PackageMaker 3: Building a Leopard or Snow Leopard Installer Package Step By Step, version 1.1
Bill Cheeseman 2010-07-31

With the advent of Xcode 3.0, Apple told us that it prefers managed installation of applications using installer packages when ancillary software must be installed, such as a shared framework. The initial PackageMaker 3 documentation went so far as to suggest that installer packages are always preferred over the traditional manual drag install. That suggestion disappeared in an early revision of the documentation, but it is nevertheless true that an installer package is sometimes the right way to go.

The following instructions for creating an installer package are rigidly step-by-step, meant to be performed in the order given. This makes for tedious reading, but based on my experience it will save you a lot of false steps when you build your first installer. This article is based on PackageMaker 3.0.4, the current version at this writing. References to documentation and other articles on PackageMaker appear at the end of this article.

These instructions are based on a reasonably common software distribution scenario: a product consisting of a Cocoa application package and separate supporting software, in this case a shared Objective-C framework to be installed in the standard location for shared frameworks, /Library/Frameworks. Both are to be distributed subject to license. For the purposes of this example, the application is the free developer utility Event Taps Testbench, and the framework is the PFEventTaps framework, both of which I wrote for distribution under my PFiddlesoft brand <http://pfiddlesoft.com/>. Event Taps Testbench is available for download there, so you will be able to test the installer resulting from this article yourself. The product requires Mac OS X 10.5 (Leopard) or newer, so it is appropriate to use PackageMaker 3 and the so-called "distribution package" format (as opposed to "meta-package" format), available for Mac OS X 10.4 (Tiger) and newer. These are said to be more powerful yet easier to use than earlier tools and techniques, so they are recommended for products that do not have to run on older versions of Mac OS X. The only significant omission from these instructions is localization; see the 2007 José Cruz article in MacTech Magazine, cited in the Sources section at the end of this article, for details on how to localize your installer package.

Note that some features of the PackageMaker 3 user interface and functionality will be different from what is described here if you are building an installer for a Mac OS X 10.4 (Tiger) product. Here, we are building for Mac OS X 10.5 (Leopard), and PackageMaker 3 will therefore automatically generate a new-style "flat package"—that is, an installer file that is not a bundle.

These instructions were developed using PackageMaker 3.0.4. Some parenthetical notes and the Miscellaneous Issues section near the end describe numerous issues in PackageMaker 3.0.4 that you should be aware of. They will presumably be fixed in a future release of PackageMaker.

Before beginning, you should have ready to hand the source payloads for the installer, in this case the release versions of the application package and the shared framework (the framework is a folder configured as a bundle, with its bundle bit turned off). Optionally, you may also have ready to hand an image file (perhaps the master file for your application icon) in an accepted format, such as png, and relatively high resolution, such as 512x512; a Welcome file; a Read Me file; a License file; a Conclusion file. You will write a couple of shell script files to use in the installer as you go.

It is not necessary to preassemble any of the source files into a structured project directory in the Finder ahead of time, at least if you are building an installer for Mac OS X 10.5 (Leopard) or newer. This is a welcome change from earlier versions of PackageMaker. The source files can be located anywhere on your computer, although it will likely be a good idea to put them all in one place so you can build the next version of your installer without revising all the links. Be sure to leave them

where they are until you have built and successfully tested the installer; otherwise, PackageMaker will not be able to find them when you open the project to rebuild the installer after making revisions.

I have found it useful to create a folder where I keep all the PackageMaker-related files I will need to build the installer package for this application, which I name the PackageMaker ETT Installer Stuff folder and place where all my source code for this product is located. You can organize your own folder any way you find comfortable, but I have a Text Files subfolder for the License file, the Read Me file and other text files that will be used in PackageMaker's Interface Editor; a Shell Scripts folder where I keep aptly named shell scripts; a Background Image folder for the png file that will form the background of the Installer window; a Code Signing Clean Room folder where I place unsigned copies of the release versions of the application and framework to be installed; and a Payload folder where I place copies of the release versions of the application and framework after they have been code signed. I keep the PackageMaker ETT Installer Stuff folder open and visible so that I can drag desired files from it into appropriate receptacles in the PackageMaker windows as I build the installer package. As I build the installer package, generated files will be placed at the root level of the Installer Stuff folder, such as the finished Event Taps Testbench installer package and the working Event Taps Testbench Installer Project file.

I. Sign Your Code

Beginning with Leopard, Apple advises to use code signing for increased security and usability. It is said to improve your users' experience only moderately in Leopard, but it will become increasingly required in future releases of Mac OS X. I am not going to explain the theory here, but only how I sign the application and framework for Event Taps Testbench. In the process of developing this methodology (back in the days of PackageMaker 3.0.1, before the documentation was complete), I corresponded with the responsible Apple engineers and learned from them that any embedded application packages inside your application package should be placed at the top level of the Contents folder in your Application, and that you should sign each of them separately. I also learned that, if the MacOS folder in your application package contains more than one executable, you must separately sign the executables that are not marked as the application's main executable in the Info.plist file. With respect to frameworks, I learned from an online mailing list discussion that the actual version folders within a framework bundle must be separately signed, such as version A and version B. All of these requirements are now documented by Apple, and all of them apply to Event Taps Testbench and the PFEventTaps framework.

To code sign my product, I use an AppleScript script application that I keep in my Code Signing Clean Room folder along with unsigned release versions of my software to be signed. The most recent versions of Xcode allow you to set up simple code signing in your Xcode project's build settings, and you can use an Xcode build script phase for more complex requirements. However, I am more comfortable using an AppleScript application. Here is my script, which you should be able to adapt to your circumstances:

```
tell application "Finder" to set cleanRoomPath to POSIX path of (container of (path to
        me) as alias)
set ETTpath to cleanRoomPath & "Event Taps Testbench.app"
set otherExecutableETTpath to ETTpath & "/Contents/MacOS/Event Taps Testbench"
set PFEntrustETTpath to ETTpath & "/Contents/PFEntrustETT.app"
set PFRelaunchETTpath to ETTpath & "/Contents/PFRelaunchETT.app"
set PFEventTapsPath to cleanRoomPath & "PFEventTaps.framework"
set PFEventTapsAPath to PFEventTapsPath & "/Versions/A"
set PFEventTapsBPath to PFEventTapsPath & "/Versions/B"
```

```
do shell script "codesign -f -s PFiddlesoft " & quoted form of ETTpath
do shell script "codesign -f -s PFiddlesoft " & quoted form of otherExecutableETTpath
do shell script "codesign -f -s PFiddlesoft " & quoted form of PFEntrustETTpath
do shell script "codesign -f -s PFiddlesoft " & quoted form of PFRelaunchETTpath
do shell script "codesign -f -s PFiddlesoft " & quoted form of ETTpath
do shell script "codesign -f -s PFiddlesoft " & quoted form of PFEventTapsPath
do shell script "codesign -f -s PFiddlesoft " & quoted form of PFEventTapsAPath
do shell script "codesign -f -s PFiddlesoft " & quoted form of PFEventTapsBPath
```

The PFiddlesoft parameter in each of the codesign commands refers to the self-signed certificate I created for myself, named "PFiddlesoft Code Signing Certificate," using the Certificate Assistant command in the application menu of Apple's Keychain Access application (Identity Type: Self Signed Root; Certificate Type: Code Signing). The identity parameter to the shell script codesign command only requires a name that appears somewhere in the name of the certificate, as long as it uniquely identifies this certificate. You can buy a certificate from VeriSign or another authority, but a self-signed certificate is sufficient for the system to verify that this version and the next version of your product are from the same source, namely, you. Once I have run this script to code sign my software, I move the signed application package and the signed framework bundle from the Code Signing Clean Room folder into my Payload folder, from which I will drag them into the appropriate receptacles in the PackageMaker window in order to record their paths.

I*I. Build the Installer Package*

A. Begin

1. Launch PackageMaker 3. Its main window opens showing an Untitled package near the upper-left corner and the package's Configuration pane on the right.

An **Install Properties sheet** is automatically presented. Enter the identifier for your organization in the form "com.<company name>". For my Event Taps Testbench installer, it is "com.pfiddlesoft". Set the Minimum Target to the minimum version of Mac OS X that your product requires. For my Event Taps Testbench installer, it is Mac OS X v10.5 Leopard, but you can choose versions as far back as Mac OS X v10.3 Panther. (As of PackageMaker 3.0.4, you still cannot choose Mac OS X v10.6 Snow Leopard as the minimum target.)

If you cancel the sheet now, the project window will automatically close, so you must enter something and click OK before you continue. You can edit these selections later by choosing Project > Install Properties from the menu bar, but this may require you to go back and change other settings, so it is better to complete this step now.

B. Edit the Interface

2. Click the **Edit Interface button** at the right end of the toolbar (alternatively, click the Edit Interface button at the bottom of the package's Configuration pane or choose Project > Edit Interface in the menu bar). I recommend editing the interface at the beginning, to get the easy part out of the way. The Interface Editor window opens to an "en" (English) localization. This window is a mockup of the window that the Installer application will present when your users install your product, with textual instructions for editing it which disappear as soon as you make changes. The Interface Editor window opens with the first, or Background, page selected (indicated by a subtle blue color for the bullet). The window's drawer is always open; if you drag it closed, it pops open again.

3. Drag your **background image file** and drop it onto the Interface Editor window (alternatively, choose the File radio button in the drawer and choose the Choose menu item in the action menu to

navigate to the image file, or type its full path in the text field). I use a 512x512 png version of the application icon. The image appears in the window behind the text and user controls; the selected radio button in the drawer changes from Default to File; and the path to the image file appears in the text field. (The Absolute and Relative to Project menu items in the action menu are always dimmed and the Absolute item is always checked.)

4. Use the Alignment and Scaling pop-up menus in the drawer to move and scale the image to taste. The Alignment pop-up menu items align the indicated edge of the image with the indicated edge of the window, running off the opposite edge in the case of very large images.

5. Click Continue (alternatively, click the Introduction step on the left). The Introduction step becomes selected on the left; the instruction text changes; and the title and user controls in the drawer change.

6. You can drag and drop a **Welcome text file**, just as you did with the image file in step 3, but you have the option to accept the familiar Default ("You will be guided through the steps necessary to install this software.") or to type custom text directly into the window. A Welcome file is a good place to describe your product briefly and to describe "What's New" in the latest release. If you prefer just to give the user a simple notice that the installer installs a shared framework in /Library/Frameworks, click the Embedded radio button. The instruction text is replaced by the phrase "Edit text here." Select that phrase and type to replace it with your preferred text.

7. Click Continue (alternatively, click the Read Me step on the left). The Read Me step becomes selected on the left; the instruction text changes; and the title and user controls in the drawer change.

8. You have the option to accept the default None radio button in the drawer, in which case the Read Me step will not appear at all in the finished installer. You also have the options to select Embedded and to type directly into the window as you did with the Welcome text in step 6. I want to give the user a reasonably detailed introduction to the product and the framework, so I drag my **Read Me text file** and drop it into the window. The text of the Read Me file (with its rtf formatting, if any, intact) appears in the window; the selected radio button in the drawer changes from None to File; and the path to the Read Me file appears in the text field.

9. Click Continue (alternatively, click the License step on the left). The License step becomes selected on the left; the instruction text changes; and the title and user controls in the drawer change.

10. The same considerations apply to this step as step 6 and 8. My application is free of charge but nevertheless comes with a license. The framework is also free for use with Event Taps Testbench, and it comes with its own license, separate from the application license. I could display the framework license separately from the application license by putting one of them in the Welcome, Read Me or Finish Up steps, but I prefer to put the full text of both licenses in the License step, in a single file. Drag the **Licenses text file** and drop it into the window.

11. Click Continue (alternatively, click the Finish Up step on the left). The Finish Up step becomes selected on the left; the instruction text changes; and the title and user controls in the drawer change.

12. The same considerations apply to this step as steps 6, 8, and 10. If you prefer not to make the installation process longer than necessary, leave the default None selected, and the Finish Up step will not appear at all in the finished installer. I prefer to let the user know that installation was successful, so I drop a **Conclusion text file** into the window. It warns the user not to uninstall the shared framework if other applications are using it.

13. Close the Interface Editor window. Now would be a good time to choose File > Save; I won't remind you again. The Save menu item is disabled while the Interface Editior window is open, so close it first. I save it in my PackageMaker ETT Installer Stuff folder under the name Event Taps Testbench Installer Project.

C. Configure the Installer Package

14. With the **Untitled Distribution package** selected in the left pane, just above the Contents heading, choose the **Configuration tab** in the main PackageMaker window if it is not already selected. The package is still labeled "Untitled (Distribution)" at this point because, although you've saved the project, you have not yet given the installer itself a title.

15. **Enter the installer's name** in the Title text field. This should be the name of the product without any other words because it is used by PackageMaker in various ways. I enter Event Taps Testbench as the title. As you type, the title appears in the left pane and at the top of the right pane.

16. In the **User Sees pop-up menu**, choose whether to show both Easy and Custom Install options or either option alone. In the Installer application, if you choose both, these will be echoed in buttons titled "Standard Install" and "Customize" in the lower-left corner of the installer window during the Installation Type step. I want to give the user the option to install the shared framework without installing the application, because the framework can be used by other applications. I therefore choose Easy and Custom Install.

17. Select one or more of the **Install Destination checkboxes**. These control several features when the installer is run, such as whether the familiar Change Install Location button appears near the lower-right corner of the installer window during the Installation Type step to allow the user to choose a different volume. Applications can be installed on any attached volume, so I select this checkbox. (I will still be able to restrict installation of the framework to a specific location later; see below.) I leave the other two checkboxes unselected because I do not want to restrict installation to the system volume or the user's home directory. (You will not be warned if you uncheck all of the Install Destination checkboxes, but unchecking all of them will generate an error when you build the install package.)

18. I understand that Apple recommends signing the installation package. However, the self-signed certificate available from an earlier step appears to be inappropriate for this purpose. I have found that the installer fails with an "untrusted" error message, even if I have used Keychain Access to designate the certificate as trusted. If you have a commercial signing certificate, click the **Certificate button**. A sheet is presented asking you to choose a certificate from Keychain Access. Select it, click the Show Certificate button to verify that it's the right one, and click Choose.

19. Type some text into the **Description text view**. I do not know whether or where this description is used, but it seems advisable to provide a very brief description of your product. My Description is "Event Taps Testbench is a free developer utility to test Apple's Quartz event taps API." Save the project.

D. Set Package Requirements

20. With the Event Taps Testbench Distribution package still selected at the top of the left pane, choose the **Requirements tab** in the main PackageMaker window.

21. Click the **"+" button** to add a requirement. A sheet is presented where you can design requirements and provide the message that will appear if the requirement is not satisfied.

22. Use the **If pop-up menu** to choose a requirement type. There are many kinds of available requirements. My product runs only on Mac OS X (Leopard) or newer, so I choose the Target OS Version requirement.

23. Use the **Is pop-up menu** to choose a comparison. I choose >=. The other available requirements offer different user controls and tests.

24. Fill in the requirements for the test in the **If text field**. For the version requirement, I type "10.5.0," relying on the formatting hint in the If menu item. The other available requirements offer different user controls.

25. Type a failure message in the **Message text view**. I type "Event Taps Testbench requires Mac OS X v10.5 Leopard or newer." This will appear in the bottom text box when you click on a volume in the installer window where you choose an install location, if the volume fails to meet the requirement. The title text view cannot be selected or edited because the error message is used in an untitled text view.

26. Click OK, then choose Required or Optional and true or false in the **Required and "Pass if" columns** of the Requirements table, as appropriate.

27. Repeat steps 21-26 to add more requirements. The Megabytes Available on Target requirement is a good candidate for use in every installer. I set it to >= 10, with a failure message reading "Event Taps Testbench requires at least 10 megabytes free on the installation volume." Save the project.

E. Set Preinstall and Postinstall actions

28. With the Event Taps Testbench Distribution package still selected at the top of the left pane, choose the **Actions tab** in the main PackageMaker window.

29. I will eventually use both preinstall and postinstall actions, but I cannot set up the particular postinstall action I want until I have added a certain component to the project, so I will do that later. I can set up the preinstall action I want now, namely, an action that will quit any previous version of the application being installed that is currently running on the computer. Click the **Edit button below the Preinstall Actions table**. A sheet is presented containing several available actions in the Actions pane on the left. Preinstall and Postinstall actions are set up identically using an Automator-like interface and workflow.

30. Drag the **Get Application action** to the right pane. An Automator-like configuration tile appears.

31. Type either the identifier or the name of the application being installed. I would normally type the current Event Taps Testbench identifier, "com.pfiddlesoft.EventTapsTestbench." However, previous versions of the application had different identifiers, "com.prefab.EventTapsTestbench" and "com.prefabsoftware.testbench," due to name and ownership changes. I therefore use the official name of the application, instead, which has always been "Event Taps Testbench.app", to catch any version that might be running.

32. Drag the **Quit Application action** to the right pane. Another Automator-like configuration tile appears after the Get Application tile and connected to it to illustrate the progression of the workflow.

33. Click Save. The sheet closes and the two steps of the preinstall action appear in the Preinstall Actions table. If the installer is run while Event Taps Testbench is running, the installer should cause Event Taps Testbench to quit when it reaches the point of actually performing the installation. Save the project.

F. Add Component Choices and Components
F.1. Add the Application Component

34. Drag the **code-signed release version of the application bundle** from the Payload folder in the Installer Stuff folder and drop it onto the Contents pane of the main PackageMaker window, below the Contents heading where it says "Drop contents here" (alternatively, choose Project > Add Contents from the menu bar or from the action menu in the lower-left corner of the window, navigate to the application bundle, and choose it). An outline heading with a blue dot, known as a "component choice," appears in the Contents pane, with the name of the application. The component choice is expanded, showing the application component itself below it by icon and name, together with the path to its target installation folder beneath (/Applications by default, for application components). The actual component is selected, and the Configuration pane to the right is already fully configured.

35. Select the **Event Taps Testbench component** in the Contents section of the left pane, if it is not already selected, and choose the **Configuration tab** in the main PackageMaker window. Examine the Configuration pane to consider whether anything should be changed. Most of the items are already set to default values or values determined when you added the component, and they probably do not need to be changed. The **Install and Destination text fields** show the source application package's path and the default target installation folder path, respectively. You probably do not need to change them, but you can choose a different application package to install or a different destination if you wish, either by typing new paths, dragging and dropping a different item, or using the action menus to the right of the text fields. The **Package Identifier and Package Version text fields** are used internally. You should increment the Package Version each time you release a new version of your installer package. The Version Number bears no relation to the product version numbers. I increment it to 2 because this is the second release of Event Taps Testbench to use an installer package. The **Package Location setting** at the bottom of the window defaults to Self-Contained, which is probably what you want; use the action menu to the right to change it if, for example, this component is to be downloaded from the Internet. The **Patch button** is for use with upgrade installations; I haven't used it. The **Restart Action pop-up menu** defaults to none, which is appropriate for this installer.

36. The **"Allow custom location" checkbox** is selected by default. This allows the user to install this component in another location than the default Destination. You can deselect it, but applications can typically be installed anywhere so you might want to leave it selected.

37. The **"Require admin authentication" checkbox** is selected by default. This allows the user to install in privileged locations like the /Applications folder by authenticating as an administrative user. You can deselect it if, for example, you specify a non-privileged location such as the user's home folder as the Destination. Save the project.

38. Choose the **Contents tab** in the main PackageMaker window.

39. Select the **Event Taps Testbench.app** line in the table.

I do not know what the **Filters button** does, and it is not documented. If you click it, a sheet appears with several regular expressions. You can apparently add more and save them, but I have not tried that because I don't know what they do. Although DS_Store files are included among the filters that are included by default, DS_Store files nevertheless appear in my finished installer package (and I leave them there). The checkboxes in the Valid column cannot be deselected, at least for the default filters, but default filters can be deleted.

In most cases, you should NOT click the **Apply Recommendations button**. If you do,

PackageMaker will automatically set the owner, group and privileges of the component and all its contained items to "recommended" values. How PackageMaker determines what settings are recommended is not documented, but I understand that it uses values that you may have used in a previous installation, which may not be appropriate.

Instead, check the **Read, Write and Execute checkboxes for Owner, Group and Others**, and choose a value from the **Owner and Group pop-up menus**.

Most, but not all, of Apple's applications have system as the owner (choose root in the Owner pop-up menu if you want to set this value) and wheel as the group, with all of the checkboxes set except the Write checkbox for group and others. However, if you choose these settings for your own application, your application probably won't run on Snow Leopard because setguid access is not allowed, at least if you don't have a commercial certificate.

A sizable number of Apple's applications have system as the owner and admin as the group, with Write access allowed for both owner and group. This is what I use, mostly because several of the sources cited at the end of this article recommend it, including Apple's PackageMaker User Guide. It works for me. (Note that it may be necessary to be running as an administrative user to set these values; I always run as an administrative user, so I don't know whether this will work if you don't have administrative privileges.)

Third-party applications are all over the lot, but many of them set Owner to the current user ("cheeseb" on my computers) and Group to admin, giving Write access only to Owner. I've tried that, too, and it works just fine.

To set the Event Taps Testbench.app privileges, owner and group, you must expand the component by clicking its disclosure triangle, and manually expand every single subocmponent until all are visible, then hit Command-A to select all of them. Then set the checkboxes and choose values from the pop-up menus. This is tedious and error-prone; it would be nice if there were a Select All setting, but there isn't. In some cases, it may be appropriate to set individual subcomponents to different values, but you'll have to know more than I do about permissions to figure that out. The checkboxes in the left column of the table default to selected, but I have no idea what they do, if anything, and it is not documented.

I have verified that setting the owner, group, and privileges at this stage will not cause the code signing mechanism to reject the files as having been modified. I don't care in this case because I'm going to set the PFEntrustETT component application to root privileges in a postinstall script.

40. Leave the **"Include root in package" checkbox** selected. Otherwise, the installer will install the component's subcomponents without their enclosing container, which is normally not what you want when you're installing an application package or framework bundle. It might be useful to deselect this checkbox if, say, you dragged and dropped a folder full of scripts for convenience but you want the scripts to be installed without their containing folder. Save the project.

41. Choose the **Components tab** in the main PackageMaker window.

42. The **Allow Relocation checkbox** is selected by default because applications can typically be run from any location. This checkbox is often said, erroneously, to control whether the user is allowed to move the component after it is installed. In fact, it controls whether the installer, if run again, will search for the component in all possible locations and replace it somewhere other than in the designated Destination if it is found somewhere else.

This may seem like a desirable setting, because it honors the user's choice about where the application belongs: The user will find the new version right where the old version was located. A

user who has an extra copy on the compouter may be surprised to discover that only one of them was updated, but that is a relatively rare circumstance and the user will presumably realize pretty quickly what the problem is.

Developers must use this option with caution, however. Allowing relocation can cause you to become confused during testing, if you test your installer on the same computer where you developed it, because the installer will find the application in your Builds folder or in the Payload folder of your Installer Stuff folder and therefore fail to install it in the specified Destination. The moral is to do your testing on a "virgin" computer, or delete or zip all existing versions of the application.

43. Click to select the component and enable the Scripts and Relocation buttons, then click the **Relocation button**. A sheet is presented where you can set many parameters to specify how the installer will search for this component in case the user relocated it after it was first installed. You will probably be able to leave the settings at their default values, a Combination Search. For upgrade installations later on, you may want to set the "Min version" and "Max version" text fields. As the legend near the bottom of the sheet explains, a combination search succeeds if the application is found at the specified path (i.e., by name), or by LaunchServices, or by doing an exhaustive search. I leave the identifier set to com.pfiddlesoft.EventTapsTestbench even though earlier releases had different identifiers, because here we get a name search, too.

44. The **"Allow Downgrade" checkbox** is deselected by default. I leave it that way because I do not want my users installing old versions of my software after a new version is released and installed.

45. You can add a preinstall or postinstall script for this specific component. Here is an example that doesn't actually do anything. **Start by composing a shell script** in any text editor (TextEdit will do). You can use Perl if you prefer. Be sure to format it as a plain text file, not an RTF file. Save it with a file name that will be clear to you next year when you come back to revise your installer for an update to your product. I name my script file "Postinstall-EventTapsTestbench-ComponentScript.txt." Here is the text of my sample script:

```
#!/usr/bin/env bash

# The $2 parameter is the full path to the application package Event Taps Testbench.app. The installer package runs this script after installation.

touch -c "$2"

echo "Ran Event Taps Testbench.app component postinstall script to touch $2"

exit 0
```

The $2 variable expands a shell parameter containing the text of the full path to the application component after it is installed, even if the user told the installer to install it somewhere other than the default destination. For the default destination, the variable expands to "/Applications/Event Taps Testbench.app". Note that, for a component script, the variable includes the application package, so you don't have to add it yourself. This is because this script was installed using the Scripts button in the Components pane of the Event Taps Testbench component window; as you will see below, a script installed using the Scripts pane of the component window expands the $2 variable to the folder in which the component is installed, in this case /Applications. For a list of other available parameters and environment variables that might be useful, consult Apple's Software Delivery Guide. This script is, at least theoretically, unnecessary because the Installer runs the touch command automatically.

46. Back in the PackageMaker Components pane, click the **Scripts button at the bottom**. A sheet is presented containing Preinstall and Postinstall text panes (if you were building for Mac OS X 10.4 (Tiger), you would see text panes for additional script phases that are not needed in Mac OS X 10.5 (Leopard)).

47. Drag the saved shell script's Finder icon and drop it on the Postinstall text field (alternatively, choose Choose in the action menu to the right, navigate to it, and choose it, or type its full path into the text field). The full path appears in the text field. Click Save. When you build the installer package, PackageMaker will automatically configure it as an executable, rename it "postinstall", and place it properly in the finished installer package. (You may have to select the Postinstall text field by clicking it before it will accept the drop. Also, the path in this text field has a habit of disappearing randomly when I make changes to the installer package and open and close it a lot, so be sure to double-check this field every time you make a change to the installer package.) Save the project.

48. Choose the **Scripts tab** in the main PackageMaker window. The Scripts pane contains a Scripts directory text field where you can designate a folder where you keep source scripts, but this requires you to watch out that stray or invisible files are not included. For a simple installer like this, I prefer to designate the scripts specifically. As it happens, my application contains an embedded helper application package and its executable must run setuid, as root, when the user runs my application and chooses a particular option. To arrange for this to work after my application is installed, I must compose another script that the installer will run after installation is complete to change the permissions and owner of the embedded helper application's executable.

49. **Compose another shell script** and save it with a memorable file name. I name mine "Postinstall-EventTapsTestbench-Script.txt." Here is the text of my script:

```
#!/usr/bin/env bash

# Make the PFEntrustETT helper application in the Event Taps Testbench application package a setuid application, because the AXMakeProcessTrusted function must run as root.
# The $2 parameter is the full path to the folder in which the Event Taps Testbench package was installed. The installer package runs this script after installation.

installedHelperPath="$2/Event Taps Testbench.app"/Contents/PFEntrustETT.app/Contents/MacOS/PFEntrustETT

chmod 6555 "$installedHelperPath"
chown root:admin "$installedHelperPath"

echo "Ran Event Taps Testbench.app postinstall script to setuid $installedHelperPath"

exit 0
```

50. **Drag the saved shell script to the Postinstall text field** (alternatively, choose Choose in the action menu to the right, navigate to it, and choose it, or type the full path into the text field). The full path appears. (The text fields in this pane accept drops even if they are not selected.) Save the project.

F.2. Add the Framework Component

51. Repeat steps 34-50 as appropriate to add the framework "component choice" and component. Here are some different settings that might be appropriate for a framework as compared to an application: In the framework component's Configuration pane, the "Allow custom location"

checkbox is unchecked by default because frameworks should always be installed in /Library/Frameworks. Leave the "Require admin authentication" checkbox selected because /Library/Frameworks is a privileged location. In the Components pane, the "Allow Relocation" checkbox is deselected by default for the same reason. For my framework, I click the Scripts button to add another postinstall script, similar to the script in step 45, above, to make sure the newly-installed framework's information appears immediately in the Finder. For my framework, I do not need a script in the Scripts pane.

F.3. Configure the Application Component Choice

52. Select the Event Taps Testbench **"component choice" in the Contents pane**. A component choice governs the checkbox displayed in the Installer application when the user chooses to customize an installation that allows customization.

53. Choose the **Configuration tab** in the PackageMaker main window if it is not already selected.

54. To make the choice clearer to the user, edit the **Choice Name text field** to "Event Taps Testbench application." The name of the component choice in the Contents pane to the left changes to match.

55. You can edit the **Identifier text field** to any name that you like, as long as it is unique within the installer, such as "Event_Taps_Testbench_app_choice." But this is used internally, and I will leave it as is ("choice0").

56. I want this component choice to be initially selected and enabled, so I leave the **Initial State checkboxes** set that way, as they are by default.

57. The documentation says that the component choice **Destination text field** can be used to override the Destination as set in all of the components in this component choice. I do not want to override that Destination, but the documentation also says that this field must always be filled in. So I type "/Applications" (alternatively, use the action menu's Choose menu item).

58. Applications can be installed on any volume available to the computer, so I select the **"Allow alternate volume" checkbox**.

59. Type short help tag text into the **Tooltip text field**. This will appear when the user hovers the mouse cursor over this choice in the customization window. My tooltip says "The Event Taps Testbench application requires the PFEventTaps framework" to alert users that they had better not deselect the framework choice (in fact, I will disable the framework choice later so the user cannot deselect it.)

60. Type a short description of this component choice in the **Description text field**. This will appear in a text box when the user clicks this component choice in the customization window. My description says "Install the Event Taps Testbench application. It requires the PFEventTaps framework." Save the project.

61. Choose the **Requirements pane** in the PackageMaker main window. It works just like the installer package Requirements pane. Since I already required Mac OS X 10.5 (Leopard) for my entire product, I choose not to repeat the requirement here.

F.4. Configure the Framework Component Choice

62. Repeat steps 52-61 as appropriate to set up the framework "component choice." Here are some different settings that might be appropriate for a framework as compared to an application: Change

the Choice name from "PFEventTaps" to "PFEventTaps framework" for clarity. I deselect the Enabled checkbox, because I do not want to permit the user to install the application without the framework, or to install nothing at all. I set the Destination to /Library/Frameworks. I do not select the "Allow alternate volume" checkbox because frameworks must be on the startup volume. For a tooltip, I type "Deselect Event Taps Testbench to install the PFEventTaps framework by itself for use by other applications." For a Description, I type "Install the PFEventTaps framework. It may be used by other applications as well as Event Taps Testbench."

63. Choose the Requirements tab in the PackageMaker main window. I could enable and disable the framework component choice depending on the selected or deselected state of the application component choice by adding a Choices requirement to this component choice. However, I already set the framework choice's initial state to disabled, and I want to leave it that way under all circumstances. Save the project.

G. Set Preinstall and Postinstall actions

64. Choose the **Event Taps Testbench distribution installer** again at the top of the left pane of the PackageMaker main window, then choose the **Actions tab**. I can now add the postinstall action I promised earlier.

65. Click the **Edit button** below the Postinstall Actions table. A sheet is presented containing several available actions in the Actions pane on the left.

66. Click the **Show File in Finder action** and drag it to the right pane. An Automator-like configuration tile appears.

67. Choose Component in the **Type pop-up menu**, if it is not already chosen.

68. Click the **arrow button** to the right. The sheet flips over, and the back side now shows both components that I have added to the installer project.

69. Select the **Event Taps Testbench component**, then click the **Event Taps Testbench application** when it appears in the first column of the pane to the right, then click the **Choose button**. The sheet flips back to its front side.

70. Click Save. The sheet closes and the single step of the postinstall action appears in the Postinstall Actions table.

H. Build the Installer

71. Click the **Build or Build and Run button** at the left end of the toolbar in the PackageMaker main window. A Save sheet is presented. Give the finished installer package a name (the name of the product) and choose Where to save it, then click Save. I save it in the Installer Stuff folder.

72. If you have previously saved the installer package using the same name, another sheet will be presented asking you whether to replace it. You will normally do so, unless you are creating an upgraded installer and want to preserve the old version. If you chose earlier to codesign the installer, you may be asked for permission to use your signing certificate in an alert; click Always Allow.

73. Any build errors and warnings will appear in the PackageMaker main window, where you can examine them. If you want to make changes, click the "Return to Editing" button in the main window's toolbar or the Return button near the bottom of the window. I find that I can ignore the occasional warning about non-matching permissions on .DS_Store files. In fact, I ignore all

warnings resulting from my having set permissions differently than what PackageMaker seems to have wanted me to set.

74. If you clicked the Build button, the Install Package did not run. You can run it by clicking the "Open in installer" button in the PackageMaker main window, or by clicking the "View in Finder" button and then double clicking it in the Finder.

I. Test the Installer Package

75. It is vital that you test your new installer in every situation you think it may encounter after it is released into the wild. If yours is a Universal product, test it on PPC and Intel computers. Test it on a "virgin" computer that has never seen it before, and on computers where it is already installed. Test it on computers where it is already installed in odd locations, and where older (and perhaps fake newer) versions are already installed. Install it on computers that have a different user. Be sure to take full advantage of the installer Log feature of the Installer application to see what is happening; open the log by choosing Window > Installer Log while Installer is running. To examine the contents of the installer package, you can use Pacifist, an excellent shareware application by Charles Srstka from CharlesSoft <http://www.charlessoft.com/>.

Miscellaneous Issues

If running the installer package replaces an older version of an application or a framework, Installer gives the application file and the framework folder the current date by running the touch command. Unfortunately, if running under Leopard, this does not update the Version column in a Finder window, so a Leopard user who watches the Version column may be misled into believing that the application was not updated. The Finder's Get Info command also shows the old version number. However, QuickLook gets it right, and running the application and opening its About dialog shows that the new version was in fact installed, too. Running an AppleScript 'update' command on the application or framework does not fix this problem. Logging out and back in does, and eventually the passage of time will fix the problem, as well. This does not appear to be a problem under Snow Leopard.

It appears that, under circumstances that are not clear, some versions of PackageMaker may build the installer using previous settings. To avoid this problem, always save before building. It would be prudent also to quit PackageMaker and relaunch it before building the final installer package, just to be sure.

You can work on multiple installer projects at once by opening multiple PackageMaker main windows. However, it is not very convenient because of these issues affecting some versions of PackageMaker:

• if a document-modal sheet is open on any one of the main windows, you will be blocked from editing any of the other main windows.

• Depending on what you do while you are working in one PackageMaker main window, other open PackageMaker main windows may go blank. Drag any other window over them to force them to redraw.

• If you close one PackageMaker main window, the other main windows close immediately and the application quits.

• Under circumstances that are not clear, it can become impossible to close the last PackageMaker main window or to quit the application. In that case your only resort is to Force Quit it and then reset anything that had not been saved before the crash.

Other Sources
Note: I do not vouch for the accuracy or completeness of any of these, Including Apple's.

For Code Signing:

Apple's Code Signing Guide

Apple's Code Signing Release Notes

Apple's Technical Note TN2206: Mac OS X Code Signing In Depth

Apple's Software Delivery Guide

For PackageMaker 3:

Apple's PackageMaker User Guide

Apple's Distribution Definition Reference

Apple's Software Delivery Guide (not yet updated for PackageMaker 3, but contains essential information)

The Manual Page for the packagemaker command line tool

José R.C. Cruz, "The Flat Package: Examining a newer package format," MacTech Magazine (vol. 26, no. 2, Feb 2010), page 52

José R.C. Cruz, "Scripting PackageMaker: Checking Requirements: Learn how to validate a target for installation," MacTech Magazine (Aug 2009), page 42

José R.C. Cruz, "Packagemaker: Delivering Applications: Learn how to build an application installer for OS X 10.5," MacTech Magazine (vol. 25, no. 5, May 2009), page 26

Greg Neagle, "MacEnterprise: Packaging for System Administrators: Building Installer packages for software distribution," MacTech Magazine (vol. 25, no. 3) <http://www.mactech.com/articles/mactech/Vol.25/25.03/2503MacEnterprise-PackagingforSystemAdministrators/index.html>

José R.C. Cruz, "Packaging Special Payloads with PackageMaker: Bringing drivers, libraries, and plug-ins to the OS X target," MacTech Magazine (vol. 24, no. 10, 2008) <http://www.mactech.com/articles/mactech/Vol.24/24.10/PackagingSpecialPayloadswithPackageMaker/index.html>

José R.C. Cruz, "Packaging for Leopard: Introducing the new PackageMaker 3.0," MacTech Magazine (vol. 24, no. 6, Jun 2008) <http://www.mactech.com/articles/mactech/Vol.24/24.06/IntroducingthenewPackageMaker3.0/index.html>

Simone Manganelli, "PackageMaker and Installer" <http://homepage.mac.com/simx/technonova/tips/packagemaker_and_installer.html>

For PackageMaker 2 and older:

Edward Marczak, "Mac in the Shell: Packaging and Installing: Simplify and scale your install methodology," MacTech Magazine (vo. 24, no. 6, 2008) <http://www.mactech.com/articles/mactech/Vol.24/24.06/2406MacintheShell/index.html>

Steven Osborn, "XCode PackageMaker Tutorial" (May 9, 2007) <http://steven.bitsetters.com/articles/2007/05/09/xcode-packagemaker-tutorial/>

José R.C. Cruz, "Distributing with PackageMaker: Building a distribution installer package," MacTech Magazine (Vol. 22, no. 12, 2006) <http://www.mactech.com/articles/mactech/Vol.22/22.12/2212PackageMaker/index.html>

Stéphane Sudre, "PackageMaker How-to" <http://s.sudre.free.fr/Stuff/PackageMaker_Howto.html>

Andrew Anderson, "PackageMaker Pro Tips," O'Reilly Mac DevCenter (Sep 16, 2003) <http://www.macdevcenter.com/pub/a/mac/2003/09/16/packagemaker.html>

Conclusion

I hope you find this helpful. Please bring any errors or omissions to my attention at bill@cheeseman.name.