 'Version' -Value '2'

PSADT abstracts complex checks like whether the key exists or needs creation.

□ Configuration: AppDeployToolkitConfig.xml

Located in Toolkit\AppDeployToolkit, this file configures:

- **Toolkit options** (admin rights, temp paths, default registry/logging behavior)
- **UI customization** (logos, banners, timeout settings, notifications)
- **MSI settings** (install switches, log paths)
- **Languages and messages** for international support

Apply it once and reuse across all deployments.

💡 Development Tips

- Use **PowerShell ISE** for editing scripts.
- Enable **autocomplete** by placing the AppDeployToolkit folder into:

  makefile
  CopyEdit
  C:\Users\<username>\Documents\WindowsPowerShell\Modules

  Then save PSAppDeployToolkitMain.ps1 as PSAppDeployToolkit.psm1.

- Use **AppDeployToolkitHelp.ps1** to explore available functions and examples.

□? Script Structure: Deploy-Application.ps1

Three primary **actions**:

1. **Install**
2. **Uninstall**
3. **Repair**

Each has three **phases**:

- Pre-Action (e.g., kill processes, display warnings)
- Action (e.g., Execute-MSI)
- Post-Action (e.g., success messages, cleanup)

Example – Install Orca MSI:

Installation phase
Execute-MSI -Action Install -Path 'Orca.msi'

Example – Uninstall by GUID:

Execute-MSI -Action Uninstall -Path '{85F4CBCB-9BBC-4B50-A7D8-E1106771498D}'

🚀 Running the Script

Recommended method:

powershell.exe -ExecutionPolicy Bypass -File Deploy-Application.ps1

To uninstall:

powershell.exe -ExecutionPolicy Bypass -File Deploy-Application.ps1 -DeploymentType Uninstall

Alternatively, use Deploy-Application.exe to wrap the script execution.

□ Benefits

- **Centralization**: One config for all deployments
- **Consistency**: Same UI, behavior, and logging
- **Simplicity**: Minimal PowerShell knowledge needed
- **Scalability**: Works for small tools and enterprise apps

This toolkit is ideal for **system administrators** and **IT departments** looking to automate software delivery efficiently while maintaining professional standards and reliability.