

Developing Conduits for The Missing Sync for Palm OS

© 2004 Mark/Space Inc. All Rights Reserved.
July 29, 2004 (rev 3)

This document applies to The Missing Sync for Palm OS version 4.0

For the latest developer information, see:
<http://www.markspace.com/developers.html>

Introduction

Palm conduits are dynamic libraries that are loaded by The Missing Sync during the synchronization process. Each conduit typically synchronizes one type of information between the desktop machine and the Palm OS device. The Missing Sync supports existing Carbon/CFM conduits as well as Carbon/Mach-O and Cocoa/Mach-O ones. (If you don't know what this means, see the glossary below)

For Carbon conduits, nearly all of the documentation at PalmSource's web site still applies. See <http://www.palmsource.com/developers/>. Even if you are writing Cocoa conduits, you'll want to familiarize yourself with that documentation.

The Missing Sync for Palm OS 4.0 supports the same calls as PalmSource's Conduit Developers Kit version 4.03 plus a subset of newer "Dm" database calls for the Tapwave Zodiac. Let us know which new calls you need so we can prioritize their implementation.

Steps to Building a Conduit

Step 1. Choose a development environment, object format, and API set to use

The chart below summarizes the common choices that conduit developers make. If you want compatibility with Palm's HotSync Manager, you'll have to use CodeWarrior to create CFM conduits. If you only need to support The Missing Sync, you can use any combination of CodeWarrior, Xcode, CFM, Mach-O, Carbon, and Cocoa.

Requirement	IDE	Object File Format	Carbon or Cocoa	Link against Palm's HotSync Libraries or The Missing Sync framework?
Support both MacOS 9 and MacOS X	CodeWarrior	CFM	Carbon	Palm's
Support both Palm's HotSync Manager and The Missing Sync	CodeWarrior	CFM	Carbon	Palm's
Share syncing code with a Windows conduit	CodeWarrior or Xcode	CFM or Mach-O	Carbon	Either
Support databases with >64K records	CodeWarrior or Xcode	CFM or Mach-O	Either	The Missing Sync
MacOS X and The Missing Sync only	CodeWarrior or Xcode	CFM or Mach-O	Either	The Missing Sync

Step 2. Find a sample conduit and duplicate the directory tree

The Missing Sync ships with sample Carbon and Cocoa conduits for Xcode. Palm's Conduit Developers Kit ships with sample CodeWarrior conduits and templates for new conduits. Choose whichever you prefer.

Step 3. Rename the files and update the plist and resources

Change the executable name, plist signature, resource strings, etc, that refer to the original sample conduit.

Step 4: Update the include paths and libraries if needed

If you want to link against the Missing Sync framework from a CodeWarrior conduit, you'll need to change the library and header file paths to point to The Missing Sync framework and away from PalmSource's CDK.

Step 5: Add support for the calls OpenConduit and ConfigureConduit

ConfigureConduit puts up the conduit settings dialog. OpenConduit is called when it is time to sync.

Step 6: Test, rinse, repeat

Debugging Your Conduit

Currently, it's not possible to debug your conduit using Xcode or CodeWarrior's debugger. We typically use the HotSync logging calls or printf to a file. We're looking into ways of fixing this.

Installers

Some older installers checked specifically for Palm's HotSync Manager application before proceeding with the installation. Although we've tried to emulate that file within the Missing Sync bundle, it may not work with all installers. Instead, your best bet is weak link some installer code against "HotSync Libraries" and test to see if any symbol is available. For example:

```
#include <CodeFragments.h>
#include "SyncMgr.h"

if (SyncAddLogEntry != kUnresolvedCFragSymbolAddress) {
    // HotSync or The Missing Sync is installed
}

// test to see if newer "Dm" calls are available

#include <CodeFragments.h>
#include "SyncDm.h"

if (SyncDmWriteRecord != kUnresolvedCFragSymbolAddress) {
    // The Missing Sync has implemented that call
}
```

If you don't wish to write custom code, you can look for "HotSync Libraries" and "HotSync Libraries.cfm" in /Library/CFM Support. If either exists, proceed with the installation.

You should use the CDK "User Manager" calls to determine where to install files relative to the current Palm OS user. These calls are documented by PalmSource and available in UserMgr.h.

Additional Functionality

Progress Bar

If you want to support the Missing Sync's progress bar in your conduit, you can pass specially formatted strings to the conduit progress function. To set the progress bar to 10% (1/10th) and the text to "Doing Something", you would use the following string:

```
%%1%%10%%Doing Something
```

The string is of the format %%numerator%%demoninator%%Status string

Tapwave Zodiac

The Missing Sync for Palm OS implements the following additional calls for the Tapwave Zodiac:

```
// from SyncDm.h
//
// these calls support database records greater than 64K in size
//
// the official documentation for these is in version 6 of the Windows CDK

typedef unsigned char    HSByte;
typedef HSByte          *HSBytePtr;
typedef long             HSError;

HSError SyncDmWriteRecord(
    HSByte handle, UInt32 *pRecordID, UInt16 categoryIndex,
    HSByte attributes, UInt32 dataOffset, UInt32 dataSize,
    HSBytePtr pRecordData);

HSError SyncDmWriteResourceRecord(
    HSByte handle, UInt32 resourceType, UInt16 resourceID,
    UInt32 dataOffset, UInt32 dataSize, HSBytePtr pResourceData);

HSError SyncDmReadRecordByID(
    HSByte handle, UInt16 *pRecordIndex, UInt32 recordID,
    UInt16 *pCategoryIndex, HSByte *pAttributes, UInt32 dataOffset,
    UInt32 *pDataSize, HSBytePtr pRecordData, UInt32 *pDataRemaining);

HSError SyncDmReadRecordByIndex(
    HSByte handle, UInt16 recordIndex, UInt32 *pRecordID,
    UInt16 *pCategoryIndex, HSByte *pAttributes, UInt32 dataOffset,
    UInt32 *pDataSize, HSBytePtr pRecordData, UInt32 *pDataRemaining);

HSError SyncDmReadResourceRecordByIndex(
    HSByte handle, UInt16 resourceIndex, UInt32 *pResourceType,
    UInt16 *pResourceID, UInt32 dataOffset, UInt32 *pDataSize,
    HSBytePtr pResourceData, UInt32 *pDataRemaining);
```

Note: *These calls are supported only for the Tapwave Zodiac right now and you have to link against Mark/Space's framework rather than Palm's HotSync Libraries.*

Contacts

Developer web page: <http://www.markspace.com/developers.html>
Developer mailing list: see the above web page for instructions
PalmSource CDK and documentation: <http://www.palmsource.com/developers>

Frequently Asked Questions

Where do I go for more information?

<http://www.markspace.com/developers.html>

<http://www.palmsource.com/developers>

How do I debug my conduit?

At the moment (July 27, 2004) it's difficult to debug your conduit using Xcode or CodeWarrior. We're working on solutions to this. Internally we generally use the HotSync logging facility to print messages to the log or printf to a file.

Which is easier to develop conduits with, Xcode or CodeWarrior?

Not much of a difference.

What version of MacOS X does The Missing Sync for Palm OS require?

10.2.8 or later

Any performance advantages with Mach-O over CFM?

None that we're aware of.

Can I write a conduit in Java? RealBasic?

We haven't tried either. Our header files are in C or Objective-C, so you'd have to translate those. And you'd have to create a shared library file and export the proper routines. Probably more trouble than it is worth.

Glossary

The Code Fragment Manager (CFM) is an object file format originally developed by Apple for MacOS 7.1.2 and is still supported in MacOS X. Palm conduits were traditionally CFM shared library files.

Mach-O is an object file format used exclusively on MacOS X (and NeXTStep). The Missing Sync supports both CFM and Mach-O conduits.

CodeWarrior is a development environment from Metrowerks <<http://www.metrowerks.com>>. CodeWarrior must be used to develop conduits if you require compatibility with either MacOS 9 or Palm's HotSync Manager.

Xcode is a development environment from Apple <<http://www.apple.com/developer>>. Xcode is free and uses the open source gcc compiler. Xcode can be used to create either Carbon or Cocoa conduits in the Mach-O object format. Xcode cannot be used to create conduits compatible with Palm's HotSync Manager, but can be used to create conduits compatible with The Missing Sync.

Carbon is a set of application programmer interfaces (APIs) that maintain compatibility with older (pre MacOS X) Macintosh code. Carbon programs are typically written in C/C++ but can be written in other languages as well.

Cocoa is an object oriented API using the Objective-C language and is unique to MacOS X.

PalmSource's Conduit Developers Kit (CDK) is a set of header files, libraries, and examples for writing conduits.

A Bundle is a directory tree containing MacOS executable, informational, and resource file(s) in specific internal locations. Bundles are presented to the user as a single file in the Finder and make it easier to localize your conduit or application.