

Transitioning Mac CFM plugins to Mach-O

This document contains instructions for converting VectorWorks plug-ins to work with VectorWorks 12.5.0. Previously, the VectorWorks application used the PEF file format to run on versions of the Mac OS previous to Mac OS X as well as PowerPC Macintosh OS X. With Apple's transition to the Intel CPU architecture, VectorWorks has been changed to use the Mach-O file format so that our favorite CAD application can continue to operate on both the PowerPC and Intel Macintosh computers.

This means that all VectorWorks SDK plug-ins must also be in the Mach-O file format. Fortunately this does not require a lot of changes for simple plug-ins.

At this point, it is assumed that the reader is familiar with Xcode. If not, please refer to these resources:

1. User's Guide: Launch Xcode and select "Xcode Help" from the "Help" menu. A window will appear that displays the User Guide Introduction to Xcode 2.2.
2. GCC Porting Guide: In a similar manner to the above point, select "Documentation" from Xcode's "Help" menu. In the search field of the documentation window, type "GCC Porting Guide" read this.
3. "Porting CodeWarrior Projects to Xcode" from Xcode's Help if you have PowerPlant projects to convert..
4. Free training videos at [Apple Developer Transition Resource Center](#). Scroll down to the Tools section and click the QuickTime bullet links to watch the various training videos. You'll need an ADC Membership, which is free.
5. And for more complex plug-ins, refer to [Scoping Your Transition Project](#).

Contents

1. Become familiar with the location of our sample project files.
2. Convert a sample PEF plug-in to a Mach-O Universal Binary Plug-In.

1. Become familiar with the location of our sample project files.

You'll find sample Xcode projects of plug-ins in the "/Source/Sample" directory. These include:

- a. SampleObject
- b. SampleObjectTool
- c. SamplePluginLibrary
- d. SamplePowerPlantDialog
- e. SampleTool
- f. Ellipsoid
- g. SampleLayout

2. Convert a sample PEF plug-in to a Mach-O Universal Binary Plug-In.

As mentioned above, there are several types of plug-ins. For our demonstration, we will show how to convert a VectorWorks Tool PowerPC PEF Plug-In to a Mach-O Universal Binary Plug-In. This example will convert an example plug-in titled "MyTool". The "MyTool" directory would be in the "/Source/Shipping" directory.

Create the project.

1. Create a directory called “MyTool” in the “/Source/Shipping” directory.
2. Navigate to the “/Source/Sample/SampleTool” directory.
3. Copy the “SampleTool.xcodeproj” file into the “MyTool” directory.
4. Navigate to the “MyTool” directory, select the “SampleTool.xcodeproj” project file and rename it to “MyTool.xcodeproj”. Xcode project files are actually bundles (special directories), and as such, the real information is contained in files inside that special folder. If the SampleTool project was locked, you will have to unlock your copy of it:
 - a. Right-click the “MyTool.xcodeproj” file and select “Show Package Contents”.
 - b. A window will appear with the file “project.pbxproj”. Right-Click that file and select “Get Info”. Make sure that the “Ownership & Permissions” is set to “Read and Write”.
 - c. Close the Window.
5. The “MyTool” directory will have its own source and resource files: “MyTool.cpp” and “MyTool.rsrc”. This represents your own proprietary code and resource files that you will have with your own existing PEF plug-ins. For this example, copy a .cpp file and .rsrc file from one of your existing plug-ins to the MyTool directory, and rename those files to “MyTool.cpp” and “MyTool.rsrc”, respectively.

Modify the source of your project to match your new source.

6. Double-click the “MyTool.xcodeproj” file. This will launch Xcode and open up the project file.
7. In the “Groups & Files” pane, the “SampleTool.cpp” and “SampleTool.rsrc” files should be displayed in red text. This means that the project does not know where they are. Select these two files and delete them.
8. Right-Click the “MyTool” project icon and select “Add->Existing Files...”. In the file dialog, navigate to the “MyTool” directory and select the “MyTool.cpp” and “MyTool.rsrc” files. These files will now be part of your project.

Update the name of the Plug-In to match the desired name.

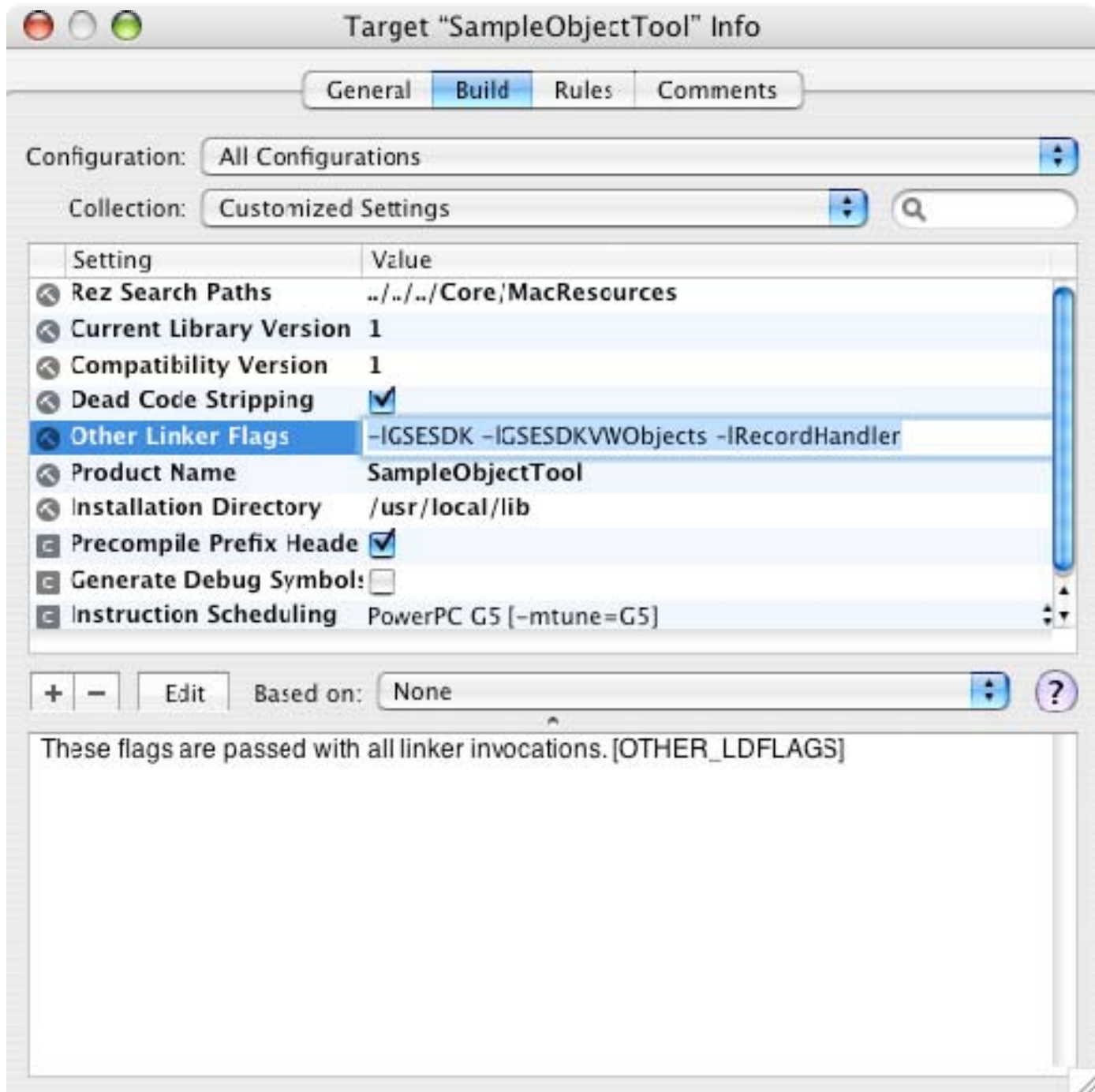
9. Open the “Targets” group in the “Groups & Files” pane. Select the “SampleTool” target. Type “Command-I” or click the blue info button.
10. Click the “General” tab and rename the target to the new target name, “MyTool”. This is not the name of the Plug-In that will be built, just a human-readable way of identifying that this is the target that builds the plug-in. You can have multiple targets per project (see Xcode Help for more information).
11. Click the “Build” tab. In the search field, type “Product Name”. One setting should appear with the value “SampleTool”. Double-Click that value and type “MyTool”. This renames that actual built plug-in. Close this window.

Build your Plug-In.

12. Choose to build either the “Dev” configuration, or the “Release” configuration. The Dev configuration builds only for the native architecture (either PowerPC or Intel), while the Release configuration builds a Universal Binary that will run natively on both the Intel and PowerPC Macintosh computers.
13. Click “Build” and you should have a successfully built plug-in.

Note: If you wish to debug your plug-in, select “New Custom Executable...” from the “Project” menu. In the resulting Assistant dialog, give the custom executable some name, provide the path to the host application (VectorWorks) and add it to your project. You will be able to set break points inside your plug-in and you can click “Build and Debug”.

Note: If your project requires other libraries, you should specify them in the "Other Linker Flags" for the Target. This is done by adding "-l<LibraryName>" (-l == lower case L) to the "Other Linker Flags" setting. Normally, libraries on the disk are named lib<LibraryName>.a. For example the RecordHandler library is named "libRecordHandler.a". See below:



Note: Make sure that your SDK plug-in name has the correct extension:

1. ".vwlibrary" for VectorWorks Libraries
2. ".vwobject" for VectorWorks object tools
3. ".vwplugin" for regular VectorWorks plug-ins

These are also listed in the NNA_BuildPlugin.xcconfig files in /SDKLib/Include/OnlyMac/.