Creating Installation Packages

MacWorld SF 2009
Session IT813

Presentation Logistics

Creating Installation Packages

- Q&A managed with Google Moderator
  - http://tinyurl.com/633v6e
  - Sign in using your personal Google account
  - Find session matching IT813
  - Submit or vote on questions for this session
- Download PDF of this presentation
  - Join Wireless MW09_PRESENTATION_DOWNLOAD
  - Open web browser for automatic redirect

About the Presenters

Who are these guys?

- Dave Pooser

  Alford Media Services:
  Manager of Information Services

- Kevin M. White

  Owner: Macjutsu, Inc.

  Author:
  Apple Certified Training - Mac OS X Support Essentials
  Apple Certified Training - Mac OS X Deployment

Agenda

Creating Installation Packages

- Why Installation Packages?
- General Installation Overview
- Creating Basic Installation Packages
- Using Automation in Installation Packages
- Snapshot Installation Packages
- Deploying Installation Packages
- Understanding the Installation Process

Why Installation Packages?

The future of system image creation

Why Installation Packages?

Building blocks for your deployment

- Simplifies complex item deployments for end users
- Guarantees consistent deployment of items every time
- Extra-handy with open source apps across multiple computers
- Easy to track and audit after installation

Why Installation Packages?

Building blocks for your deployment

- Repackage odd-ball installers to a consistent format
- Installation is easily automated
- Works with many administration tools
- Cornerstone to the modular system image creation workflow
- Super easy to create with Mac OS X 10.5 PackageMaker (part of Xcode)

General Installation Overview

Time for some definitions...

Installation Definitions

What does it all mean?

- Installation Package - “single” file containing your installation items
- Installation Metapackage - contains multiple packages, sometimes allowing the user to choose which items are installed
- PackageMaker Project - document that describes the configuration of an installer before it’s built
- Payload - the collection of items that will be installed by your installer package

Bundle-Based Packages

Nothing is as it seems

- For systems prior to Mac OS X 10.5
- Packages actually folders (bundles)
- Easy to view contents from the Finder
- Adds layer of complexity to deployment because you have to place installer package in archive container
- Receipts after installation are also bundles... more on this later

Flat Installation Packages

Flat is the new package

- Only supports Mac OS X 10.5
- Packages flattened to a single monolithic file (as far as the file system knows)
- Contents can be viewed with Flat Package Editor (also part of Xcode)
- Much easier to deploy when using non-Apple storage and transfer mechanisms
- Receipts after installation saved to new receipts database... more on this later

Creating Basic Installation Packages

Let’s build something!

Creating Basic Installers

Almost as easy as an "i" app

- Organization is used to identify your specific installation packages
- Minimum Target will determine type
  - Mac OS X 10.5 will create flat installation package
  - Mac OS X 10.4 or older will create bundle-based
- Simply drag-and-drop components into the PackageMaker interface
- Click Build to create your installation

Settings Live at Each Layer

Layers, like an onion

- Be sure to explore settings at each "layer" in the PackageMaker interface
- "Installation Product Package" defines settings for the installer whole
- "Installation Choice" defines settings for a specific installation payload
- "Installation Component" defines settings for a specific set of items

Important Customizations

Make the installer your own

- Check Requirements to ensure the payload items are appropriate
- Check Choice States if you want to manage what the user can choose
- Check Contents for proper permissions
- Use the WYSIWYG interface editor to customize the user’s experience

Using Automation in Installer Packages

Power to the packages!

Automation Basics

Creating Installation Packages

- Basic pre and post install Actions are defined similar to Automator Actions
- Any scripts can be defined at:
  - PreInstall and PostInstall
  - Preflight and Postflight (bundle only)
  - Preupgrade and Postupgrade (bundle only)
  - Per individual component (Flat only)
- “Payload-free” installers are possible

Scripting Details

Script it... script it good

- Use any shell scripting language you prefer just avoid dependencies
- Always use absolute paths and/or variables:
  - $1 - The full path to the installation package being installed; /Volumes/Projects/Testing/PackageName.pkg
  - $2 - The full path to the installation target destination; /Applications/Utilities
  - $3 - The mount point of the destination volume; / or /Volumes/External_Disk
  - $4 - The root folder for the current system, that is, /

Snapshot Installation Packages

Making packages for the lazy

“Snapshots” in Tiger

Some third-party software required

- logGen (Thanks, U of Michigan!)
  - Run before install to create a baseline image
  - Run after install to find differences
  - Output to a text file - will need some cleanup
- PackageDitto
  - Simple shell script that creates a PackageRoot from text file
- PackageMaker - use PackageRoot

PackageMaker Snapshots

Insert “Apple Magic” here

- Mac OS X 10.5 PackageMaker can automatically create packages based on changes to the file system
- Leverages fsevents framework so no laborious pre-scanning required
- Incredibly useful for repackaging those nasty third-party installers
- Aside: Check out fseventer and fslogger

Understanding the Installation Process

Don’t take candy from a stranger’s installer

How to Inspect an Installer

What's in there anyhow

- The Installer application will show the content for most installations
- Manually explore the contents
  - Bill of Materials (.bom) file contains payload list

    ```
    $ lsbom –p UGMsF path_to_archive.bom
    ```

  - Payload can be decompressed in the Finder
  - Scripts live in the Resources folder
  - Expand any flat packages with

    ```
    $ pkgutil --expand name.pkg destination
    ```

- Use the fabulous Pacifist utility

Installation Receipts

Every installer leaves a trail

- Bundle-based installation packages
  - Receipt in /Library/Receipts
  - .bom files stored within the receipt .pkg
  - Other resources stored within the receipt .pkg
- Flat installation packages
  - Receipt data stored in receipt database
  - List receipts with $ pkgutil --pkgs
  - .bom files also stored in /Libary/Receipts/boms
- $ lsbom –p UGMsF path_to_archive.bom

Installation Mainteneance

How to really "reinstall"

- Remove the receipt to reinstall!
- Bundle-based installation packages: simply delete the receipt package
- Flat installation packages:

  ```
  $ sudo pkgutil --forget package-id
  ```

- Repair Permissions ONLY WORKS with certain Apple items... don't believe me?
  - `$ diskutil repairPermissions /`
  - `$ /usr/libexec/repair_packages --list-standard-pkgs`

Deploying Installation Packages

This is the whole point, after all

Deploying Install Packages

This is the whole point, after all

- A primary reason to create an installation package is to facilitate user-based deployment
- Apple Remote Desktop 3 is a great package deployment tool, especially when using a Task Server
    - Caches installation packages to be deployed
    - Automatically installs when client available

Deploying Install Packages

This is the whole point, after all

- As part of your System Image... IT823
- Third-party System Management suites
  - JAMF Casper
  - FileWave
  - LANrev
  - LANDesk
  - PUPPET

Presentation Logistics

Creating Installation Packages

- Q&A managed with Google Moderator
  - http://tinyurl.com/633v6e
  - Sign in using your personal Google account
  - Find session matching IT813
  - Submit or vote on questions for this session
- Download PDF of this presentation
  - Join Wireless MW09_PRESENTATION_DOWNLOAD
  - Open web browser for automatic redirect

Creating Installation Packages

MacWorld SF 2009
Session IT813