# 📦 Application Packaging Guidelines: Creating Silent EXE Installs & Uninstalls

This guide provides a consolidated, step-by-step procedure for a mid-level Application Packager to successfully automate the silent installation and uninstallation of **Executable (.EXE) installers**.

## 1. 📋 Pre-Packaging Requirements Checklist (SME/App Owner)

Before starting the packaging process, you **must** obtain the following mandatory information from the Subject Matter Expert (SME) or Application Owner. This prevents rework and ensures the package meets deployment requirements.

| Requirement Check | Detail to Confirm | Rationale |
| --- | --- | --- |
| **Application Name & Version** | Exact, full name (e.g., "AppX Viewer v4.1.0.12"). | Necessary for asset inventory and package naming convention. |
| **Silent Install Switches** | The **exact vendor-provided command-line parameters** for silent installation. | This is the most crucial piece of information for EXE files. |
| **Uninstallation Command/Switch** | The exact silent command for uninstallation (often the same as install: /s or /S). | Required for deployment management and cleanup. |
| **Installation Scope** | Does it install per-user (HKCU) or per-machine (HKLM)? | Determines the required execution context (User vs. System). |
| **Installation Path** | The standard/required destination folder (e.g., C:\Program Files\AppX). | Ensures compliance with organizational standards and avoids configuration issues. |
| **Custom Configuration** | Are there any required **post-installation** registry keys, configuration files, or license files (e.g., settings.ini, license.dat)? | These must be captured and included in the final package structure. |
| **Dependencies** | Does the application require any prerequisites (e.g., .NET Framework, Visual C++ Redistributables, Java)? | Dependencies must be packaged and deployed **before** the main application. |
| **Execution Context** | Does the installer require elevated privileges (Administrator/System)? (Always assume **Yes** for silent installs). | Ensures the installer can write to HKEY\_LOCAL\_MACHINE and C:\Program Files. |

## 2. 🔍 Process to Determine Silent Switches for EXE

Because there is no universal standard for EXE installers, you must use a systematic approach to find the correct silent switches.

### Step 2.1: Check Vendor Documentation (The Best Method)

1. **Search the Vendor's Knowledge Base (KB) or Support Site:** Use the application name, version, and terms like "silent install," "command line," or "unattended installation."
2. **Look for Installer Type:** If the vendor mentions the installer uses a specific tool (e.g., **NSIS**, **Inno Setup**, **InstallShield**), proceed to the common switch testing below.

### Step 2.2: Command Line Help (The Quickest Method)

Attempt to run the EXE with standard help switches to see if it lists supported parameters.

| Command to Try | Description |
| --- | --- |
| AppInstaller.exe /? | Universal help switch. |
| AppInstaller.exe /help | Detailed help switch. |
| AppInstaller.exe /h | Short help switch. |

If a dialogue box appears with options, look for keywords like "silent," "unattended," or "quiet."

### Step 2.3: Common Switch Testing (The Guessing Method)

If documentation is unavailable and the help switches fail, test the following common switches used by popular EXE installer wrappers. **Run the installer from an elevated command prompt.**

| Installer Type | Common Silent Switch (Install) | Common Silent Switch (Uninstall) |
| --- | --- | --- |
| **NSIS** (Nullsoft Scriptable Install System) | AppInstaller.exe /S | Same as install: Uninstall.exe /S |
| **Inno Setup** | AppInstaller.exe /VERYSILENT /SUPPRESSMSGBOXES | Same as install: Uninstall.exe /VERYSILENT |
| **InstallShield (Basic/Silent Mode)** | AppInstaller.exe /s | Use the UninstallString from ARP (see Step 3.2), often with /s. |
| **Generic/Simple EXE** | AppInstaller.exe /silent or AppInstaller.exe /q | Same as install, or check the UninstallString. |

### Step 2.4: Check for Embedded MSI (The Advanced Method)

Many large EXEs are simply **self-extracting archives** that contain an MSI file. If the common switches fail, you must try to extract the contents.

1. Run the EXE with switches known to trigger extraction (e.g., /extract, /a, /e, or standard ZIP commands if it's a ZIP archive).
2. If the installer starts and prompts for a path, let it extract, but **do not click "Install"**. Look in the %TEMP% folder for an .msi file or other setup files.
3. If an **MSI** is found, the installation is simpler, using standard MSI commands:
   * **Install:** msiexec /i "Extracted.msi" /qn
   * **Uninstall:** msiexec /x "{ProductCode}" /qn

## 3. ⚙️ Step-by-Step Packaging Execution

Once the correct silent switches are determined, execute the installation and uninstallation, and capture the process.

### Step 3.1: Silent Installation

**Objective:** Run the installer in a way that requires zero user interaction and captures all files/registry keys created.

1. **Open an Elevated Command Prompt (Run as Administrator).**
2. Navigate to the directory containing your application EXE.
3. Execute the installation command using the identified silent switch.

| Installer Type | Example Command (Install) |
| --- | --- |
| **NSIS** | AppInstaller.exe /S |
| **Inno Setup** | AppInstaller.exe /VERYSILENT /SUPPRESSMSGBOXES |
| **InstallShield** | AppInstaller.exe /s |

1. **Verification:** Check the target installation folder (e.g., C:\Program Files\AppX) and verify that shortcuts are created and the application launches successfully.

### Step 3.2: Silent Uninstallation

**Objective:** Determine the application's unique Uninstall String from the Windows Registry and append the silent switch.

1. **Open the Registry Editor (regedit).**
2. Navigate to the Uninstall key for your installation scope:
   * **64-bit Apps:** HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall
   * **32-bit Apps:** HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall
3. **Find the Application Key:** Search the subkeys for a DisplayName matching your application. This subkey contains the uninstallation command.
4. **Extract the UninstallString:** Copy the value of the UninstallString entry.
   * *Example Value:* "C:\Program Files\AppX\Uninstall.exe"
5. **Execute the Silent Uninstall:** Append the silent switch (usually the same as the install switch) to the extracted command.

| Scenario | Extracted UninstallString | Final Command (Uninstall) |
| --- | --- | --- |
| **Standard EXE** | "C:\Program Files\AppX\Uninstall.exe" | "C:\Program Files\AppX\Uninstall.exe" /S |
| **EXE (with parameters)** | "C:\Program Files\AppX\Uninstall.exe" -options | "C:\Program Files\AppX\Uninstall.exe" -options /S |
| **NSIS** | "C:\Program Files\AppX\Uninstall.exe" | "C:\Program Files\AppX\Uninstall.exe" /S |
| **Inno Setup** | "C:\Program Files (x86)\AppX\unins000.exe" | "C:\Program Files (x86)\AppX\unins000.exe" /VERYSILENT |

1. **Verification:** After the command completes, verify the following:
   * The application's installation folder is removed.
   * The entry is gone from **Programs and Features (ARP)**.
   * No residual shortcuts remain on the desktop or start menu.

### ⚠️ Critical Note for Deployment

Always execute the final silent installation and uninstallation commands using the **System** account context within your deployment tool (e.g., SCCM, Intune, or other MDM) to replicate the environment of a standard automated deployment.