

Creating A Mac OS X Package for Omnis Applications

Revised February 18, 2007

Mark Phillips, Mophilly & Associates

Focus on Omnis Studio v4.2 and Mac OS X 10.4

Introduction

This document is the result of the efforts of several people. The primary contributors, in order of appearance, are Larry Barcot, Kelly Burgess, and Michael Monschau. Without Larry's "simple" question at AmerOmnis 2003, none of us would have investigated this technique. I would also like to thank Christopher Cozad and Bob Whiting for their commentary and observations, and Walter Venable for updating this document to cover Omnis Studio v4.2 and for constructing the accompanying sample application "Active Widget."

This document presents the steps necessary to build a Mac OS X application package with Omnis Studio v4.2 or higher. It is assumed that you are proficient with Omnis Studio and application development in general, and that you have an Omnis Studio based application ready to be integrated into a Mac OS X app package.

The goal is to have your entire application "under" a single icon on the Mac OS X desktop. In addition, the Mac OS X application menu will display the name of your application.

For a thoroughly professional appearance, you will want to add code to your library to allow the "Preferences" command on the Mac OS X application menu to open your preferences window, and the About command to display the correct name and open the correct window class. In addition, you will want to create file icons and a creator code. These tasks are outside the scope of this document.

However, sample solutions to many of the extra issues involved in making your application appear like a "real" application instead of a Studio library are demonstrated in the associated sample library "Active Widget". The following explanation refers to this example, and the names and values used in various steps are those of the Active Widget sample application.

You need to be familiar with the following Apple Computer utilities, which are part of the Apple Developer Tools.

Property List Editor (or a commercial version such as PlistEdit Pro)

Icon Composer

For more in-depth information read the developer documentation provided by Apple Computer:

<http://developer.apple.com/documentation>

<http://developer.apple.com/documentation/MacOSX/Conceptual/BPRuntimeConfig/Concepts/ConfigFiles.html>

The Process

This process takes about 30 minutes to one hour to complete. Allow about two hours for your first time through. Make backup copies of all your sensitive files before you begin.

Step 1 - Install Omnis Runtime

Install an Omnis runtime and make a copy of the runtime icon. In Omnis v4.2 (and higher), all of Omnis' support files are actually inside the Runtime icon, which is really what is called a "package." More on this later. Rename the copy to whatever makes sense for your application as this will become the application icon for the end user. In our example, we renamed the runtime to "Active Widget". This is referred to as the "Application Package".

Step 2 - Reveal Application Package Contents

Open the Application Package. You do this by right-clicking on the Application Package icon (Active Widget) and selecting the "Open Package Contents" command from the context menu. The package contents will open in a separate Finder window. Switch the window to "column view" and you will see a directory tree populated with the various files and directories that comprise the Runtime and, soon, your application as well.

Step 3 - Place Your Studio Application in the Package

If your "Contents" Finder window is displaying in column view, you can click on the "Contents" folder and see that it contains several items, including a folder named "MacOS". Click on this folder and it will display all the files and folders that make up the main part of the Omnis Runtime. In this directory is a folder named "startup".

Place your application libraries and special files within this folder, taking care to arrange the files exactly as you would have them in the end user installation. Placing your main library in the "startup" folder will cause it to be loaded whenever the application is launched. Do not place the main data file which will contain user data in this folder, but you may choose to include secondary data files. Examples of this might be a simple data file to store user preferences, or data files that contain relatively constant data like lookup lists of states or ZIP codes. In our example, we place the two files "Active Widget.lbs" and "AppPreferences.df1" in the "startup" folder. For the Active Widget app, "AppPreferences.df1" is used to hold user preferences.

Test this setup rigorously so that any problems are resolved before you attempt to build the package. Do this by closing the "Contents" window and double-clicking on your application icon (Active Widget). Your application should start up and run normally, with the application menu (next to the Apple menu) still saying "Omnis". If your application has trouble accessing any secondary data files you placed in the "startup" folder, refer to the code in "Active Widgets.lbs", method "Startup_Task.prefsinitialize" for an example of how "Active Widgets" locates and opens its preferences datafile "AppPreferences.df1".

Step 4 - Modify the info.plist File

At the top level inside the "Contents" folder is a file called "info.plist". Open the info.plist file with the Property List Editor. There are a number of entries to consider, only some of which we will need to change to customize the Studio Runtime.

These are described in greater detail in the Apple document "Examining and Modifying Property Lists". It can be found at this rather lengthy URL:

http://developer.apple.com/documentation/DeveloperTools/Conceptual/SoftwareDistribution/Concepts/sd_pkg_flags.html#//apple_ref/doc/uid/20001940/TPXREF14

In the Property List Editor, expand the root node and update the following properties in info.plist. Save the changes and close the file when you are done.

Property Name	Type	Value
<u>CFBundleExecutable</u>	String	Omnis The name of the Unix executable file in Contents:MacOS. It is best just to leave this parameter with the value "Omnis" and to leave the Unix executable file in Contents:MacOS named Omnis. This name is only used internally, and will not be visible to the user when they are running your application.
<u>CFBundleGetInfoString</u>	String	Active Widget 1.0 © 2007 Walter M. Venable Distributed by Mophilly and Associates This is the version info that appears in the "Get Info" window in the Finder. In most cases, you will only use a single line of version information, for example just "Active Widget 1.0 © 2007 Walter M. Venable" in our example. If you wish to have a secondary line of information below the first like we do in Active Widget, you must compose the two lines of text in a word processor or TextEdit and put a carriage-return between them. Then select the two lines and paste them into the "Value" box in the Property List Editor. There is no other way we know of to enter a carriage-return into the "Value" box. After you have finished editing you may have to restart your computer to see this value show up in the Finder's Get Info window.
<u>CFBundleIconFile</u>	String	omnis.icns This is the name of the icon file in the Contents:Resources folder. We recommend leaving it set to "omnis.icns" and just modify that file. If you wish, you can rename the file to something appropriate to your application and then modify the renamed version. If you do this, this value must be changed to match the renamed icon file.
<u>CFBundleIdentifier</u>	String	com.ActiveWidget.MophillyAssociates This does not seem to show up anywhere that is visible to the user, but we change it anyway.
<u>CFBundleName</u>	String	Active Widget This is the name that will appear as the application's name in the menu bar (next to the Apple menu). When you click on that name to open the application's menu, this name will also appear on the menu items "Hide <applicationname>" and "Quit <applicationname>". It seems to be the copy of this parameter in InfoPlist.strings (see Step 5 below) that actually controls what shows up as the application name in the menu bar, probably since this is the copy that will be localized if you customize your app for different languages. The value of the copy in Info.plist does not seem to matter, but should be set to the English version of the name anyway.
<u>CFBundleShortVersionString</u>	String	1.0, © 2007 Walter M. Venable This is the version info that appears in the right hand panel (preview panel) when you click on your application in the Finder's "column" view. As with the CFBundleGetInfoString (see above), after you have finished editing you may have to restart your computer to see this value show up in the Finder's "column" view.

CFBundleVersion

String 1.0

This is the version number of your "bundle." As with the CFBundleIdentifier (above) this does not seem to show up anywhere that is visible to the user, but we change it anyway.

Step 5 - Modify the Language Resources File(s)

In the package contents folder is a folder named "Resources". Open it and you will see a folder "English.lproj". This folder contains a file named InfoPlist.strings. (Depending on your runtime and the country you are delivering in, there may be other ".lproj" folders. Each will contain an InfoPlist.strings file, and each must be modified in a similar manner.) This file can be opened with the Property List Editor, although that is not the default application for these files.

You may notice that the content of this file is very similar the info.plist file. It is not precisely the same, so don't attempt to merely copy the info.plist file and rename it. Open the InfoPlist.strings file(s) with the Property List Editor and copy the values for the appropriate properties from info.plist to the InfoPlist.string file(s). (If you are modifying an InfoPlist.strings file for another language, you will probably need to adjust the values to make sense for that language, although we have not tried this). Close and save the file(s).

Step 6 - Add Your Icons

You have a choice of providing a new icons file or updating the "omnis.icns" file with your icons. These files are found in Contents:Resources. Since we used the the string "omnis.icns" for the value of CFBundleIconFile in the info.plist file, we will modify the existing "omnis.icns" file and put our own icons in it

If you choose to rename the file "omnis.icns" or create a new one with a different name, then your file name has to match property CFBundleIconFile in info.plist and InfoPlist.strings (see Step 4).

In either case you will use the utility program "Icon Composer" to put your icons in the file. Start off with a 128x128 bit "Thumbnail" icon picture. This is what will appear in the right hand panel (preview panel) when you click on your application in the Finder's "column" view. Copy your picture to the clipboard, click on the Thumbnail icon box and paste in your picture.

For the other size icons you can choose to hand draw each size, but it is very easy to generate the rest of the sizes by dragging. First click on the "Thumbnail" picture and drag it into the "Huge (48x48)" box. Then click on the Huge icon and drag it into the "Large (32x32)" box. Finally click on the Large icon and drag it into the "Small (16x16)" box. This will create a set of different icon sizes with minimal effort.

Step 7 - Add the Finder Creator and Type codes (optional)

If you have registered your application creator type with Apple you will want to modify the file named "PkgInfo". If you have not registered a creator type, you can skip this step.

BEdit will open this file and the creator and type codes are entered on one line. E.g. the Omnis creator code is OO\$\$, so the string in PkgInfo is "APPLOO\$\$. This tells the Finder that the package is an application (APPL) with a signature of "OO\$\$. If you use a custom creator code, there is a related attribute "CFBundleSignature" in the info.plist and InfoPlist.strings files that must contain your creator code.

Step 8 - Wrapping up

Close the Finder window for the package Contents. On my machine the Finder icon did not update automatically when I closed the Contents window. To resolve this select the package icon, which still displays the Omnis icon, and open the Get Info dialog. Click on the Omnis icon in the upper left corner and press the delete key. The Finder will read the package contents and update the icon. If you then restart your computer, the Finder's version information for your application should update as well.

You are now ready to ship the Mac OS X version of your Omnis application.