

How to Create A Mac OS X Package for Omnis Applications

Revised October 29, 2004

Edited by Mark Phillips

Contact via email: mark@mophilly.com

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.

This document presents the steps necessary to build a Mac OS X application package with Omnis Studio v4.0 or higher (1). 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.

It is possible to use this technique with Omnis Studio 3.3.3, but version 3.3.3 is different from Omnis Studio 4. Please review the foot notes before attempting this with Omnis Studio v3.3.3.

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.

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

Property List Editor

Icon Composer

It is recommended that you read the developer documentation provided by Apple Computer:

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

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

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

Step 1 - Install Omnis Runtime

Install an Omnis runtime and make a copy of the entire directory tree. Rename the copy to whatever makes sense for your application as this will become the application folder for the end user. This folder is referred to as the "application folder".

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. Your main library can be placed in the startup folder so that it is loaded whenever the application is launched.

Test this setup rigorously so that any problems are resolved before you attempt to build the package.

Step 2 - Reveal Runtime Package Contents

Open the Omnis package inside the Omnis folder. You do this by selecting the "Open Package Contents" command from the context menu for the Omnis application. The package contents is a directory tree populated with various files and directories that comprise the application.

Step 3 - Name the Bundle and Executable

Decide on a name for your bundle and Omnis executable inside Contents:MacOS. They do not have to be identical. Short names are best for the Bundle Executable. In addition, leave out spaces and punctuation.

For this example, let's say we are a company named Widgets Wannanet Company and the product is Active Widget. Therefore, we have this information:

Company: Widgets Wannanet Company

Product Name: Active Widget

Version: 1.0

So we choose these names:

Bundle: Active Widget

Omnis Executable: Active Widget

Now, open the folder named "MacOS" and rename the following files:

Original name	New name
Omnis	ACTIVEWIDGET
OmnisPDE.bundle	ACTIVEWIDGETPDE.bundle

Close the MacOS folder.

Step 4 - Modify the info.plist file

Open the info.plist file with the Property List Editor. There are a number of entries to consider. These are described in 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/

Expand the root node and update the following properties in info.plist (2). Save the changes and close the file when you are done. In my tests, I encountered a crash when using a lower case name for CFBundleExecutable. This is merely anecdotal, but the package stopped crashing when I ran through these steps again using an executable name in all capitals.

CFBundleExecutable	String	ACTIVEWIDGET
CFBundleGetInfoString	String	1.0 © 2004 Widgets Wannanet Company
CFBundleIconFile	String	activewidget.icns
CFBundleIdentifier	String	com.widgetswannanetcompany.activewidget
CFBundleInfoDictionaryVersion	String	6.0
CFBundleName	String	Active Widget
CFBundlePackageType	String	APPL
CFBundleShortVersionString	String	1.0

Step 5 - Modify the language resources files

In the package contents folder is a folder named "Resources". Open it and you will see two folders, "English.lproj" and "German.lproj". These folders each contain one file named InfoPlist.strings (3). These files 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 these are very similar to the info.plist file. They are not precisely the same so don't attempt to merely copy the files and rename them. Open each with the Property List Editor and copy the values from info.plist to the each of the InfoPlist.strings files. Close and save the files.

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 string "activewidgets.icns" for the value of CFBundleIconFile in the info.plist file, we created a new icns file named "activewidgets.icns" and placed it in the same directory as the file "omnis.icns".

If you choose to modify "omnis.icns" then rename the file "omnis.icns" to "activewidget.icns". Again, this name has to match property CFBundleIconFile in info.plist and its kin.

Step 7 - Add the Finder Creator and Type codes

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.

BBEdit will open this file and the creator and type codes are entered on one line. E.g. the Omnis creator code is OOS\$, so the string in PkgInfo is "APPLOO\$". This tells the Finder that the package is an application (APPL) with a signature of "OOS\$". There is a related attribute in the info.plist and InfoPlist.strings files that must agree with the contents of PkgInfo.

Step 8 - Add your application files

Move all files and folders that you need to ship with your application from the Omnis root folder into the MacOS folder inside the bundle. In most cases this will be all the files and folders installed by the Omnis Runtime installer, excluding the runtime itself, plus your application library, related externals and files.

Place your main library in the "startup" folder so that it will load automatically. Name the file so it loads last, to preserve the order of things.

Step 9 - Wrapping up

Close the Finder window for the package Contents. On my machine the Finder icon did not update automatically when I close 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.

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

Footnotes

1. This technique was pioneered using Omnis Studio 3.3.3 and it does not work with any version prior to Omnis Studio 3.3.3. This is due to changes in the Omnis core.
2. For Omnis Studio 3.3.3, you will have to add the property "CFBundleExecutable" as it does not exist in the info.plist.
3. Step 5 "Modify the language resource files" applies only to Omnis Studio v4.0 and, presumably, higher.