DOC-GUI-APPLICATIONS

RICHARD KOCH

1. GUI Applications

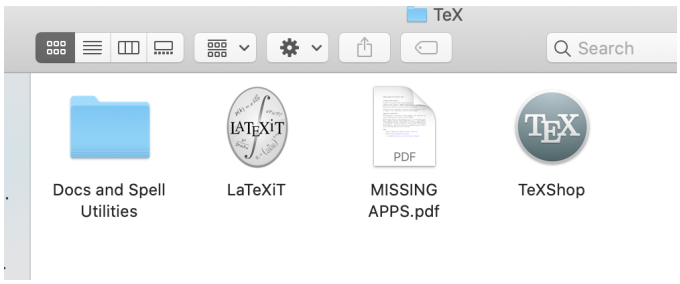
This is the portion of the TUG documentation which deals with making the GUI Applications package.

2. OverView

This package installs TeXShop, and LaTeXiT in /Applications/Utilities. It cannot contain Bibdesk and TeX Live Utility because those are not notarized, but it has a pdf file listing the URL's for these programs. It cannot contain cocoAspell for the same reason, but again it contains a URL for that spell checker. It also installs brief documentation for some features in MacTeX. The programs LaTeXiT and TeXShop must be added to the apps folder. Then buildPackage.sh moves them to root/Applications/TeX, which becomes the source for the install package.

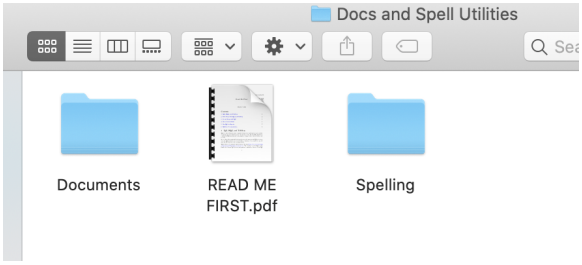
3. Graphic View of Installation in /Applications/TeX

In 2018, we revised the look of /Applications/TeX to make it much cleaner. At the top level, we see the main GUI applications and one folder. This folder contains a documents folder, a spelling folder, and a READ ME explaining how to get started working with TeX. See the illustration below.

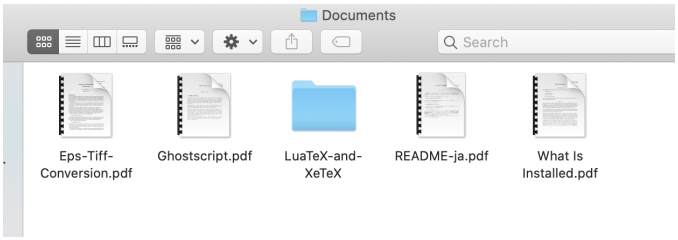


Date: March 12, 2020.

Below is a picture of Docs and Spell Utilities:



Here is a picture of the contents of the Documents Folder:



4. Programs in Apps

To save space, the "apps" directory does not contain documentation or GUI applications. Thus before building, it is necessary to add Applications to this directory. The following is a list of these applications, and of web sites where the current versions can be obtained. Note that currently we only install LaTeXiT and TeXShop, but adding the others to the folder will do no harm.

- TeXShop, available at `http://pages.uoregon.edu/texshop/`
- BibDesk, available at `http://bibdesk.sourceforge.net/`
- LaTeXiT, available at `http://chachatelier.fr/programmation/latexit_en.php`
- TeX Live Utility, available at `http://mactlmgr.googlecode.com/`
- cocoAspell, available at `http://people.ict.usc.edu/~leuski/cocoaspell/home.php` and `https://github.com/leuski/cocoAspell`.

5. Building

The first step in building a new package is to update all the supplied programs. TeXShop, Bibdesk, LaTeXiT, and TeX Live Utility all use the Sparkle mechanism to update, so it suffices to read these programs and select the "Check for Updates..." menu.

Then update the documentation. Note that source code for the documentation is in the TeX-Source folder.

Finally, build by

```
sudo sh buildPackage.sh
```

6. Documentation

Currently, we provide the following documents:

- READ ME FIRST: This document is designed to help new users begin using the system
- LuaTeX-and-XeTeX: This document was added in 2010 to show users how to use system fonts in LuaTeX, and remind them that XeTeX has the same ability.
- Eps-and-PdfLaTeX: This was added to explain that TeX Live 2010 automatically converted eps files to pdf files when running pdfTeX.
- What is installed: An explanation of what is installed and where. Particularly useful when people want to remove things.

7. Building the Final Install Package

Applications in this package must be signed and notarized, and must adopt a hardened runtime. This is not something we do; the authors must provide versions with these features. As this document is being written, only TeXShop has complied.

The install package is constructed using scripts in the folder "CreateInstallPackage." This folder also contains Resources and Scripts which will become part of the package.

8. Tricky Points

The following considerations had to be done just once, but are mentioned in case revisions are needed.†

GUI-Applications is the most tricky install package to build because by default it doesn't put applications in their intended spot, but instead searches for a random copy of the application and updates that. We carefully turn this feature off. This had to be done with the graphical package maker, and now also with the command line construction of the install package.

Consider the script createdistplist. This script is almost exactly like pkgbuildscript below, except that instead of outputting an install package, it outputs "dist.plist", a plist description of the installation. Run this script:

```
sh createdistplist.sh
```

The result will be a new file, "dist.plist".

Edit this file by hand. Very close to the top, find the entry with key BundleIsRelocatable. Change the value of this entry from true to false.

9. Straightforward Steps

Now examine the script for the command GUI-ApplicationsStart.pkg. Notice that this script has a flag which reads the file dist.plist. Build GUI-ApplicationsStart.pkg by

```
sh pkgbuildscript.sh
```

This script reaches up a level to find the root folder containing the package material. Next create GUI-Applications-Temp.pkg by running a second script

```
sh productbuildscript.sh
```

Finally, created a signed package GUI-Applications.pkg by running the script

```
sh signPackage.sh
```

This script contains my Apple signing identity; future builders will first have to join Apple's Developer Support and get their own signing identity.

The CreateInstallPackage folder contains two other files and one other script. The files are requirement.xml and distribution.xml, and the script is create_distribution.sh. In future years, it may be enough to edit distribution.xml by hand. This file contains the GUI-ApplicationsStart.pkg name, and references to the background image, Welcome dialog, ReadMe dialog, and License dialog. It contains the identifier of the package, currently "org.tug.mactex.gui2019b." And it contains the restriction that the package can only be installed on machines running system 10.12 or higher.

But if necessary, the file can be recreated as follows. The system requirement is contained in requirement.xml. Edit this first. Then run the script create_distribution.sh. This script provides some information in distribution.xml, and outputs a sample distribution.xml file. But other items must be added by directly editing the file.

10. Notarizing the Final Install Package

Scripts to notarize GUI-Applications.pkg are in the folder "NotarizePackage" inside "CreateInstallPackage." The actual GUI-Applications.pkg created above can be left where it is; it need not be moved to the new folder.

To notarize,

```
sh sendnotarizerequest.sh
```

There will be a delay when nothing is printed in Terminal as the package is uploaded to Apple. When the upload is complete, a message of the form

```
RequestUUID = 96f0932f-b50f-4d73-ad65-1220fd9c8efe
```

will be printed. This magic number can be used to discover what went wrong if notarization fails, so I recommend copying it to the file requestUUID.tex. There will be a pause of about ten minutes and then a notification from Apple and an email from Apple will arrive. If all is well, this message says that notarization succeeded. To add the certificate to the install package

```
sh stapleresult.sh
```

The package GUI-Applications.pkg is then ready for uploading and release.

In case of failure, open the script detailedinfo.sh and notice that the magic UUID number is part of this script. Copy and paste the new number from requestUUID.tex to this script. Then run it.

```
sh detailedinfo.sh
```

This command will return a web url. Use Safari to go to that url, where a very detailed error report will clearly state what went wrong.

The detailedinfo script runs a command named xcrun which communicates with developer support. Apple has adopted two-factor authentication security for such communication, but this security is not very convenient when running shell scripts. So developers can log into their Apple developer account and assign a password to xcrun and a few similar commands. Then when the script runs, typing this password is enough to run the command. Note that the detailedinfo script contains my account name and password. When someone else makes these packages, they will replace this information with their own personal information.