

Simple Installer Builder

Universal Standalone Windows Installer (.exe) & Setup Package Creator

Instant Standalone Executables

Produces fully self-extracting .exe installers with zero external dependencies, runtimes, or scripting engines.

SmartScreen & Zone Unblock

Automated Zone.Identifier unblocking eliminates frustrating "Windows protected your PC" roadblocks for users.

Single File & Folder Modes

Bundle single-file .exe utilities in seconds or package complex multi-directory applications with automatic dev junk filtering.

Clean Windows Uninstaller

Integrated uninstallation engine with desktop/start menu shortcut cleanup and Windows Settings app registration.

1. Introduction & Architecture

Simple Installer Builder is a lightweight, modern, and completely free application designed to eliminate the complexity of creating software installers for Windows. Traditional tools such as Inno Setup, NSIS, or WiX require mastering complex proprietary scripting languages, compiling external scripts, and manually handling shortcut links and registry entries.

Simple Installer Builder replaces this tedious workflow with an intuitive, modern graphical interface and a powerful CLI automation engine. Anyone can package an executable or an entire application folder into a polished, professional setup installer in under 15 seconds.

Architecture Overview

Installers created by Simple Installer Builder are built upon an embedded **Setup Stub Engine**. Standalone installers do not require temporary batch files or separate archive unzipppers to extract files. Instead, the final binary is assembled using an appended-payload architecture:

Binary Structure of Standalone Installer:

```
+-----+
| SimpleInstaller.Stub.exe (WPF / .NET Native 64-bit Setup Engine) |
+-----+
| Payload.zip (Application files, metadata manifest, extracted app icons) |
+-----+
| Magic Footer [STUBZIP1 (8 bytes) + ZipOffset (8 bytes) + ZipLength (8 bytes)] |
+-----+
```

When the user executes the resulting `[App]-Setup.exe`, the stub engine reads the 24-byte magic footer from the end of its own process file, opens a virtual stream into the embedded payload zip without extracting the zip to disk first, parses the application manifest, and presents a modern Fluent-styled installation dialog.

No Admin Privileges Needed by Default: Installers target `%LocalAppData%\Programs\[AppName]` by default. This ensures any standard Windows user can install your software without triggering User Account Control (UAC) prompts or requiring local administrator credentials.

2. SmartScreen & Security Unblocking

One of the biggest obstacles developers face when distributing Windows applications online is Microsoft Defender SmartScreen. When a user downloads a ZIP file or an executable via a web browser (Google Chrome, Microsoft Edge, Firefox), Windows automatically attaches an alternate data stream named `Zone.Identifier` (Zoneld=3 for the Internet Zone).

How Simple Installer Builder Solves This

- Automatic In-Memory File Unblocking:** During installation, the setup stub inspects all extracted executable files (`.exe`, `.dll`, `.ico`) and invokes the Windows Kernel API (`kernel32.dll!DeleteFileW` with `:Zone.Identifier`). This permanently removes the mark of the web from your application binaries.

- **Bootstrap Script (`Install.cmd`):** When generating the distribution ZIP package (`[App]-Windows.zip`), Simple Installer Builder includes an automated 1-click bootstrap script:

```
@echo off
chcp 65001 >nul
echo =====
echo   MyApp - Setup & Installation
echo =====
powershell -NoProfile -ExecutionPolicy Bypass -Command "Get-ChildItem -Path '%~dp0*' -Recurse | Unblock-
start "" "%~dp0MyApp-Setup.exe"
```

When users extract the ZIP and run `Install.cmd` , PowerShell recursively unblocks all files before launching the setup installer.

Best Practice: Always provide your end-users with the generated `[App]-Windows.zip` file. It ensures the smoothest first-run experience across all modern Windows versions.

3. Step-by-Step Walkthrough

1 Section 1: Application or Files to Install

Select how your application files are organized:

- **Single executable file (.exe) [Default & Recommended]:** For self-contained single-file applications. Simple Installer Builder only packages the single executable. Packaging takes just 1–2 seconds.
- **Entire application folder:** For applications that require shared DLLs, subdirectories (e.g., `assets/` , `locales/` , `plugins/`). The tool automatically excludes development caches like `.git/` , `.vs/` , and `obj/` folders.
- **Drag & Drop:** You can drag any `.exe` file or folder directly into the Simple Installer Builder window from File Explorer.

2 Section 2: Application Details

Configure how your application appears in setup dialogs, shortcuts, and Windows Settings:

- **Application Name:** Automatically detected from the executable's version metadata.
- **Version:** The version string (e.g., `1.0.0`).
- **Publisher / Developer:** Company or developer name.
- **Application Icon:**
 - **Extract from .exe (Default):** Extracts the embedded icon directly from your application's PE resource header.
 - **Generic default icon:** Uses the modern built-in installer icon.
 - **Custom file:** Choose an external `.ico` or `.png` file (automatically converted to multi-size icon).

3 Section 3: Installation Settings

- **Default Installation Directory:**
 - `%LocalAppData%\Programs` (Per-user): Recommended. Installs without UAC prompt or admin rights.
 - `C:\Program Files` (All users): Machine-wide installation. Prompts for administrator elevation.
- **Create Desktop Shortcut:** Places a shortcut with the app icon on the desktop.
- **Create Start Menu Shortcut:** Adds your program to the Windows Start Menu programs folder.
- **Automatically unblock security tags:** Enables in-engine Zone.Identifier stripping.
- **Register uninstaller (Uninstall.exe):** Creates `Uninstall.exe` and registers your app in *Windows Settings* > *Apps* > *Installed apps*.

4 Section 4: Output Setup Package & Build

Specify the output path (defaults to desktop or source directory as `[AppName]-Setup.exe`). Leave "**Also create a distribution ZIP package**" checked. Click  **Build Installer (.exe)**.

4. Uninstaller & Clean Removal

Simple Installer Builder automatically equips every installed application with a dedicated uninstaller.

Uninstallation Capabilities

- **Windows Settings Integration:** Users can search for your application in *Settings > Apps > Installed apps* (or *Control Panel > Programs and Features*) and click **Uninstall**.
- **Process Termination:** If the application is currently running when uninstallation starts, the uninstaller automatically closes the running processes to avoid file lock errors.
- **Shortcut Removal:** Cleans up desktop and Start menu `.lnk` shortcuts.
- **Registry Cleanup:** Removes uninstall registry keys under `HKCU\Software\Microsoft\Windows\CurrentVersion\Uninstall\[AppName]` .
- **Self-Deleting Base Directory:** Deletes application files and executes a non-blocking delayed command to delete the installation folder and `Uninstall.exe` cleanly.
- **Silent Uninstall Support:** System administrators can invoke silent uninstallation via:

```
Uninstall.exe --uninstall --silent
```

5. Command Line Interface (CLI) Automation

Simple Installer Builder includes a full command-line mode for automated build scripts, continuous integration (CI/CD) pipelines, and batch generation.

Syntax

```
SimpleInstallerBuilder.exe --build <path-to-config.json>
SimpleInstallerBuilder.exe -b <path-to-config.json>
```

JSON Configuration Reference

Property	Type	Description	Example
AppName	String	Display name of the application (Required)	"MyTool"
AppVersion	String	Semantic version string	"1.2.0"
Publisher	String	Developer or organization name	"Acme Labs"
SourceMode	String	"SingleFile" or "Folder"	"SingleFile"
SingleFilePath	String	Absolute path to the executable (when SourceMode is SingleFile)	"C:\\Build\\MyTool.exe"
SourceFolderPath	String	Absolute path to source directory (when SourceMode is Folder)	"C:\\Build\\AppDir"
MainExeName	String	Executable file name inside folder (when SourceMode is Folder)	"MyTool.exe"
IconMode	String	"FromExe" , "Custom" , or "Default"	"FromExe"
IconPath	String	Path to .ico or .png (when IconMode is Custom)	"C:\\Assets\\logo.png"
DefaultDirType	String	"LocalPrograms" or "ProgramFiles"	"LocalPrograms"
CreateDesktopShortcut	Boolean	Create shortcut on user's desktop	true
CreateStartMenuShortcut	Boolean	Create shortcut in Start Menu programs	true
AutoUnblock	Boolean	Strip Zone.Identifier internet mark	true
CreateUninstaller	Boolean	Include Uninstall.exe and register in Windows	true

Property	Type	Description	Example
CreateZipPackage	Boolean	Create [App]-Windows.zip with Install.cmd	true
OutputExePath	String	Full destination path for generated setup .exe	"C:\\Dist\\MyTool-Setup.exe"

Example Configuration File (setup_config.json)

```
{
  "AppName": "PhotoEnhancer",
  "AppVersion": "2.4.1",
  "Publisher": "PixelCraft Studios",
  "SourceMode": "SingleFile",
  "SingleFilePath": "D:\\Projects\\PhotoEnhancer\\bin\\Release\\PhotoEnhancer.exe",
  "IconMode": "FromExe",
  "DefaultDirType": "LocalPrograms",
  "CreateDesktopShortcut": true,
  "CreateStartMenuShortcut": true,
  "AutoUnblock": true,
  "CreateUninstaller": true,
  "CreateZipPackage": true,
  "OutputExePath": "D:\\Releases\\PhotoEnhancer-Setup.exe"
}
```

6. Troubleshooting & FAQ

Q: How do I distribute my application to end users without trigger warnings?

Distribute the generated [App]-Windows.zip file. Instruct users to right-click the ZIP, select **Extract All...**, and double-click **Install.cmd**. This runs the PowerShell unblocker and immediately begins clean setup.

Q: Can my application include configuration files (e.g. SQLite databases, JSON settings)?

Yes! Choose **Entire application folder** mode. Place your default configuration files in the folder alongside your .exe. When users install the app, the exact folder structure and all nested files are preserved.

Q: Why does the setup installer default to %LocalAppData%\Programs?

Microsoft guidelines strongly recommend modern desktop software (like VS Code, Discord, and Slack) install in %LocalAppData%\Programs. This allows seamless auto-updating, requires zero administrator rights, and ensures your users never get stuck behind corporate UAC permission prompts.

Q: Can I run the installer silently in unattended enterprise deployments?

Yes, the setup stub engine supports unattended silent installs and uninstalls using the `--silent` or `/s` switch:

```
MyApp-Setup.exe --silent
Uninstall.exe --uninstall --silent
```

Support & Community: Simple Installer Builder is developed and maintained as an open-source productivity utility. For updates and downloads, visit the official project page.