Developing with Copland

This note details some of the steps you should take and issues you will discover as you start developing with this release of Copland. This is an early release of the system and is intended to let tools developers begin adapting compilers, linkers, integrated development environments, debuggers, frameworks and other tools to Copland's new foundations.

Before you can make any progress developing software to run on Copland, it is very important that you follow the detailed installation and configuration instructions described in "Installing and Working w/ Copland". It is essential that a debug host Macintosh running the Power Mac Debugger be connected via serial cable to the Power Macintosh running Copland. Many components in Copland include benign asserts and DebugStr messages. Please note such asserts if they in fact lead to subsequent problems.

Once you are able to successfully boot Copland, you are ready to start making progress building software!

This release of Copland is intended to primarily host PowerPC native software development and it is essential that you start with a native, CFM code base before proceeding. Subsequent developer releases of Copland will feature a complete 68K emulation environment.

There are several tips in using the Power Mac Debugger that will improve your ability to build code and debug with this release of Copland:

1.Turn **off** the **Stop for DebugStrs** menu item option in the **Extras** menu if you are experiencing a large number of asserts or other passive DebugStr messages.

2.In the **General Preferences…** dialog found in the **Edit** menu, turn **off** the "Stop on code loads for existing processes" check box found in the "Process Control" area. This will remove the debugger breaks that are encountered each time the system loads new code.

3.During development, use embedded symbols in your code so when viewing stack crawls with the debugger, you will have more meaningful pointers back to your source. The use embedded symbols option in the debugger is located in the same **General Preferences…** described above.

4.At times, it is necessary to manually map .SYM files when they appear non-targeted. To do so, select **Open** from the **File** Menu and then select Map **Symbolics to Code** from the **File** menu. Your fragment should be displayed selected in a list of fragments in the context. Click O.K., and the .SYM file window should say Targeted at the bottom.

5.Read the document "Interfaces and Libraries ReadMe" included on this CD carefully and make sure you properly specify the appropriate compiler flags so that the declarations you expect are available.

6. There is some sample code provided on the CD. It is a good place to start.