



Developing Palm OS Conduits for PowerPC and Intel Macs Using Xcode

Technical Whitepaper
July 29, 2005

Table of Contents

Background.....	3
Intel-related Concerns.....	3
Conduit Compatibility	4
Introducing the Bridge Library.....	4
Developing a Conduit from Scratch	4
Converting an Existing Conduit to Xcode.....	5
Debugging Conduits	5
For More Information.....	6

Background

Palm OS conduit development for the Macintosh has typically been accomplished using Metrowerks CodeWarrior, an aging development environment that is no longer being updated or, at the time of this writing, even available for sale. This situation is negatively impacting Mac/Palm conduit development, as even the handful of enterprising developers who have managed to write their core code using Apple's modern Xcode development environment are still tied to CodeWarrior when it comes to writing necessary shim code. As such, the majority of developers are stuck in the veritable dark ages using CodeWarrior.

With Apple's Intel announcement at their 2005 Worldwide Developers Conference, and their recommendation that all developers switch to Xcode as soon as possible for portability to the new Macs, the time is right for conduit development to be done entirely in Xcode.

Mark/Space plans to develop a universal binary of The Missing Sync™ for Palm OS—the company's HotSync Manager replacement—that will run on both Intel and PowerPC processors without relying on Apple's compatibility layer, Rosetta. This move gives developers the advance notice necessary to plan for the arrival of Intel-based Macs and to begin reaping the benefits of using modern tools to for conduit development.

Intel-related Concerns

Mark/Space plans on supporting Intel processors by delivering universal binaries of all components contained in The Missing Sync for Palm OS. However, in order for The Missing Sync to load conduits and run natively, all conduits must also be compiled as universal binaries.

According to Apple, only Xcode produces Mach-O universal binaries. While The Missing Sync already supports conduits compiled using Xcode, Palm's HotSync Manager software does not, and conduit developers are keen to target the largest possible market, including users of both The Missing Sync and HotSync Manager.

If even one of the conduits on a user's system is not compiled as a universal binary, then The Missing Sync will be forced to run in the Rosetta compatibility layer, which will most likely be slower than running natively, though just how much slower is not yet apparent.

So how does all of this affect the end user?

Initially, conduits developed in Xcode or CodeWarrior will act the same when loaded by either The Missing Sync or HotSync Manager. (There are some advantages today to writing conduits that run exclusively under The Missing Sync, including a progress bar for use by the conduit, planned ability to launch a helper application from within the user interface, as well as other upcoming features that have yet to be announced.) However, when Apple begins shipping Intel-based Macs, conduits developed in CodeWarrior will require The Missing Sync to run in a compatibility mode, potentially degrading performance. HotSync Manager, as it currently stands, will always run in a compatibility mode; no plans have been announced to modernize HotSync Manager using Xcode.

Conduit Compatibility

Conduit developers have stuck to creating CFM based conduits in order to address the largest possible market with their products. Additionally, many conduits are developed using PowerPlant, which is not easily usable with Xcode.

Mark/Space engineers have managed to overcome both of these issues by developing a bridge library and documentation outlining the changes necessary for PowerPlant to compile under Xcode 2.1 or later.

Introducing the Bridge Library

Mark/Space's bridge library is available free of charge to all developers. This library is a universal binary that will operate natively on both Intel- and PowerPC-based Macs. Built as a Mach-O static library, it will communicate with HotSync Manager's libraries in addition to The Missing Sync's framework. HotSync Manager is a CFM application, so all calls must be made through CFM. The Missing Sync handles CFM and Mach-O calling through the use of shims, so this bridge library is geared more towards compatibility with HotSync Manager, such that calls from a Mach-O conduit can access HotSync Manager's CFM shared library.

Simply put, linking with and using Mark/Space's universal binary bridge library will allow developers to create conduits in Xcode and have them work with both The Missing Sync for Palm OS and HotSync Manager, whether on Intel or PowerPC processors.

Mark/Space encourages conduit developers to plan for the future by using this new bridge library and updating their conduits today. Because the resulting conduits will remain compatible with both The Missing Sync and HotSync Manager, there are no downsides to pursuing this course of action. The advantage is continued operation on Mac OS X in such a way that The Missing Sync will be able to operate and launch conduits natively on Intel-based Macs.

Developing a Conduit from Scratch

If the Palm OS CDK is not already installed, please see the instructions contained in the sample code archive available from the developers section of the Mark/Space website, <http://www.markspace.com/developers.html>.

To create a conduit that works with both The Missing Sync for Palm OS and HotSync Manager, start with the provided SampleConduit Carbon source code. This code is an Xcode project that links in the Mark/Space bridge library and demonstrates how to make various calls into the Sync Manager, User Manager, etc., and shows various entry points that are used when a conduit is run. (Developers wishing to use Cocoa for their conduits need either to use the Cocoa conduit sample code at the same location, which is only supported in The Missing Sync, or adopt this sample code for use with Cocoa. At this time, Mark/Space is not aware of an easy way to do this such that a Cocoa conduit would operate under both Missing Sync for Palm OS and HotSync Manager.)

Conduits developed using Xcode can either use old style resources (.r) files or newer Carbon *nib* files. For ease of developing user interfaces, Mark/Space recommends that developers use Carbon nib files. This is demonstrated in the sample code.

Converting an Existing Conduit to Xcode

Many recently developed or updated conduits written in CodeWarrior are bundled conduits and not just a single executable. These conduits may be slightly easier to move to Xcode than conduits that are not bundled. The time consuming aspects of moving conduits from CodeWarrior to Xcode involve getting the PowerPlant framework to compile and handling differences in the gcc compiler vs. the CodeWarrior compiler. This basically means compile, fix errors/warnings, compile and repeat until there are no more errors/warnings.

In the sample code archive, Mark/Space has provided a shell for converting a PowerPlant-based project to Xcode. Additional PowerPlant files may have to be added to the project depending on the specific needs of the conduit. Compiling the conduit as a universal binary may produce a large number of warnings. Developers may choose to either fix these warnings or to ignore them. The access paths for the PowerPlant and/or CDK may need to be changed, depending upon the locations of these base files. Simply click on the group in the Xcode project, Get Info on it, and then change the path for the group. All of the CDK and PowerPlant files are referenced relative to these groups, so no additional changes will be necessary. Prior to compiling the shell, apply the PowerPlant patch file Mark/Space has provided in the archive. The patch file is for Metrowerks CodeWarrior 9.0 (PowerPlant version 2.2.2).

In addition to making these changes in PowerPlant, all resources in the conduits that are resource fork-based, i.e. *.rsrc* files, must DeRez into *.r* files and be included in the Xcode project. If PowerPlant *ppobs* are used, they will need to be converted into Carbon *nibs* or standard *.r* files and then manipulated according to standard resource conventions for the chosen file type.

As shown in the PowerPlant shell, all referenced PowerPlant *.cp* and *.h* files must be added to the Xcode project in order for it to compile. Pre-compiled libraries (as provided in the Palm OS CDK) will not work in Xcode, since they were compiled using CodeWarrior.

In order to speed up future compiles, a developer may elect to create a pre-compiled header file that includes all the PowerPlant headers. Also, one could add all the PowerPlant *.cp* files in a new project and create a statically linked library so that PowerPlant files do not have to be recompiled. When the statically linked library is compiled, it should be compiled as a universal binary.

As an example, the entire conversion of the TimeCopy conduit from CodeWarrior to Xcode, including figuring out the PowerPlant changes, took approximately 3 hours. Granted, this conduit is relatively simple, but when one factors out the time needed to figure out the PowerPlant changes, the time to do the conversion is under one hour.

Debugging Conduits

Once a conduit is successfully compiling in Xcode, debugging the conduit for either the conduit settings or the conduit operation itself is quite easy. Mark/Space found during development that creating a symbolic link to the conduit in

/Library/Application Support/Palm HotSync/Conduits is necessary for debugging and also speeds development.

To get started, choose New Custom Executable from Xcode's Project menu, navigate to the location of Missing Sync for Palm OS.app and name the executable User Interface. Copy the path that is presented, as you will need it in a coming step. One more time, choose New Custom Executable and name it Conduit Manager. Click the Choose button and then press Command-Shift-g. Paste in the path you just copied and append Contents to it. Then navigate to SharedSupport and find Conduit Manager.app.

When you're ready to debug your user interface, select Set Active Executable from the Project menu and choose User Interface. Now you can debug with breakpoints and all the magic of gdb.

If you want to debug the conduit itself, you first must tell the Conduit Manager that you want to debug conduits. From the command line, type `defaults write com.markspace.missingsync.palms debugConduits -bool YES`. (You need only do this step once and it won't interfere with normal operation of The Missing Sync.) Select Set Active Executable and choose Conduit Manager. Begin debugging, and when you're ready to go, press the HotSync button on your handheld.

For More Information

- Mark/Space has also published a whitepaper on its Intel strategy to be used as a companion to this document.
- Sample code can be found at: <http://www.markspace.com/developers.html>
- Mark/Space hosts a conduit developers mailing list. Please visit: <http://lists.markspace.com> for information on signing up. Mark/Space engineers regularly read and respond on this list and are more than willing to help developers convert their conduits. Post to this list for assistance.

© 2005 Mark/Space, Inc. All rights reserved. Mark/Space and The Missing Sync are trademarks of Mark/Space, Inc. The names of actual companies and products mentioned herein may be trademarks of their respective owners. Product specifications are subject to change without notice. This material is provided for information purposes only; Mark/Space assumes no liability related to its use.