AhkSetup 2.0 - short documentation

1. Build the setup executable

- Inside the windows command line, navigate to the ahksetup directory
- build a setup executable by using the ”build” command:

  build [source] [-li license|-gnu_gpl] *[-d destination] [-lang language]*

Source	Any file from any valid source directory. (See *2. Valid source directories*)
-li license	By specifying -li the setup will show a custom file as license text. **license** must be a textfile containing the applications license text. (See *3. License files*)
-gnu_gpl	By specifying –gnu_gpl the setup will use the GNU General Public License as license text.
-d destination (optional)	Specify **destination** to use a custom output destination for the setup executable. The output filename must be included! If ommited, the standard output is: %SourceDir%\%AppName% Setup.exe
-lang language (optional)	**language** determines the setups language. Can be “EN” or “DE” Default is “EN”

2. Valid source directories

- A valid source directory must contain a file called “appinfo.ini” with a section [AppInfo]
- All filenames in any AppInfo keys must be relative to (and inside) the source directory!
- The following AppInfo keys are obligatory:

AppName	The application name
AppVersion	The major application version of the current release
AppUpdateVersion	The update version of the current release
AppAuthorName	The name(s) of the application’s author(s)

- The following AppInfo keys are optional or have default values:

AppID	The unique application identifier. This value is used to determine existing installations and to adress registry keys. AppID must always stay consistent between several releases, while AppName can change. Default is **AppName**
AppAuthorEmail	The authors email adress or a support contact email adress for the application.

AppWebsite	The authors website or the application website/repository/…
AppChangelog	A file containing a changelog that will be available in the setup.
AppIcon	A .ico file as custom setup icon. Default is the AhkSetup icon
AppPortability	A number (0-2) that determines the portability of the setup. 0 = only standard installation, no portable installation 1 = no standard installaton, only portable installation 2 = both are possible, setup will ask the user (See *4. Portable installation*) Default is 0
AppStdInstall	The default name of the program directory in %A_ProgramFiles%. The final program loation can still be changed by the user. Default is **AppName**
AppStartMenu	The name of the application's start menu folder, which contains the application links. Default is **AppName**
AppFileTypes	A .ini file containing a declaration of all used and modified filetypes, associated with the application. (See *5. Filetypes*)
AppUninstFiles	A file inside the source directory. This file must contain a linebreak-delimited list of other (absolute) file paths. All lines inside the list must have a preceding "F:" (file) or "D:" (directory) tag to specify wheter a file or a folder will be removed. The list can be altered by the application anytime. Upon unistalling, all files specified by this list, will be removed. (This is only useful for files, that the application creates outside of the program folder, since the program folder will be uninstalled anyway.) This file will be created automatically with an unique name if ommited.
AppUninstReg	A file inside the source directory. This file must contain a linebreak-delimited list of registry keys. The list can be altered by the application anytime. Upon unistalling, all registry keys specified by this list, will be removed. (This is only useful for registry keys, that the application creates outside of its software key, which is HKLM\Software\%AppID%) This file will be created automatically with an unique name if ommited.
AppExtraInit	A .ahk file that will be included in the install routine. This can be used to do any further initialization. (See *6. Further initialization*)
AppUpdateRemove	A file inside the source directory which contains a linebreak-delimited list of files and folders (relative paths) that have to be removed from an already existing program folder before installing an update. All lines inside the list must have a preceding "F:" (file) or "D:" (directory) tag to specify wheter a file or a folder will be removed. This can be used to get rid of obsolete files of older application versions when a newer version is installed.

3. License files

A license file can be any textfile. The following variables will be replaced by their respective content upon build process:

%AppName% **%AppVersion%** **%AppUpdateVersion%**

%AppAuthorName% **%AppAuthorEmail%**

4. Portable installation

While a normal installation will support a custom InstallDir and will also install Software-, ApplicationPath-, Uninstall- & FileType-keys to the registry and will be able to create application links on the desktop and inside the startmenu, a portable installation does neither of it and simply extracts the program folder to the setups directory.

Make sure that you don't enable portable installations for programs that rely on the registry.

5. Filetypes

Filetypes that are associated to the application can be declared with a .ini file, where each section represents a filetype, following this pattern:

[*FileType1-Key*]

;--------- Filetype obligatory info ----------

Type_Name=*Displayed name of the filetype*

Type_Extension=*filetype extension (no preceding dot!)*

;--------- Filetype optional info ----------

Type_Icon=*filetype default icon (optional, .ico file, relative path)*

Type_NewFile=*template file for new files of this type (optional, must be of the declared type)*

;------- Context Menu ---------

Default=*Default Entry-ID*

Menu_*Entry1-ID=Entry1-Name*,*Entry1-Cmd*

Menu_*Entry2-ID=Entry2-Name*,*Entry2-Cmd*

Menu_ …

[*FileType2-Key*]

…

If a filetype is already existing on the users system, default icons and default context menu entries can’t be changed. However, new context menu entries can still be added.

Example 1: Adding a new context menu entry to .exe files:

```
[exefile]
Type_Name=Application
Type_Extension=exe


Menu_newaction=New Action,"%1" ;Simply run the executable
```

Example 2: Adding an own filetype (called “Example-File”) associated to the application.

```
[myAppExampleFile]
;--------- Filetype -----------
Type_Name=Example-File
Type_Extension=exmp
Type_Icon=example_icon.ico
Type_NewFile=Template.exmp
;------- Context Menu ---------
Default=Run
Menu_Edit=Edit file,"notepad.exe" "%1"
Menu_Run=Run with myApp,"myAppStart.exe" "%1"    ;pass the file to myAppStart.exe
                                                 ;(where AppPath has been installed
                                                 ;by the setup previously, since
                                                 ;myAppStart.exe was specified as
                                                 ;source file)
```

6. Further initialization

The .ahk file specified in AppExtraInit, can be any normal .ahk script.

The script will be executed after the main installation is done (files are all extracted, registry keys written, but no links are created). Furthermore, the script can use a set of variables. (Which should be treated as readonly-variables, since changing the values may result in faulty installations.)

Available variables:

CONST_SETUP_TITLE	The setup title. (A combination of name and version) "%AppName% %Appversion%"
CONST_SETUP_APPNAME	AppName key
CONST_SETUP_APPID	AppID key
CONST_SETUP_APPVERSION	AppVersion key
CONST_SETUP_APPUPDATEVERSION	AppUpdateVersion key
CONST_SETUP_STD_FOLDER	AppStdInstall key
CONST_SETUP_APPSTARTMENU	AppStartMenu key
CONST_SETUP_APPWEBSITE	AppWebsite key
CONST_SETUP_APPAUTHORNAME	AppAuthorName key
CONST_SETUP_APPPORTABILITY	AppPortability key
CONST_SETUP_APPWEBSITEAVAILABLE	If **CONST_SETUP_APPWEBSITE** is set = 1 (true) is not set = 0 (false)
CONST_SETUP_APPCHANGELOGAVAILABLE	1 (true) means, that %A_Temp%\changelog.txt contains the changelog text. 0 (false) means, that no changelog is available.
CONST_SETUP_APPEXE	The source file, that the setup will run after its completion. (if run option is checked)
CONST_LICENSE_TEXT	The plain license text.
AppExistingInstallation	If the setup found an existing installation, this will be set to 1 (true), else 0 (false) Therefor, 1 also means that the setup is currently executed as an Update
ExistingInstallationType	If **AppExistingInstallation** is set to 1, this either contains: 0 = normal installation or 1 = portable installation
AutoUpdate	Specifies with 1 (true) or 0 (false), if the setup was automatically started as update by the application.
AppCurrentUpdateversion	Holds the AppUpdateVersion of the currently installed application version.
AppCurrentInstallDir	Holds the installation directory of the currently installed application version.
SetupTypeNormal	Either 1 (true) or 0 (false). Specifies if the user has chosen a normal installation. (Not for updates, since an update won't give the user a choice. Use **ExistingInstallationType** for updates!)
SetupTypePortable	Complementary to **SetupTypeNormal**
LANG_*	Any text from the current language package.

7. Run setup as update

To run a setup file as update, the parameter “AutoUpdate” must be passed and the working directory of the setup must be set to the directory that is holding the program folder.

A common way is to move the update setup to the temp-folder and start it there, to avoid conflicts with the new installation.

Example:

```
;…
;download the update setup file as “Update.exe”

;copy the setup to temp-folder
FileCopy, Update.exe, %A_Temp%/Update.exe, 1

;run setup as update
Run, %A_Temp%/Update.exe "AutoUpdate", %A_ScriptDir%/..

ExitApp
```