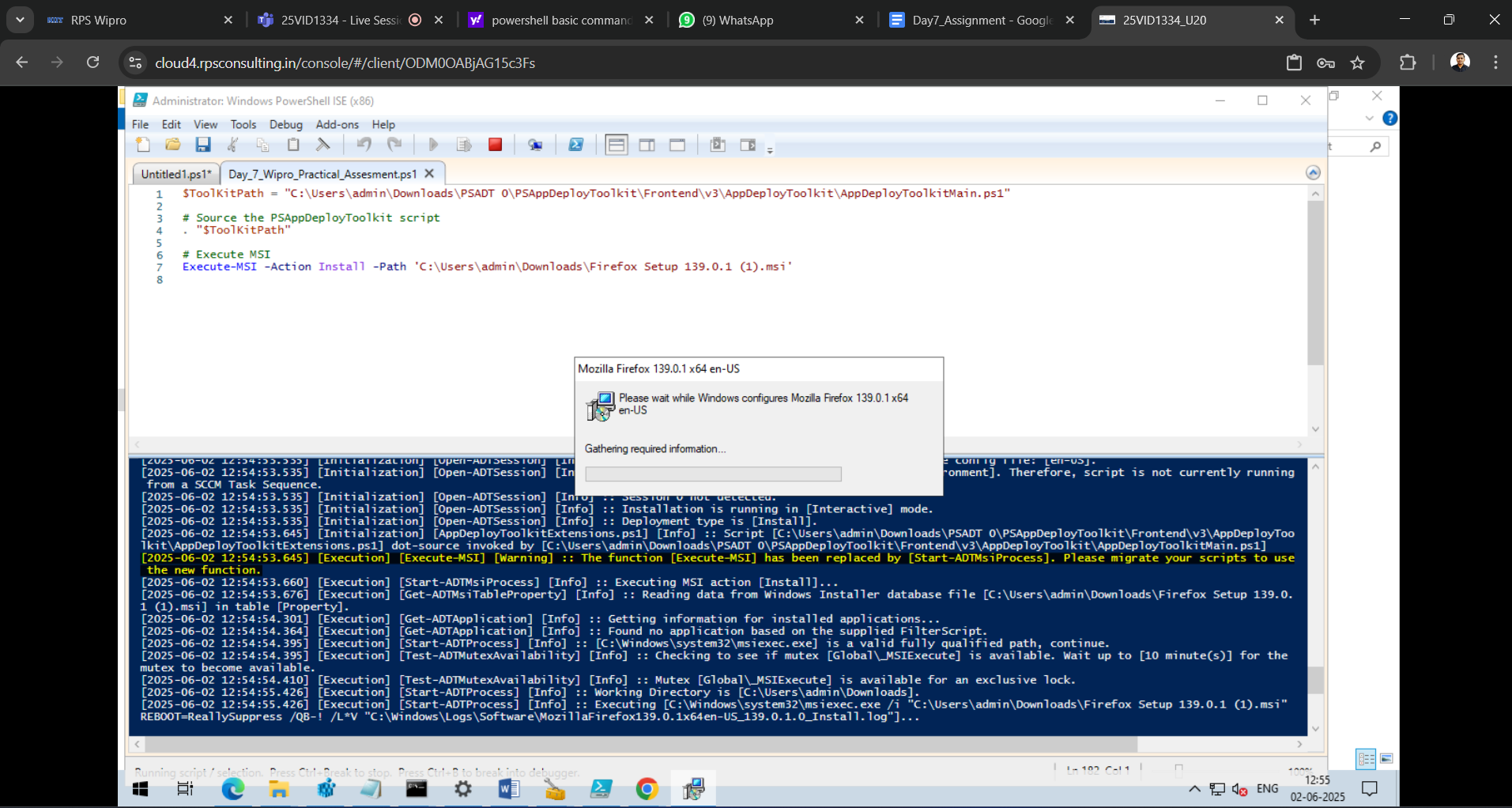
**Execute, Copy, Install and Active Setup Commands**

**List of commands:**

1. **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 'Adobe\_FlashPlayer\_11.2.202.233\_x64\_EN.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"



1. **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`""
* Execute-Process -Path 'setup.exe' -Parameters "/s /v`"ALLUSERS=1 /qn /L\* `"$configToolkitLogDir\$installName.log`"`""

1. **Execute-MSP**

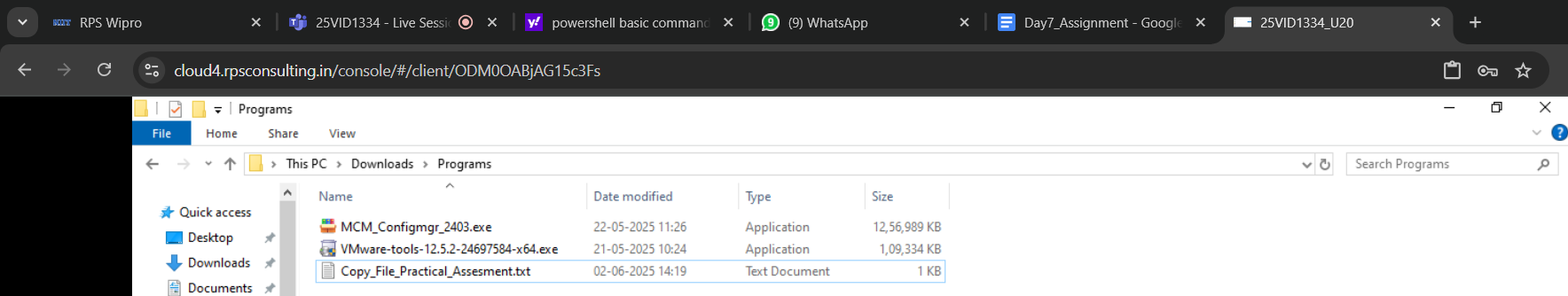
Reads SummaryInfo targeted product codes in MSP file and determines if the MSP file applies to any installed products.

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

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"



1. **Install-MSUpdate**

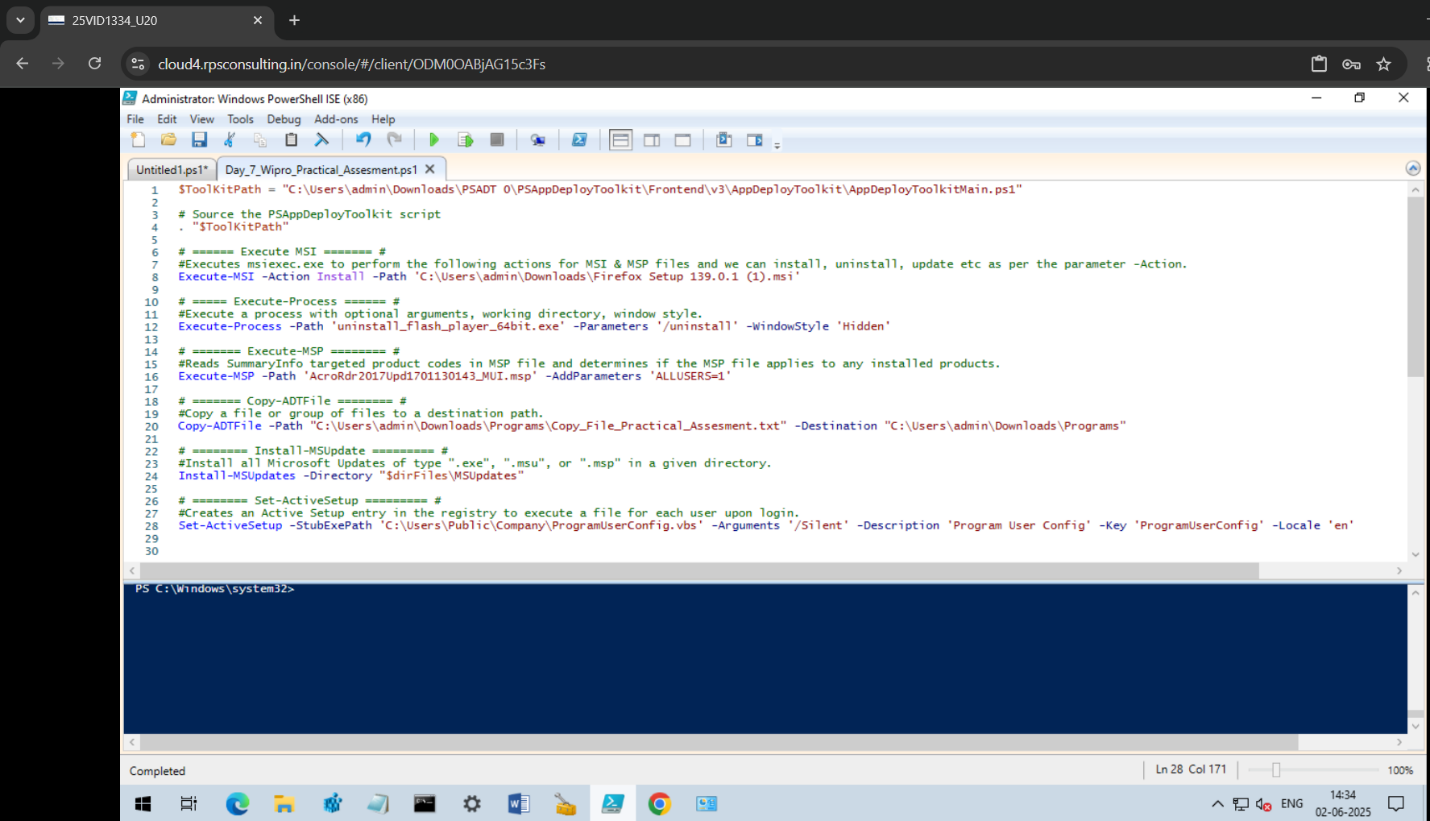
Install all Microsoft Updates of type ".exe", ".msu", or ".msp" in a given directory.

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

1. **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



**Executing MSI Installation and Uninstallation**

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 msiexec.exe while for uninstallation it uses the /x (uninstall) switch.

1. **Installation**

* **Basic Usage:** Provide the path to the MSI file and the -Action parameter set to "Install".
* **Parameters:** We can add or override default parameters to customize the installation process such as /qn for silent installation, /norestart to prevent a system restart or /L\*v for verbose logging.
* **Example:** Execute-MSI -Action Install -Path "C:\path\to\app.msi" -Parameters "/qn REBOOT=ReallySuppress"

1. **Uninstallation**

* **Basic Usage:** Set -Action to "Uninstall" and provide the path to the MSI file.
* **Parameters:** Similar to installation we can use parameters to control the uninstallation process such as /qn for silent uninstallation or /L\*v for logging.
* **Example:** Execute-MSI -Action Uninstall -Path "C:\path\to\app.msi" -Parameters "/qn REBOOT=ReallySuppress".

**Important Considerations**

* **Product Code:** We can also use the product code of an installed MSI to uninstall it rather than providing the MSI file path.
* **Logging:** Use the -LogName parameter to specify the log file path and name for more detailed information about the installation or uninstallation process.
* **Pre-flight Checks:** By default, Execute-MSI will check if the MSI is already installed before attempting to install it. You can disable this check using the -SkipMSIAlreadyInstalledCheck parameter.
* **Error Handling:** We can use the -ContinueOnError parameter to control how the script behaves if an error occurs during the installation or uninstallation process.

**Commands for Installation and Uninstallation**

In PSADT, the main commands for installation and uninstallation are Execute-MSI and Uninstall-ADTApplication. Execute-MSI is used to run MSI installers while Uninstall-ADTApplication handles uninstalling applications.

1. **Installation**

* **Execute-MSI:** This cmdlet is used to execute MSI files. We can specify the path to the MSI file and various parameters like -Action Install, -Path and -DeployMode.
* **Example:** Execute-MSI -Action Install -Path "C:\path\to\your.msi" -DeployMode Silent

1. **Uninstallation**

* **Uninstall-ADTApplication:** This cmdlet uninstalls applications that were previously installed using PSAppDeployToolkit.
* **Execute-MSI:** We can also use Execute-MSI with -Action Uninstall to uninstall applications.
* **Example:** Execute-MSI -Action Uninstall -Path "{GUID of the product}"
* **Start-Process with msiexec:** We can also use the Start-Process cmdlet to call the msiexec.exe command with uninstall arguments.
* **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 might need to be closed before an installation or uninstallation.
* **Show-WelcomePrompt:** Displays a welcome prompt to the user.
* **Remove-MSIApplications:** Removes specified MSI applications.
* **Unblock-ADTAppExecution:** Unblocks the execution of an application that was blocked.
* **Unregister-ADTDll:** Unregisters a specific DLL.