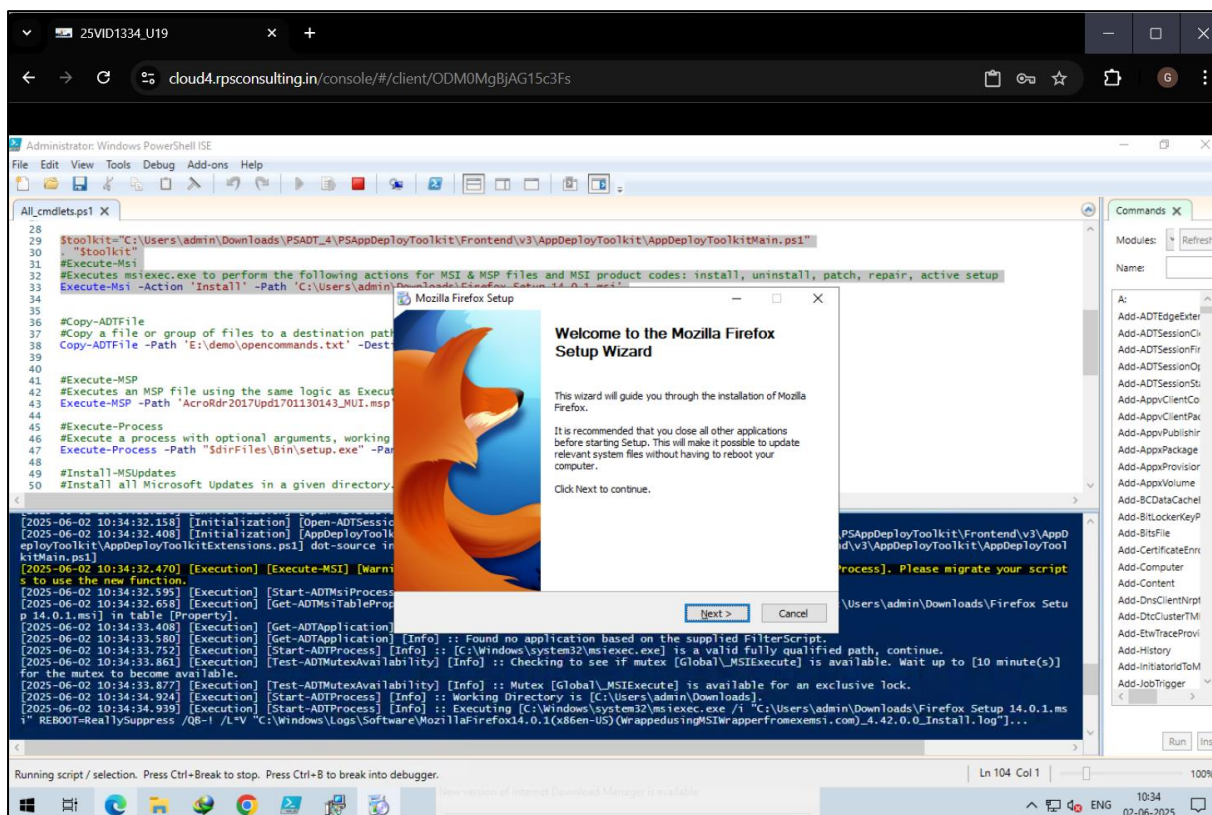


## Execute, Copy, Install and Active Setup Commands

### Execute-MSI:

Executes `msiexec.exe` to perform the following actions for MSI & MSP files and we can install, uninstall, update, etc as per the parameter `-Action`.

- `Execute-MSI -Action 'Install' -Path 'C:\Users\admin\Firefox Setup 14.0.1.msi'`
- `Execute-MSI -Action 'Install' -Path 'Adobe_FlashPlayer_11.2.202.233_x64_EN.msi' -Transform 'Adobe_FlashPlayer_11.2.202.233_x64_EN_01.mst' -Parameters '/QN'`
- `Execute-MSI -Action 'Uninstall' -Path '{26923b43-4d38-484f-9b9e-de460746276c}'`
- `Execute-MSI -Action 'Patch' -Path 'Adobe_Reader_11.0.3_EN.msp'`
- `Execute-MSI -Action Install -Path $AppMSIName -SkipMSIAlreadyInstalledCheck -ContinueOnError $False -LogName "${AppMSIName}_MSI"`



### Execute process:

Execute a process with optional arguments, working directory, window style.

- `Execute-Process -Path 'uninstall_flash_player_64bit.exe' -Parameters '/uninstall' -WindowStyle 'Hidden'`
- `Execute-Process -Path "$dirFiles\Bin\setup.exe" -Parameters '/S' -WindowStyle 'Hidden'`
- `Execute-Process -Path 'setup.exe' -Parameters '/S' -IgnoreExitCodes '1,2'`
- `Execute-Process -Path 'setup.exe' -Parameters '-s -f2 "$configToolkitLogDir\$installName.log' -WindowStyle 'Hidden'`
- `Execute-Process -Path 'setup.exe' -Parameters '/s /v "/ALLUSERS=1 /qn /L* "$configToolkitLogDir\$installName.log' -WindowStyle 'Hidden'`

### ExecuteMSP:

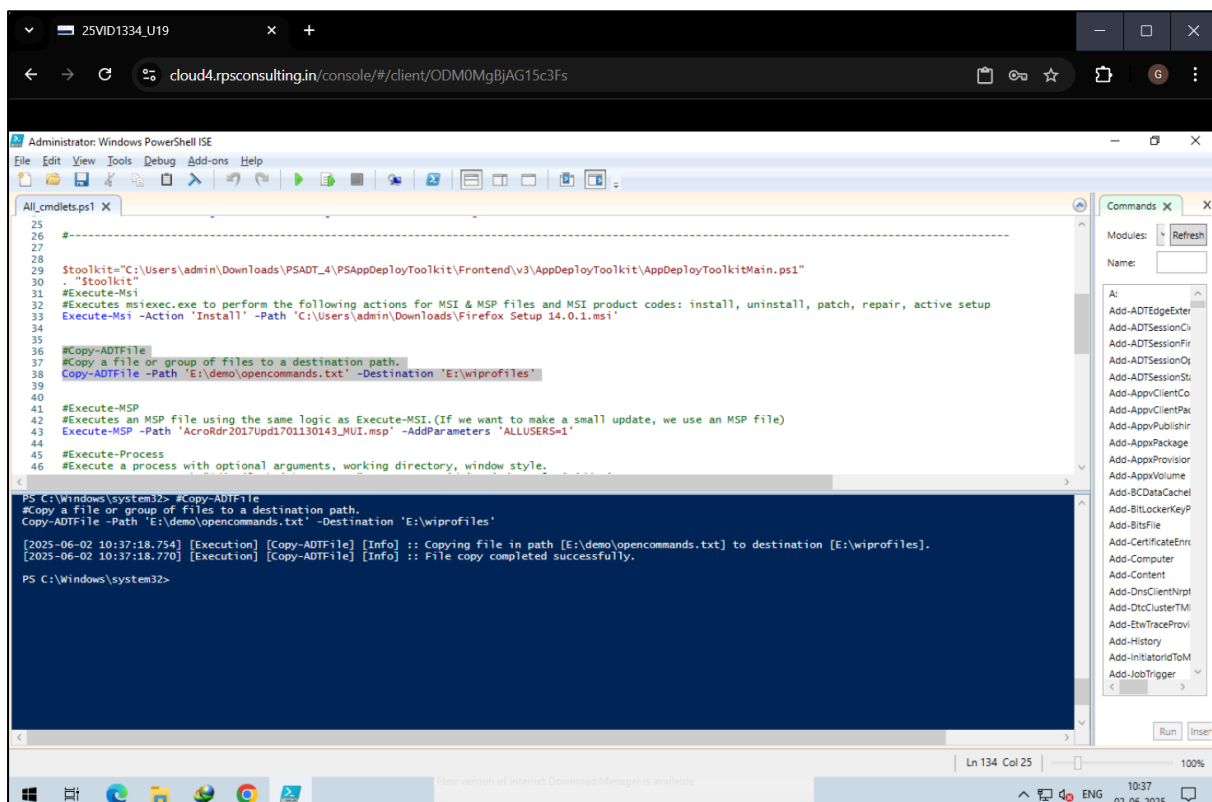
Reads SummaryInfo targeted product codes in MSP file and determines if the MSP file applies to any installed products(we use MSP file for a small updates)

- Execute-MSP -Path 'Adobe\_Reader\_11.0.3\_EN.msp'
- Execute-MSP -Path 'AcroRdr2017Upd1701130143\_MUI.msp' -AddParameters 'ALLUSERS=1'

### Copy-File:

Copy a file or group of files to a destination path.

- Copy-File -Path "\$dirSupportFiles\\*.\*)" -Destination "\$envTemp\tempfiles"
- Copy-File -Path "\$dirSupportFiles\MyApp.ini" -Destination "\$envWinDir\MyApp.ini"



### Install-Msupdate:

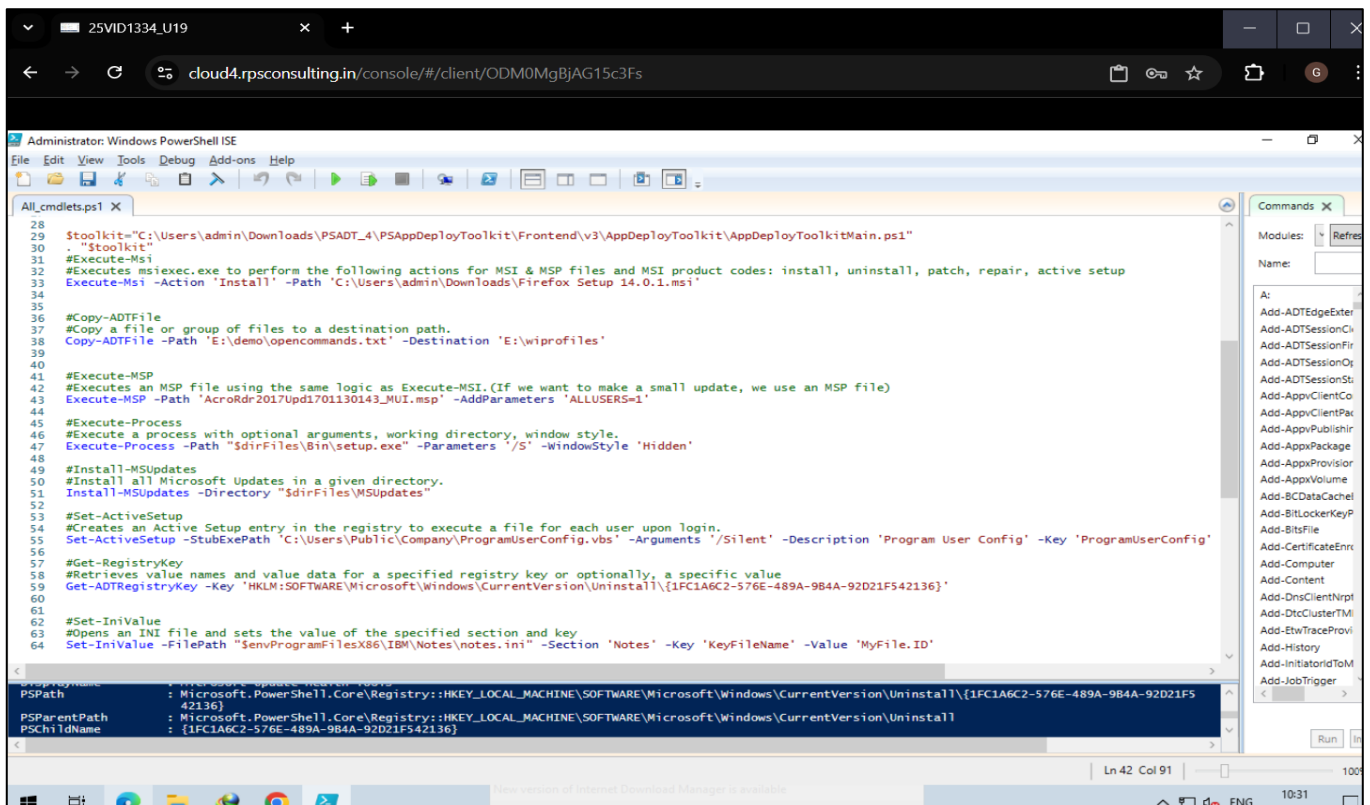
Install all Microsoft Updates of type ".exe", ".msu", or ".msp" in a given directory

- Install-MSUpdates -Directory "\$dirFiles\MSUpdates"

## Set-ActiveSetup:

Creates an Active Setup entry in the registry to execute a file for each user upon login.

- `Set-ActiveSetup -StubExePath 'C:\Users\Public\Company\ProgramUserConfig.vbs' -Arguments '/Silent' -Description 'Program User Config' -Key 'ProgramUserConfig' -Locale 'en'`
- `Set-ActiveSetup -StubExePath "$envWinDir\regedit.exe" -Arguments "/S \"%SystemDrive%\Program Files (x86)\PS App Deploy\PSAppDeployHKCUSettings.reg\" -Description 'PS App Deploy Config' -Key 'PS_App_Deploy_Config' -ContinueOnError $true`



```
28 $toolkit="C:\Users\admin\Downloads\PSADT_4\PSAppDeployToolkit\Frontend\v3\AppDeployToolkit\AppDeployToolkitMain.ps1"
29 . "$toolkit"
30 #Execute-Msi
31 #Executes msieec.exe to perform the following actions for MSI & MSP files and MSI product codes: install, uninstall, patch, repair, active setup
32 #Execute-Msi -Action 'Install' -Path 'C:\Users\admin\Downloads\Firefox Setup 14.0.1.msi'
33
34 #Copy-ADTFile
35 #Copy a file or group of files to a destination path.
36 #Copy-ADTFile -Path 'E:\demo\opencommands.txt' -Destination 'E:\wiprofiles'
37
38 #Execute-MSP
39 #Executes an MSP file using the same logic as Execute-MSI. (If we want to make a small update, we use an MSP file)
40 #Execute-MSP -Path 'AcroRdr2017Upd1701130143_MUI.msp' -AddParameters 'ALLUSERS=1'
41
42 #Execute-Process
43 #Execute a process with optional arguments, working directory, window style.
44 #Execute-Process -Path "$dirFiles\bin\setup.exe" -Parameters '/S' -WindowStyle 'Hidden'
45
46 #Install-MSUpdates
47 #Install all Microsoft Updates in a given directory.
48 #Install-MSUpdates -Directory "$dirFiles\MSUpdates"
49
50 #Set-ActiveSetup
51 #Creates an Active Setup entry in the registry to execute a file for each user upon login.
52 Set-ActiveSetup -StubExePath 'C:\Users\Public\Company\ProgramUserConfig.vbs' -Arguments '/Silent' -Description 'Program User Config' -Key 'ProgramUserConfig'
53
54 #Get-RegistryKey
55 #Retrieves value names and value data for a specified registry key or optionally, a specific value
56 Get-ADTRegistryKey -Key 'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\{1FC1A6C2-576E-489A-9B4A-92D21F542136}'
57
58 #Set-IniValue
59 #Opens an INI file and sets the value of the specified section and key
60 Set-IniValue -FilePath "$envProgramFilesX86\IBM\Notes\notes.ini" -Section 'Notes' -Key 'KeyFileName' -Value 'MyFile.ID'
```

PSPath : Microsoft.PowerShell.Core\Registry::HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\{1FC1A6C2-576E-489A-9B4A-92D21F542136}

PSParentPath : Microsoft.PowerShell.Core\Registry::HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall

PSChildName : {1FC1A6C2-576E-489A-9B4A-92D21F542136}

## Execute-MSI Installation and Uninstallation

In PSAppDeployToolkit, Execute-MSI is a function used for installing, uninstalling, patching, repairing, or performing active setup on MSI and MSP files, or using MSI product codes. For installation, it defaults to using the /i (install) switch of msixexec.exe, while for uninstallation, it uses the /x (uninstall) switch.

### Installation

**Basic Usage:** Use -Action Install and provide the path to the MSI file.

**Parameters:** we can add or override default parameters to customize the installation process/qn – Silent installation

- /norestart or REBOOT=ReallySuppress – Prevent system restart
- /L\*v – Enable verbose logging

**Example:** Execute-MSI -Action Install -Path "C:\path\to\app.msi" -Parameters "/qn REBOOT=ReallySuppress"

### Uninstallation

**Basic Usage:** Use -Action Uninstall with the MSI path or product code.

**Parameters:** Similar to installation; use /qn for silent uninstall and /L\*v for logging

**Example:** Execute-MSI -Action Uninstall -Path "C:\path\to\app.msi" -Parameters "/qn REBOOT=ReallySuppress"

### Important Considerations

- **Product Code:** You can uninstall using the MSI's product code instead of the file path.
- **Logging:** Use -LogName to specify a custom log file.
- **Pre-flight Checks:** Disable MSI installed check with -SkipMSIAAlreadyInstalledCheck.
- **Error Handling:** -ContinueOnError can be used to continue execution on failure.

## Commands For Installation and Uninstallation

### Installation

- **Execute-MSI:** Primary cmdlet to install MSI packages.

#### Parameters:

-Action Install: Defines the operation.

-Path: Path to the .msi file.

-DeployMode: Set deployment mode (Silent, Interactive, or NonInteractive).

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

### Uninstallation

- **Uninstall-ADTApplication:** Used for uninstalling apps previously installed using PSAppDeployToolkit.
- **Execute-MSI:** Can also uninstall using the product's GUID or MSI path.

**Example:** Execute-MSI -Action Uninstall -Path "{GUID of the product}"

**Start-Process with msixexec.exe:** Alternative method to manually run uninstall commands.

**Example:** Start-Process -FilePath "C:\Windows\System32\msiexec.exe" -ArgumentList "/x{GUID of the product} /qn" -Wait

#### Other related commands:

- **Get-RunningProcesses**  
Checks for running processes that may need to be closed before proceeding with installation or uninstallation.
- **Show-WelcomePrompt**  
Displays a customizable welcome message to inform or prompt the user before continuing with the deployment.
- **Remove-MSIApplications**  
Uninstalls specific MSI-based applications from the system.
- **Unblock-ADTAppExecution**  
Unblocks an application that was previously blocked by the toolkit.
- **Unregister-ADT-Dll**  
Unregisters a specified DLL file