sfxPackager v2.2

User’s Guide

This guide is intended to show you how to build install packages for Windows using sfxPackager.

The latest release and source code is available at either:
https://sourceforge.net/projects/sfxpackager/
...or...
https://github.com/keelanstuart/sfxPackager

Intro

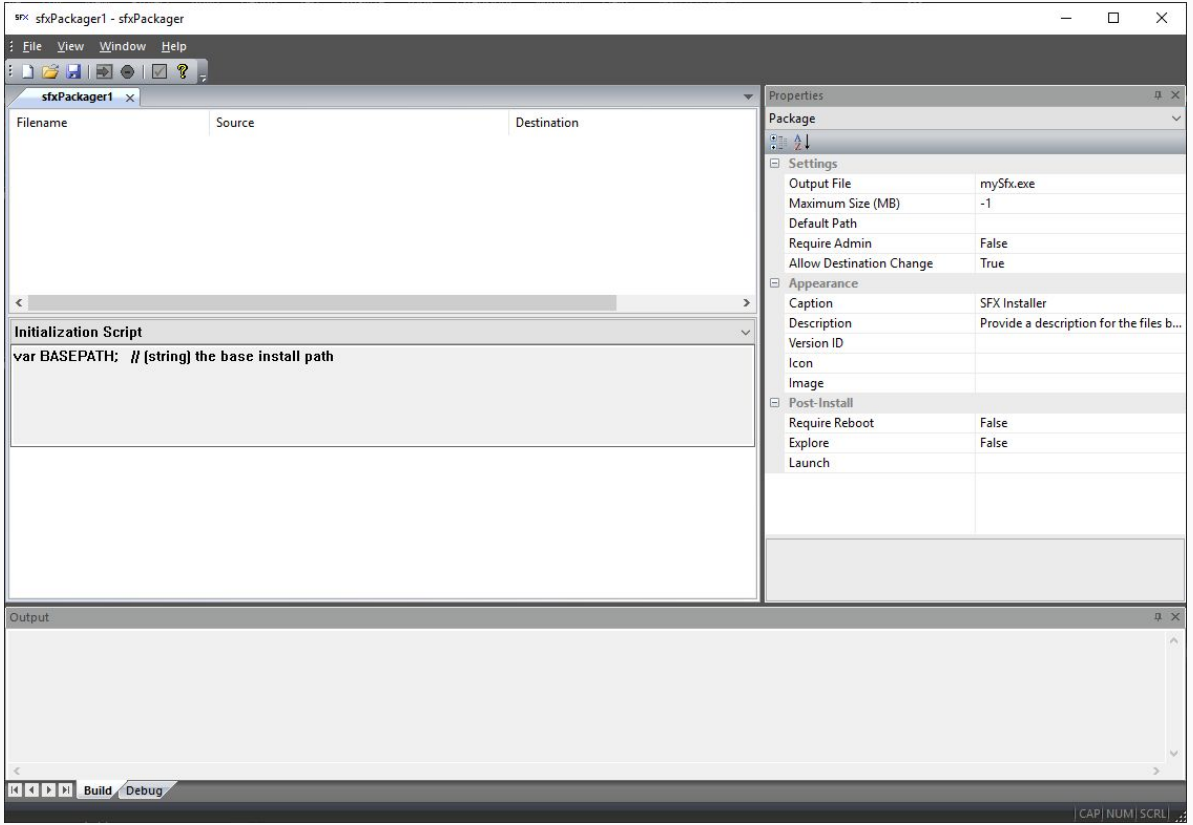
sfxPackager was designed to help you create installers for Windows platforms quickly and easily. It integrates a simple JavaScript interpreter as well as other features to customize the look and behavior of your setup.

Follow along with this tutorial to learn how to accomplish common (and some uncommon) tasks.

Project-based development

sfxPackager saves and loads files with the extension .sfxpp - short for “sfxPackager Project”.

The idea is that you build your project once and then when you need to update your installer, you load your sfxpp, click the build button, and voila! A new setup exe is generated for you.

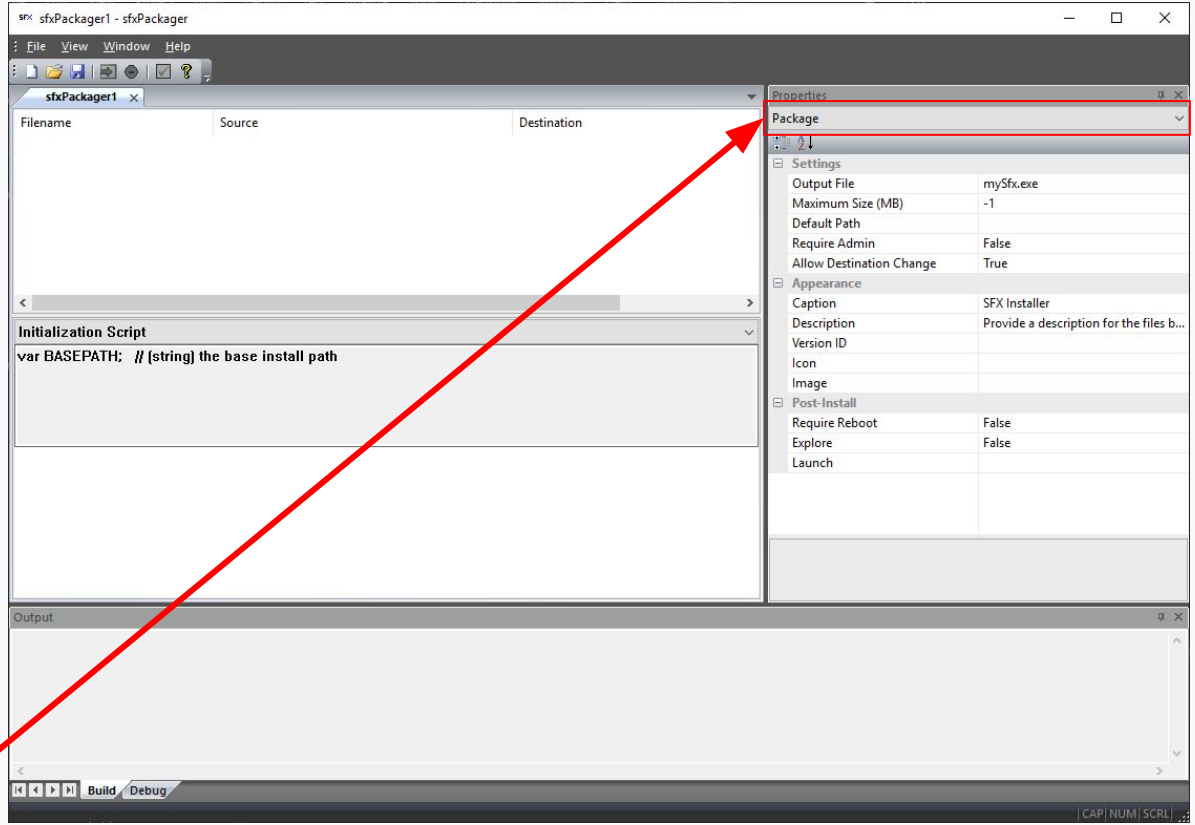


Setting Types

Settings for your project are divided into three parts:

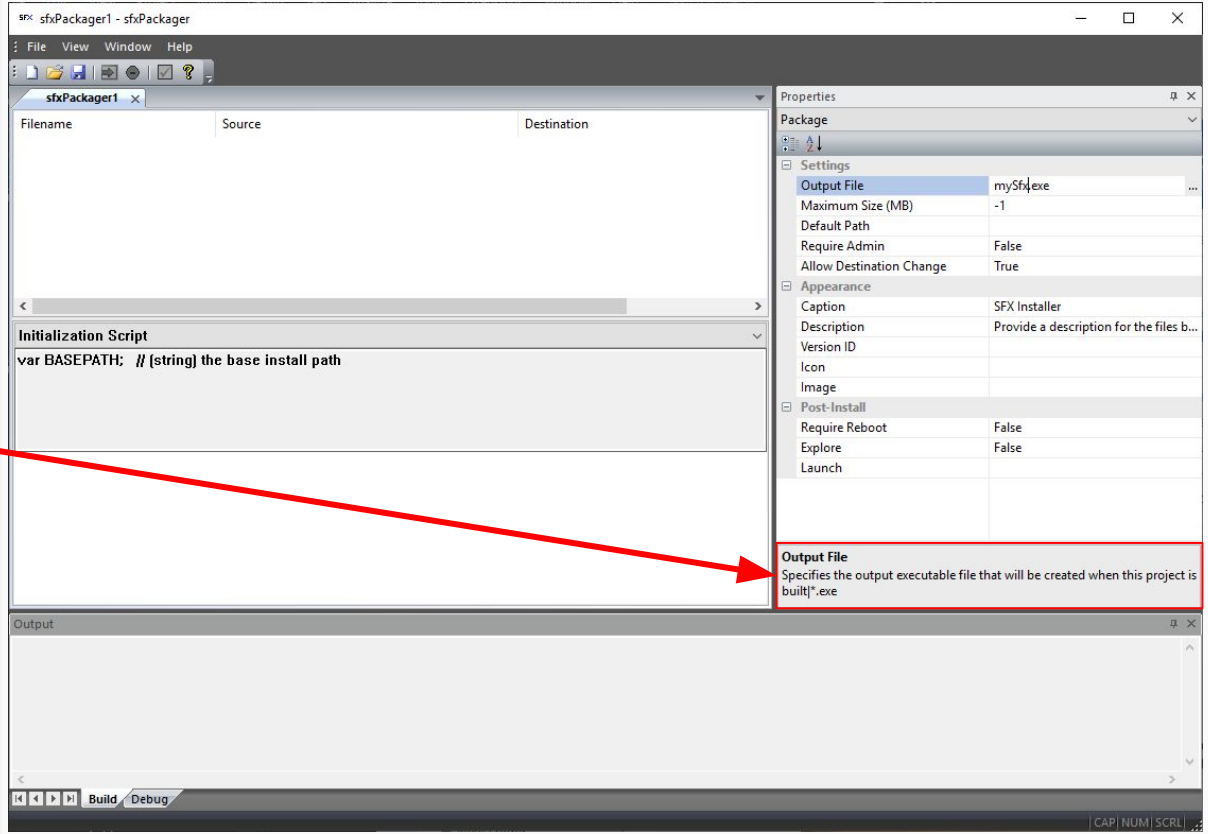
- **Package** settings that are applied to your installer as a whole, such as the output filename, window icon, image, etc.
- **File** settings for each entry in your project, such as the source and destination
- **Settings** settings, which are for the packaging application itself

You can change which setting type you are viewing with the dropdown, here...



Online Help

You can always see what any particular setting does by looking at the help text displayed at the bottom of the properties window

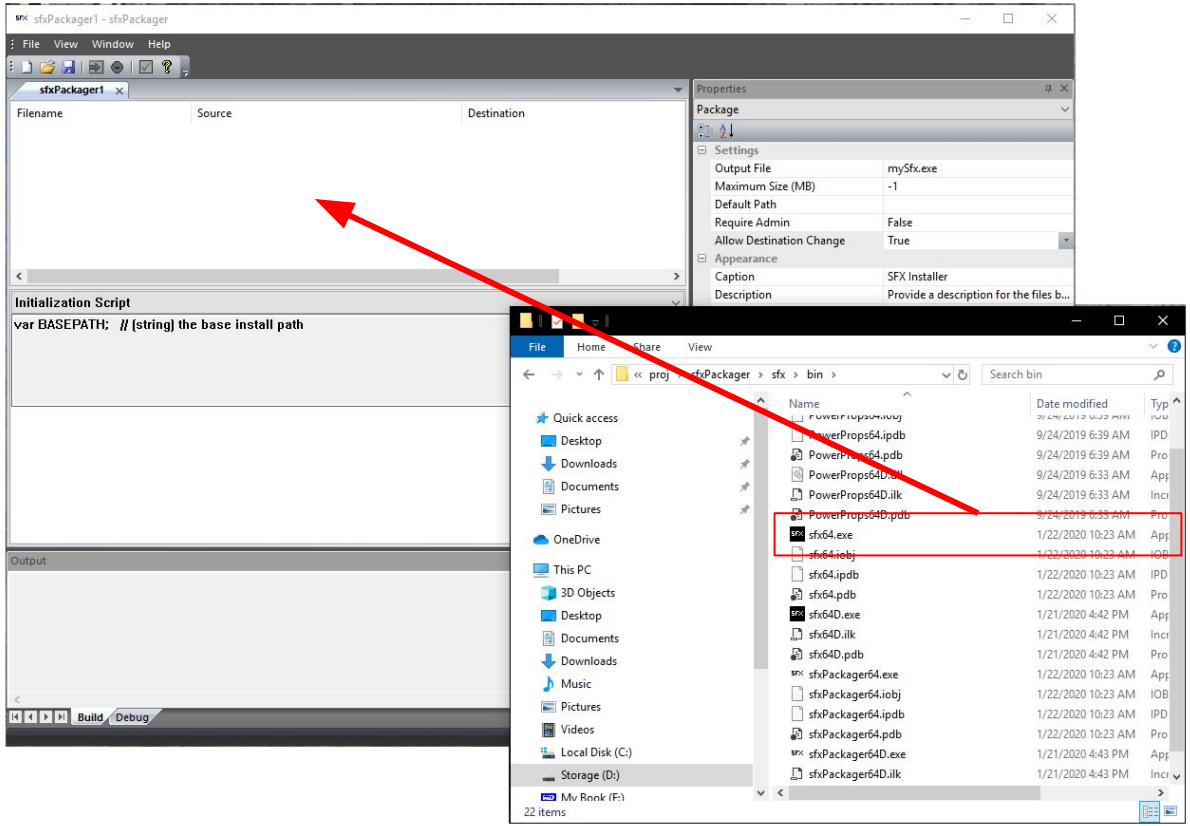


Project Settings

- **Output File**: This is the path to the .exe that will be generated when you build your project
- **Maximum Size (MB)**: sfxPackager can split your setup into multiple files no larger than the size (in Megabytes) given here. Each time one finishes, it will ask for the next file in order
- **Default Path**: This is the path that is first given as the install location to the user
- **Require Admin**: If you need administrator rights to install your package, set this to True and the user will be prompted to elevate permissions if the setup was not run with them already
- **Allow Destination Change**: Don't want to allow your users to change where they can put your packaged files? Fine; set this to True
- **Caption**: Sets the caption on the setup window
- **Description**: This is displayed in the welcome window. It can contain plain text or HTML content -- optionally, it can reference a file that contains your content
- **Version ID**: You can either place text here or the path to an .exe file, from which the version number will be extracted
- **Icon**: The path to an .ico file that will replace the default icon on the setup window frame
- **Image**: The path to a .bmp-format image file that will replace the default setup window image
- **Require Reboot**: Set this to True if your setup should prompt the user to reboot upon completion
- **Explore**: Set this to True if the setup should ask the user if they want to browse the install path upon completion
- **Launch**: Specifies a file that will be opened upon completion. The type of file will determine how it is opened and is dependant upon the end user's system configuration. Executable files will be run, HTML files will open in the default browser, etc.

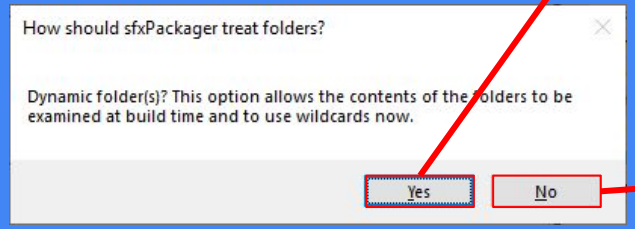
Adding Files

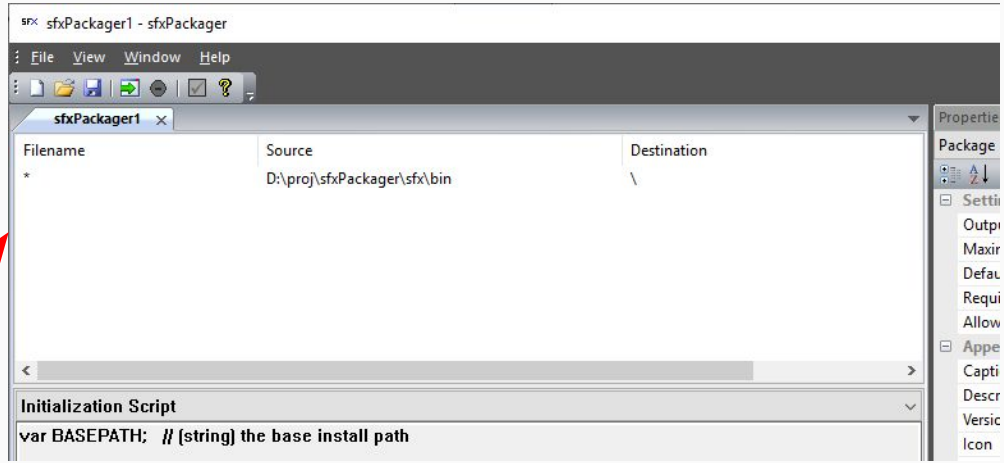
Adding files can be done in one of two ways: either by dragging files or directories to the file panel or by right-clicking the file panel and selecting “Add New File”.

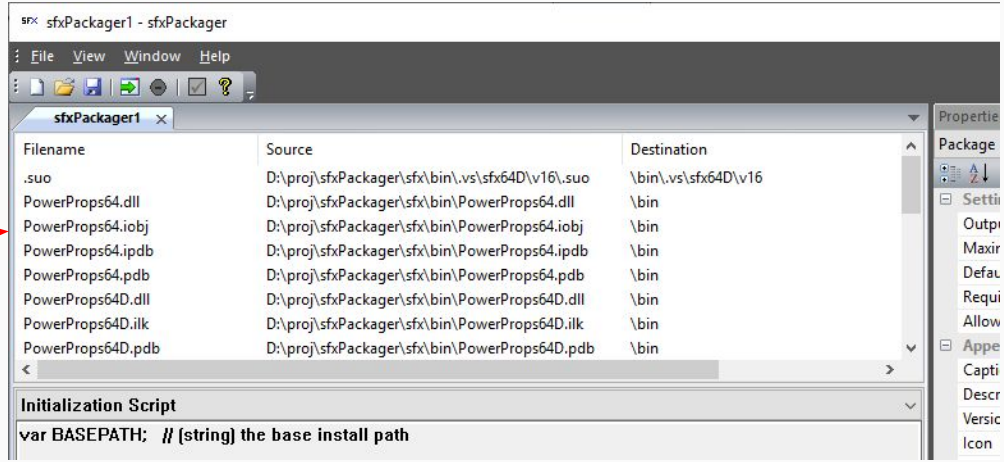


Adding Directories

If you add directory to your project, you will br prompted to either add all files now or to evaluate them when the project is built. This can help eliminate sfxpp changes when your new build ships different files.





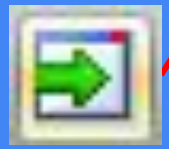


File Settings

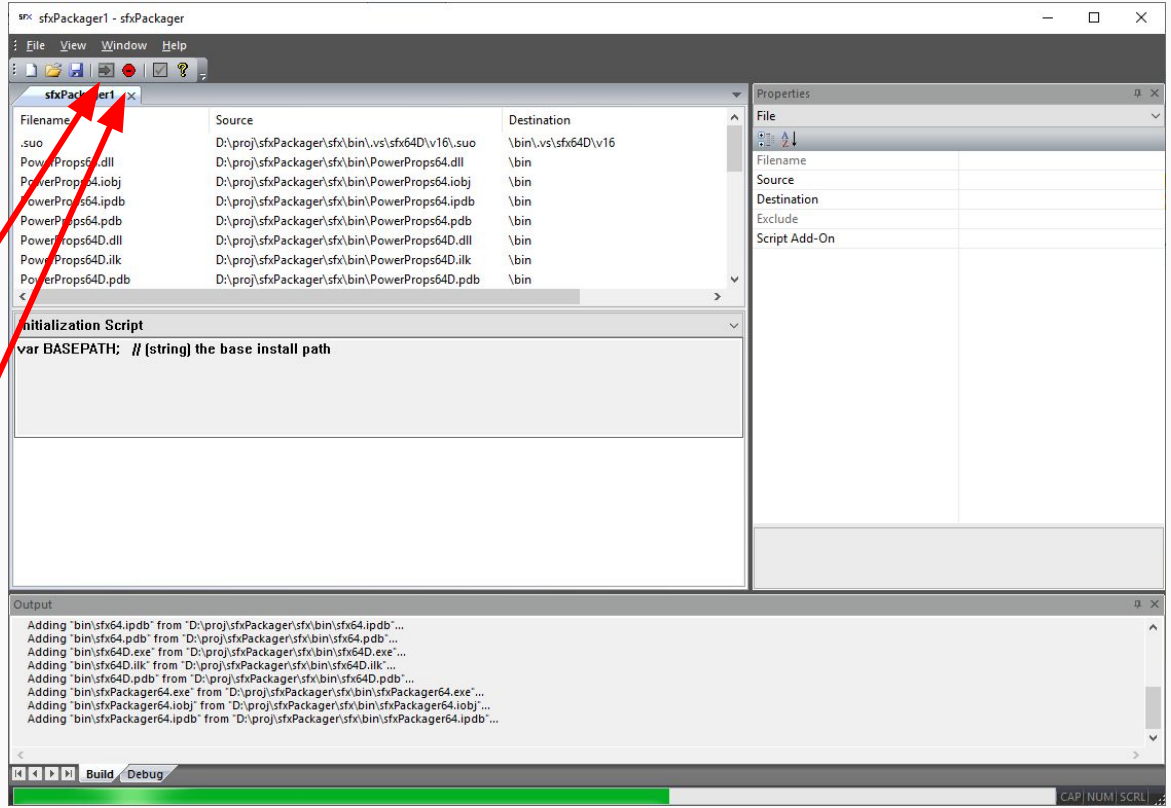
- **Filename**: The name of the file that will be installed. Some notes:
  - May contain wildcards
  - The filename can, if it is specific (i.e., does not contain wildcards), be different than the source filename, effectively renaming it when installed
- **Source**: The source location of the files to be installed; can be either a directory or a single file
- **Destination**: The absolute or relative path where the file(s) will be installed.
- **Exclude**: If you provide wildcards, you can exclude files with this semi-colon delimited list of file specifications (wildcards, too)
- **Script Add-On**: Covered later in more detail, but you can provide a JavaScript snippet that will be appended to the Per-File script and run for this file or group of files

Building Your Project

Once you have added all the files required for your installation, click the Build button to begin the build process, after which you should see an .exe at the location provided in your Package Settings



You may stop an active build at any time by clicking the Cancel button

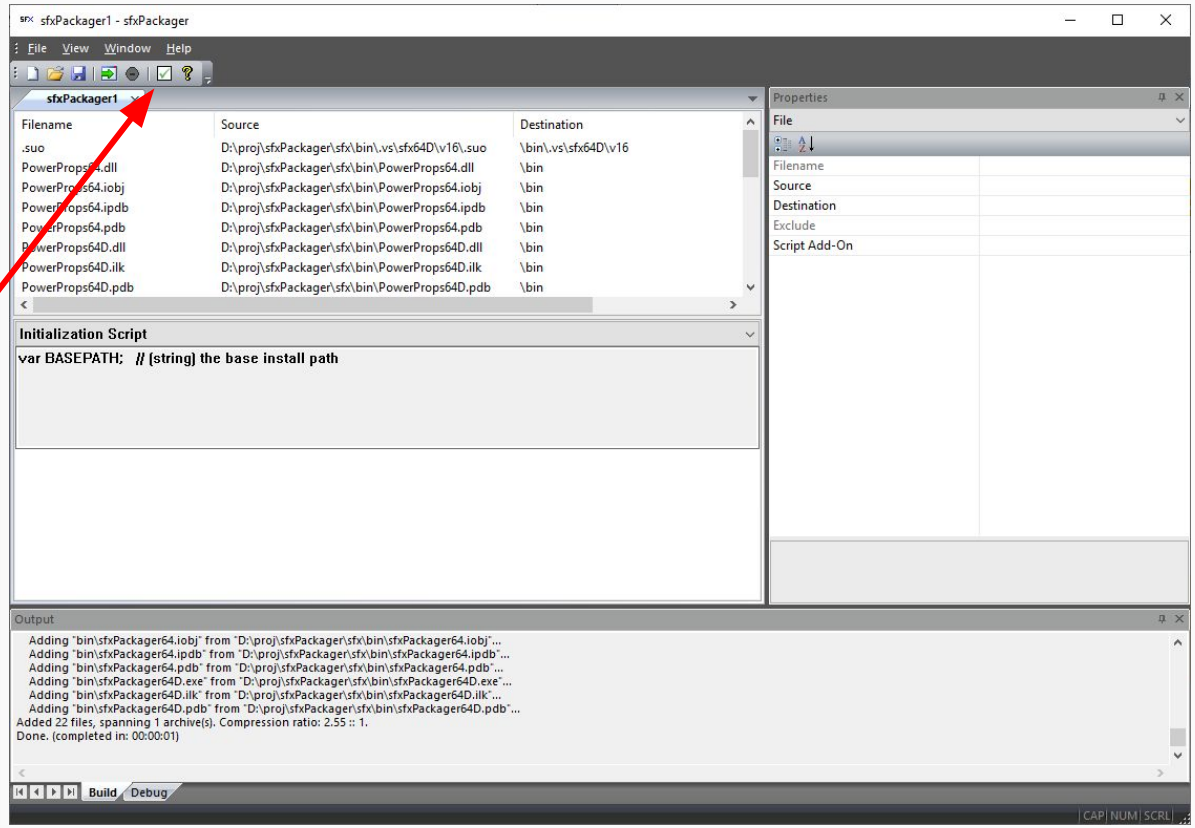


Testing Your Project

After a build has been successfully completed, you may test it by clicking the Test buton.

The Test button will only appear once the exe file exists.

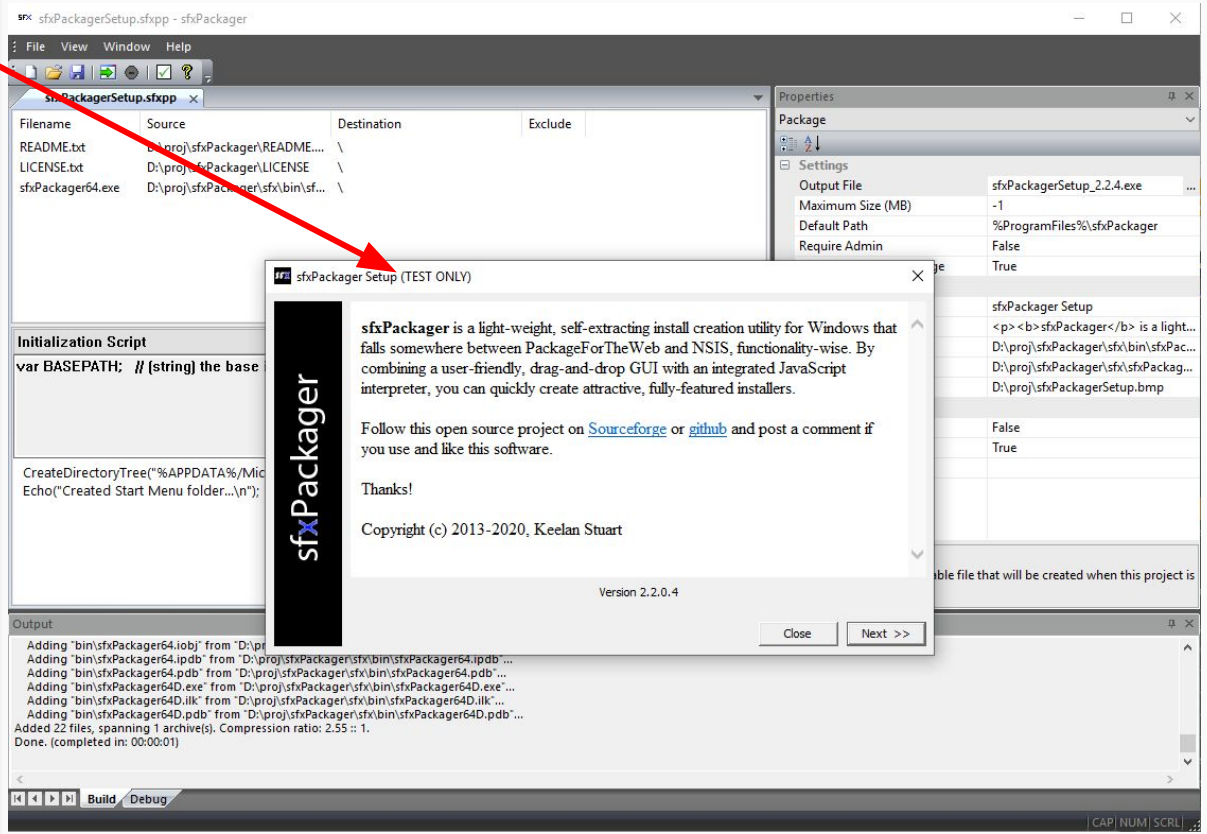
Your installer will be run with the “-testonly” command line parameter, which will only simulate the actions the installer would take if run normally.



(TEST ONLY)

If you see this text in the caption of your installer program, you know that no actual file operations will take place.

Test only mode is useful for making sure your files will go where you expect and to see that your scripts are functioning as expected.



Important Note

When supplying almost ANY filename, you can embed either **Environment Variables** or **Registry Values** by bracketing your text in ‘%’ or ‘@’ characters, respectively.

So, to set the default install location for your files to directory under *C:\Program Files*, say, *MyProgram* you can use the Environment Variable “ProgramFiles” in your path like this: “%ProgramFiles%\MyProgram”. To see what variables are available, hit Win+R and run “cmd” to open a command prompt, then type “set” and press enter.

To embed a system Registry Value a similar procedure is followed. Give a registry path like “@HKEY_CURRENT_USER\somePath\SomeKey@”. To view registry keys, hit Win+R and run “regedit”.

Scripting

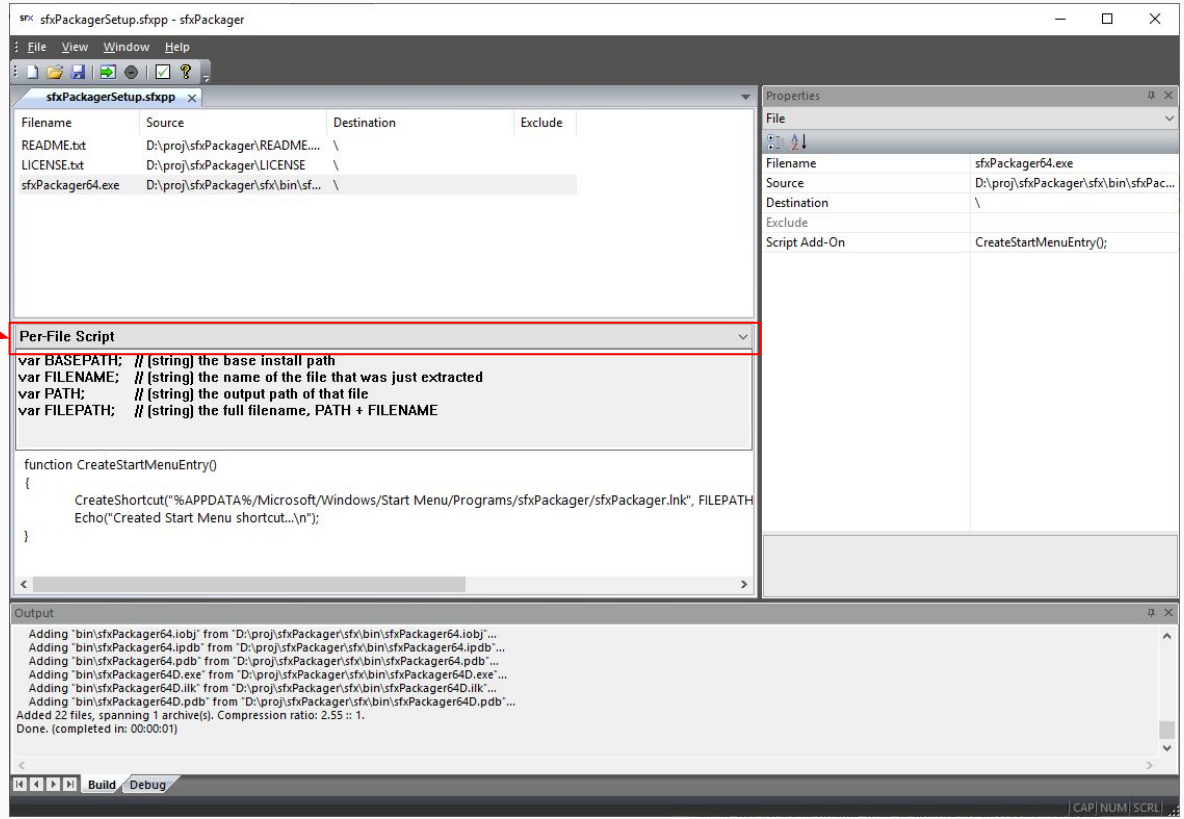
sfxPackager includes a JavaScript interpreter and some embedded functions that allow you to extend the capabilities of your installer. There are three events that cause JS code to be run: **Initialization**, **Per File** Installed, and **Finalization**.

Note: this guide is not intended to teach JavaScript programming. If you do not know the language, you may want to skip this section.

Scripting

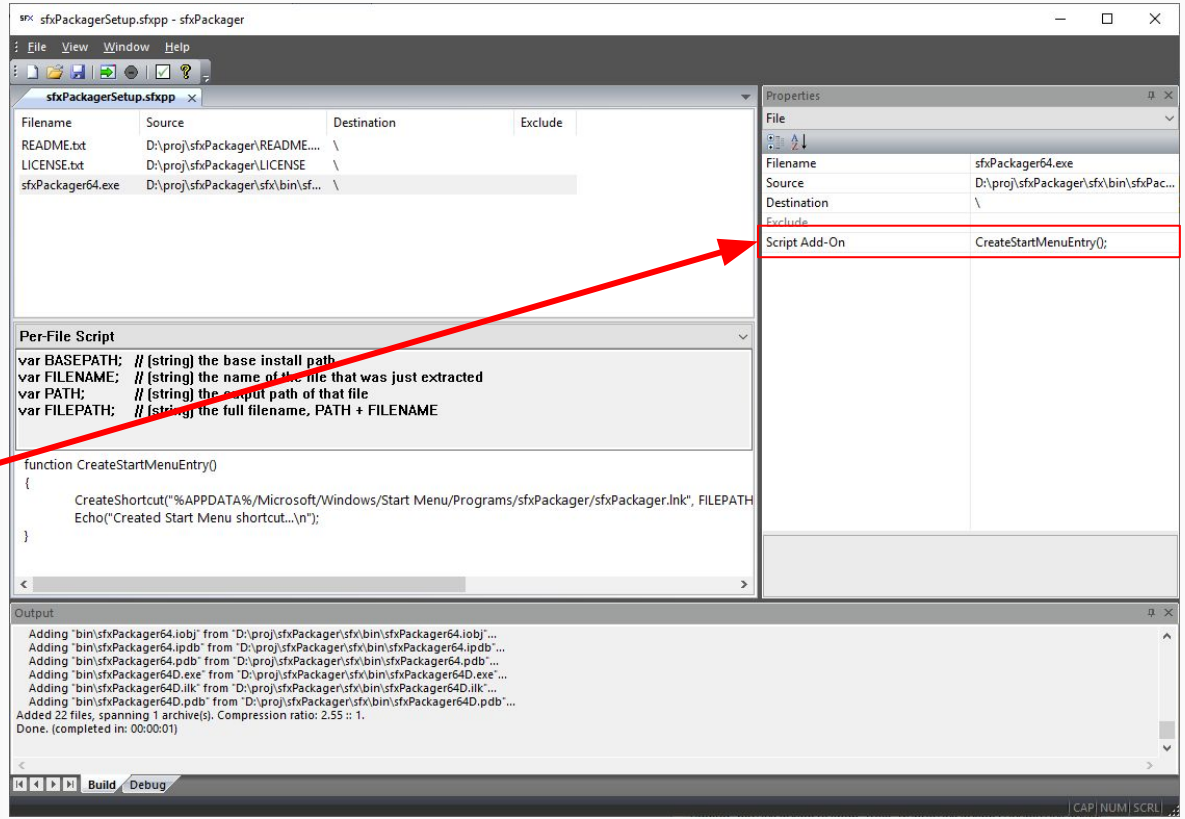
Select the type of script you wish to edit here. The corresponding code will then appear in the code editor.

Above the text editor itself is a list of system-defined, read-only global variables that are provided for that script type that you may use as you wish.



Per-File Scripts

There is only one Per-File script available to you. In order to run special, different code for multiple groups of files, it is recommended that you write functions in the Per-File Script and call them from the Script Add-On File Property.



Built-In JavaScript Functions

- AbortInstall()
- CompareStrings(str1, str2)
- CopyFile(src, dst)
- CreateDirectoryTree(path)
- CreateShortcut(file, target, args, rundir, desc, showmode, icon, iconidx)
- DeleteFile(path)
- Echo(msg)
- FileExists(path)
- GetGlobalInt(name)
- GetExeVersion(file)
- IsDirectory(path)
- IsDirectoryEmpty(path)
- MessageBox(title, msg)
- MessageBoxYesNo(title, msg)
- RegistryKeyValueExists(root, key, name)
- RenameFile(filename, newname)
- SetGlobalEnvironmentVariable(varname, val)
- SetGlobalInt(name, val)
- SetRegistryKeyValue(root, key, name, val)
- SpawnProcess(cmd, params, rundir, block)

Thanks!

Contact me:

Keelan Stuart

keelanstuart@gmail.com
+1.321.325.0329