**Execute-MSI & AppDeploy Toolkit — Commands and Examples**

PowerShell App Deployment Toolkit (PSAppDeployToolkit) commands we use for installing, patching, and uninstalling applications.

## Install (MSI) — script of ps1:

Execute-MSI -Action 'Install' -Path '7z2501-x64.msi' -Parameter '/QN'

## Uninstall (MSI by product code) — script of ps1:

Execute-MSI -Action 'Uninstall' -Path '{23170F69-40C1-2702-2501-000001000000}' -Parameter '/qn'

## Execute a regular EXE (silent 7-Zip example) — script of ps1:

Execute-Process -Path '7z2501-x64.exe' -Parameters '/S' -WindowStyle 'Hidden'

## Real-world execution (In PowerShell command):

When we actually deploy (the top-level invocation we run on the machine), we run the toolkit entry script with the action argument. Example full path invocations:  
  
C:\Users\adity\Downloads\Toolkit\Deploy-Application.ps1 Install

C:\Users\adity\Downloads\Toolkit\Deploy-Application.ps1 Uninstall

## What these mean:

• Execute-MSI: a PSAppDeployToolkit wrapper that calls msiexec.exe for MSI/MSP product install, uninstall, patch, or repair. Use -Path to pass either the MSI file path or a product GUID (for uninstall). It also automatically applies default switches from the toolkit config and generates a verbose log.

• Execute-Process: runs any executable (EXE, setup, etc.) with parameters I provide. Use this for installers that don't use MSI. We can control WindowStyle, wait behavior, and how return codes are handled.

• Execute-MSP / Install-MSUpdates: helpers that handle MSP (patch) files or batches of Microsoft updates (.msu, .msp, .exe) and call the appropriate installer logic; handy when we keep updates in a separate folder and want the toolkit to detect/apply them automatically.

## Practical notes & common gotchas

1) MSI path vs ProductCode: If we pass a file path (e.g., 'C:\Files\Example.msi'), the toolkit runs msiexec /i against it. If you pass a product GUID (the curly-braced GUID), it will run msiexec /x against that product to uninstall. Uninstall by GUID is useful when the MSI file is not available on the target machine.

2) Silent switches: Toolkits and best practice use quiet switches like /qn or /quiet so end users don’t see interactive dialogs. For MSI, common msiexec switches are /i (install), /x (uninstall), /qn (quiet, no UI) and logging switches; check MS docs when you need advanced options.

3) Return codes and detection: PSAppDeployToolkit wraps installers and interprets return codes — but always test on a VM. Some patches require extra MSI properties like REINSTALL and REINSTALLMODE when applied silently.

4) Windows Updates (.msu): Use wusa.exe or the toolkit helpers that expand .msu and apply updates; some complex cases need extraction and DISM to apply CABs directly.

## Useful PowerShell & command examples:

Execute-MSI & AppDeploy Toolkit — PSADT-style command structure

* **Copy-ADTFile** — PSADT helper (good for toolkit-aware copy, built-in logging & error handling)

Copy-ADTFile -Path "$dirSupportFiles\Readme-7zip.txt" -Destination "$envProgramFiles\7-Zip"

* **Copy-ADTFileToUserProfiles** — PSADT helper to copy into each user profile (useful for per-user configs)

Copy-ADTFileToUserProfiles -Path "$dirSupportFiles\UserSettings.ini" -Destination "AppData\Roaming\ProjectPhoenix" -Recurse

* **Copy-Item** — native PowerShell (use when PSADT features aren’t needed)

Copy-Item -Path "$dirSupportFiles\Installers\\*" -Destination "$envProgramFiles\ProjectPhoenix" -Recurse -Force

* **Remove-File** — PSADT helper for removing specific files (toolkit logging/error handling)

Remove-File -Path "$envUserDesktop\OldShortcut.lnk"

* **Remove-Folder** — PSADT helper to remove entire folder trees

Remove-Folder -Path "$envProgramFiles\OldProjectBuild" -Confirm:$false

* **Remove-Item** — native PowerShell (use -Force for hidden/read-only)

Remove-Item -Path "C:\Temp\OldLog.log" -Force -ErrorAction SilentlyContinue

* **Move-Item** — native PowerShell (PSADT doesn’t provide a dedicated move helper)

Move-Item -Path "$dirSupportFiles\chrome\_installer.exe" -Destination "D:\Backups\Installers\chrome\_installer.exe" -Force

* **Execute-MSI** — PSADT wrapper around msiexec

Execute-MSI -Action 'Install' -Path "$dirSupportFiles\7z2501-x64.msi" -Parameters '/qn'

Execute-MSI -Action 'Uninstall' -Path '{23170F69-40C1-2702-2501-000001000000}' -Parameters '/qn'

* **Execute-Process** — PSADT helper for EXE installers

Execute-Process -Path "$dirSupportFiles\7z2501-x64.exe" -Parameters '/S' -WindowStyle Hidden

* **Execute-MSP / Install-MSUpdates** — PSADT helpers for MSP/MSU/CAB workflows

# apply a patch (.msp) or call toolkit helper for grouped updates  
Execute-MSP -Path "$dirSupportFiles\Patch-Update.msp" -Parameters '/p'

# or use Install-MSUpdates when working with multiple .msu files in a folder  
Install-MSUpdates -Path "$dirSupportFiles\Updates"

* **reg.exe (CMD)** — quick string value add/update

reg add "HKLM\SOFTWARE\7-Zip" /v "InstallPath" /t REG\_SZ /d "C:\Program Files\7-Zip" /f