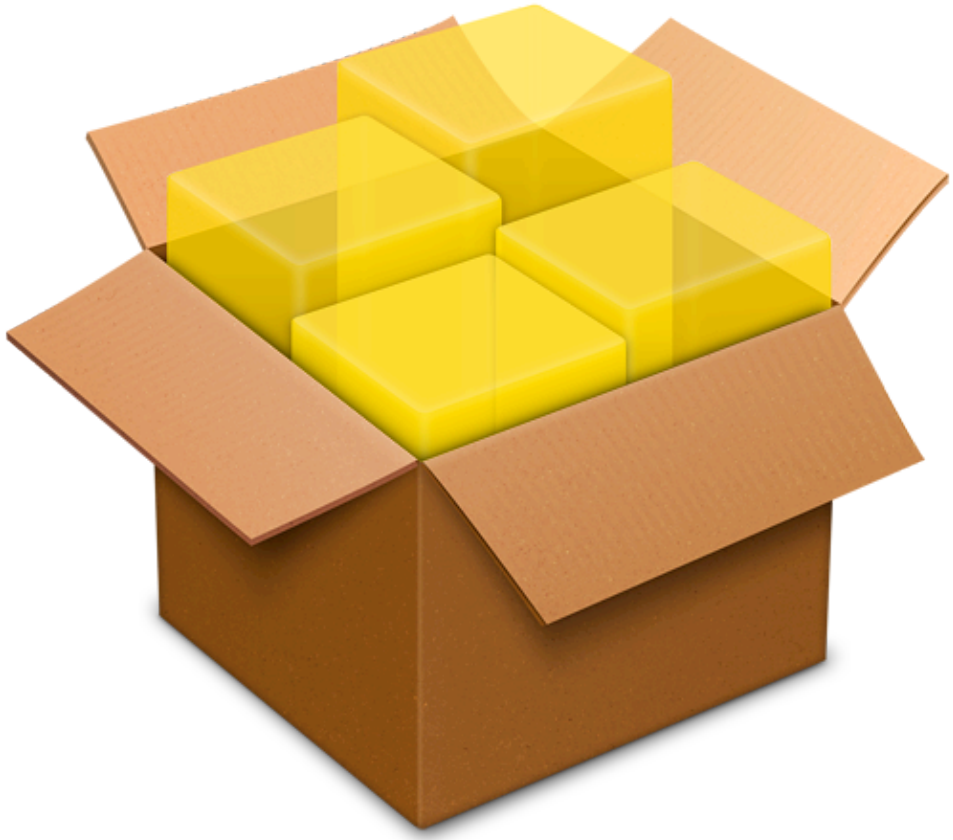
Practical Packaging

Matt Willmore
College of Arts & Letters
University of Notre Dame

mwillmor@nd.edu • @redrobot

Agenda

- Why package?
- Package uses
- Tenets of packaging
- Packaging tools
- Demo time!
- Resources
- Q&A



About Me

- IT Support Engineer @ Notre Dame since July 2010
  - Supporting faculty/staff in College of Arts and Letters
  - Approx. 800 Macs, 99% individual use

Why package?

“I dislike .pkg files on Mac OS X. I don't like how they can touch any part of the system they want. Many times the installer files are useless anyway; all they do is make sure an app is installed in /Applications, for example, which I can handle myself, thank you very much.”

– some random guy online

Why package?

Portability

- Lots of variability in end user environment
  - OS version
  - Hardware specs
  - Institutionally or personally owned?
- Admin toolsets change
  - No need to rebuild installers when you change tools

Why package?

Portability

Lingua franca for OS X
(credit to Jeremy Reichman)

Details come in the installer, but .pkg and .mpkg formats have been and will be recognized and understood

Other vendors support packages because of the low barrier to entry and existing usage

Why package?

It’s easy for the user

Users : installer packages :: Admins : Server.app

Benefits and drawbacks for all parties

Users don’t have to understand the complexity within the package, just double-click and go

We the admins can engineer the desired result without changing user behavior or frustrating them

Combine with admin tools and users may not have to see anything happen

Why package?

Logic

We can make installer packages as simple or complex as desired

Logic can be added to ensure installations are successful regardless of environment

Different environments (eg. OS version, processor, HD available space, existing installation, potential conflicts)

Why package?

Forward compatability

There are changes in the future, but what?

- Architectural
- Operating system
- Management tools (Munki, Casper Suite, etc.)

Collection of files, not an app (see: InstallerVISE)

- Installer.app is free and has guaranteed longevity*

** This is Apple we're talking about, so all guarantees are null and void at their discretion.*

Package uses

Package uses

App distribution

- Useful even for drag-and-drop installers (see Firefox, Chrome, Evernote, etc.)
  - Prevents users from misunderstanding “instructions” in mounted disk image
  - Easy to install to a custom location of your choosing
- Good for custom installs
  - App + registration + settings + add to login items all at once

Package uses

Payload-free packages

My most-feared situation: having to say “OK, open Terminal.app and type **exactly** what I say…”

Payload-free packages allow you to “package” script execution

- Same portability as with app distribution
- Equally simple for users to do
- Can apply logic in either script or package (or both)
  - How is success determined?

Package uses

Combo files & scripts

Situations not addressed by payload-free or simple packaging

Example: install files in `~/Library/Application Support` for current user and set permissions

1. Install files to somewhere like `/tmp`
2. Detect current user and their home directory
3. Move files to `~/Library/Application Support`
4. `chown` to correct user

Package uses

Modular imaging

Can make use of packages with and without payloads

Tools like Munki can be used to install without a user logged in and work beautifully with packages

Remember *portability*: package once, use repeatedly

- Modular image building
- One-off installs
- ARD package installs to individuals/groups

Tenets of Packaging

(shamelessly stolen from
Gary Larizza & MacEnterprise)

Tenets of packaging

1. Don’t assume install method

Build your package for installation not only across multiple architectures, OS versions, etc. but also *deployment methods*

- Interactive (double-click, use Install.app)
- Command-line (via `installer` process)

Non-boot volumes (eg. dual OS X installations)

- Use `$3` in shell to specify target volume

Tenets of packaging

2. Be simple

Stay away from temptations to open browser windows (looking at you, Flash Player...) post-install

Do you *have* to reboot?

Don't install more than what is necessary

As an alternative, give users an "out"

eg. Growl, Flash Player + Chrome, etc.

Try to delight the user, not bug them into hating your installers

Tenets of packaging

3. Let licensing work without the GUI

Licensed software is OK, but avoid user interaction when possible

- Especially applies to site licenses

Can it be scripted? Can I drop a key file on the system?

How easy is it for admins to revoke a license?

Tenets of packaging

4. Limit pre/post-install scripts

Only script what cannot be contained in the payload

Use native UNIX tools (**`cp`**, **`mv`**, **`ditto`**, etc.); stay away from things like **`osascript`**

Stay away from GUI scripting

Tenets of packaging

5. Be true to the OS

Test on each OS for which you plan to support

Test with and without GUI

Test on non-boot volumes

Don’t touch what’s not yours (more for 3rd party vendors)

Tenets of packaging

6. Be descriptive

Just because you don’t plan to have GUI users is not an excuse not to make it GUI-friendly

- Welcome message, read-me, description of what’s happening, etc.

Use descriptive naming

Comment pre/post-install scripts thoroughly

Packagaing tools

Packaging tools

PackageMaker

Not being developed by Apple any longer

Existing users: *can* still use but look at alternatives

New users: don’t waste your time, move elsewhere

Packaging tools

pkgbuild/productbuild

Apple's current tools for package building – all done via command line

pkgbuild creates simple component .pkg files

Use **-nopayload** flag for easy payload-free packages

Use **productbuild** for more complex distrbution-type packages (also creates .pkg files)

```
man pkgbuild
man productbuild
```

Packaging tools

JAMF Composer

Commercial packaging utility from JAMF

Easy snapshots - record files changed and grab those to form the package

Modify existing packages

Good companion for Casper Suite

http://www.jamfsoftware.com/products/composer

Packaging tools

The Luggage

Open-source project to replace PackageMaker

Uses makefiles as manifests for package creation

- Tries to solve issue of not being able to peer-review what has changed in an installer
- Examine diffs on makefiles to figure out differences

Go see Jeremy Reichman's talk "Carry on Luggage"
Friday @ 9:15am

http://luggage.apesseekingknowledge.net/

https://github.com/unixorn/luggage

Packaging tools

Packages

Free GUI alternative to PackageMaker

Same author as Iceberg – use this instead

GUI-based logic

Builds both raw and distribution packages

Cannot do Gatekeeper-compliant package signing

http://s.sudre.free.fr/Software/Packages/about.html

Packaging tools

What about Gatekeeper?

- Introduced in 10.8 as a way to alert users to unsigned packages
- Use productsign to sign with your Developer ID certificate
- Unsigned packages not an issue when not using GUI installer
    - Cannot assume we'll avoid GUI every time

```
man productsign
```

Demo time!

- `pkgbuild` & `productbuild`
- Packages

Demo

pkgbuild

```
cd ~/Desktop/Google\ Earth

pkgbuild --analyze --root ./Google\
Earth.app GoogleEarthApp.plist

pkgbuild --analyze --root ./Google\
Earth\ Web\ Plug-in.app
GoogleEarthPlugin.plist
```

Demo

pkgbuild

```
pkgbuild --root ./Google\ Earth.app --
component-plist GoogleEarthApp.plist --
install-location /Applications/Google\
Earth.app --version 1.0 --identifier
edu.nd.GoogleEarthApp.pkg GoogleEarthApp.pkg

pkgbuild --root ./Google\ Earth\ Web\ Plug-
in.plugin --component-plist
GoogleEarthPlugin.plist --install-location /
Library/Internet\ Plug-Ins/Google\ Earth\
Web\ Plug-in.plugin --version 1.0 --
identifier edu.nd.GoogleEarthPlugin.pkg
GoogleEarthPlugin.pkg
```

Demo

productbuild

```
productbuild --synthesize --package
GoogleEarthApp.pkg --package
GoogleEarthPlugin.pkg Distribution.xml

productbuild --distribution ./
Distribution.xml --package-path . ./
Installer.pkg
```

Demo

Packages

- Create raw packages for each “folder”
- Create distribution containing all raw packages

Resources

Gary Larizza – The Commandments of Packaging (archived PDF): https://notredame.box.com/s/w9tse4c2wj7chsziy7tk

Q&A

Thank you