##\*============================================================================

##\*

##\* FILE: Deploy-Application.ps1

##\*

##\* DESCRIPTION: This is the main PSADT script for installing an MSI with

##\* Active Setup.

##\*

##\*============================================================================

##\*============================================================================

##\* VARIABLE DECLARATION

##\*============================================================================

## Package Details (Required)

[string]$appVendor = "Contoso"

[string]$appName = "Sample App"

[string]$appVersion = "1.0.0"

##\*============================================================================

##\* SCRIPT EXECUTION

##\*============================================================================

# Define the installation source path

$MSIFile = "YourApp.msi"

$installSource = "$dirFiles\$MSIFile"

$msiParameters = "/qn ALLUSERS=1"

#region Installation

##\*============================================================================

##\* 1. INSTALLATION

##\*============================================================================

Write-Log "Starting the installation phase..."

# Check if the MSI file exists before proceeding.

If (-not (Test-Path -Path $installSource)) {

Write-Log "Error: The MSI file was not found at '$installSource'. Aborting installation." -Severity "Error"

Exit-Script -ExitCode 1

}

# Install the application's MSI

# Use a custom MSI product code for your application. You can find this by

# running 'Get-MsiProductInfo' or looking in the MSI properties.

# The 'Execute-MSI' function will handle logging and error handling.

Write-Log "Installing MSI: '$MSIFile' with parameters '$msiParameters'"

Execute-MSI -Action Install -Path $installSource -Parameters $msiParameters

# Check if the MSI installation was successful

If ($global:ExitCode -ne 0) {

Write-Log "MSI installation failed with exit code $global:ExitCode." -Severity "Error"

Exit-Script -ExitCode $global:ExitCode

}

##\*============================================================================

##\* 2. ACTIVE SETUP CONFIGURATION

##\*============================================================================

# Purpose: Active Setup is used to perform per-user configuration the first time

# a user logs in. This is useful for things like creating user-specific

# registry keys, copying files to the user's profile, or running a

# a small configuration script.

Write-Log "Creating Active Setup registry key for per-user configuration."

# Define Active Setup parameters. The key name should be a unique GUID.

# Use a descriptive title for the Active Setup entry.

$activeSetupGUID = "{E0D3643B-2895-460A-82D1-B83F3730261A}"

$activeSetupTitle = "Your App Active Setup"

# The path to the command or script that will run for each user.

# In this example, we'll run a PowerShell script.

# You can also use a command line to launch an EXE or a batch file.

# The command path is typically placed in the package's 'SupportFiles' directory.

$stubPath = "`"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe`" -ExecutionPolicy Bypass -File `"C:\Program Files (x86)\YourApp\UserConfig.ps1`""

# Use 'Set-RegistryKey' to create the Active Setup entry.

# The key is created under the HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components

# path. This is a machine-wide entry that will trigger for each user.

Set-RegistryKey -Key "HKLM:\SOFTWARE\Microsoft\Active Setup\Installed Components\$activeSetupGUID" -Value $activeSetupTitle -Name "(Default)"

Set-RegistryKey -Key "HKLM:\SOFTWARE\Microsoft\Active Setup\Installed Components\$activeSetupGUID" -Name "Version" -Value "$appVersion"

Set-RegistryKey -Key "HKLM:\SOFTWARE\Microsoft\Active Setup\Installed Components\$activeSetupGUID" -Name "StubPath" -Value $stubPath

Write-Log "Active Setup registry keys created successfully."

#endregion Installation

#region Post-Installation

##\*============================================================================

##\* 3. POST-INSTALLATION

##\*============================================================================

Write-Log "Post-installation tasks complete. Script finished successfully."

# This section is for any final tasks after the installation is complete.

# For example, starting a service, creating a desktop shortcut, etc.

# Note: The Active Setup configuration is done in the installation phase,

# but the command defined in 'StubPath' runs in the post-installation phase

# from a user's perspective.

#endregion Post-Installation

#region Uninstallation

##\*============================================================================

##\* 4. UNINSTALLATION

##\*============================================================================

Write-Log "Uninstallation phase initiated. This will uninstall the application."

# Example of uninstalling the application.

# Get the product code of the application to be uninstalled.

# $productCode = Get-MsiProductInfo -Name "$appName"

# If ($productCode) {

# Execute-MSI -Action Uninstall -Path $productCode

# }

# Clean up Active Setup registry keys

Remove-RegistryKey -Key "HKLM:\SOFTWARE\Microsoft\Active Setup\Installed Components\$activeSetupGUID"

#endregion Uninstallation

#region Post-Uninstallation

##\*============================================================================

##\* 5. POST-UNINSTALLATION

##\*============================================================================

# This section is for any final tasks after uninstallation is complete.

Write-Log "Post-uninstallation tasks complete. Script finished successfully."

#endregion Post-Uninstallation