



Introduction to Palm OS[®] Developer Suite

Palm OS[®] Developer Suite

Written by Brian Maas and Mark Dugger

Technical assistance from JB Parrett, Brad Jarvinen, Phil Shoemaker, Kenneth Albanowski, Yann Cheri, Greg Clayton, Kevin MacDonell, Patrick Porlan, Keith Rollin, Michael Dilloughery, Rob Stevenson, Puneet Mishra, and Keithen Hayenga

Copyright © 2003-2005, PalmSource, Inc. and its affiliates. All rights reserved. This technical documentation contains confidential and proprietary information of PalmSource, Inc. ("PalmSource"), and is provided to the licensee ("you") under the terms of a Nondisclosure Agreement, Product Development Kit license, Software Development Kit license or similar agreement between you and PalmSource. You must use commercially reasonable efforts to maintain the confidentiality of this technical documentation. You may print and copy this technical documentation solely for the permitted uses specified in your agreement with PalmSource. In addition, you may make up to two (2) copies of this technical documentation for archival and backup purposes. All copies of this technical documentation remain the property of PalmSource, and you agree to return or destroy them at PalmSource's written request. Except for the foregoing or as authorized in your agreement with PalmSource, you may not copy or distribute any part of this technical documentation in any form or by any means without express written consent from PalmSource, Inc., and you may not modify this technical documentation or make any derivative work of it (such as a translation, localization, transformation or adaptation) without express written consent from PalmSource.

PalmSource, Inc. reserves the right to revise this technical documentation from time to time, and is not obligated to notify you of any revisions.

THIS TECHNICAL DOCUMENTATION IS PROVIDED ON AN "AS IS" BASIS. NEITHER PALMSOURCE NOR ITS SUPPLIERS MAKES, AND EACH OF THEM EXPRESSLY EXCLUDES AND DISCLAIMS TO THE FULL EXTENT ALLOWED BY APPLICABLE LAW, ANY REPRESENTATIONS OR WARRANTIES REGARDING THIS TECHNICAL DOCUMENTATION, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING WITHOUT LIMITATION ANY WARRANTIES IMPLIED BY ANY COURSE OF DEALING OR COURSE OF PERFORMANCE AND ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, ACCURACY, AND SATISFACTORY QUALITY. PALMSOURCE AND ITS SUPPLIERS MAKE NO REPRESENTATIONS OR WARRANTIES THAT THIS TECHNICAL DOCUMENTATION IS FREE OF ERRORS OR IS SUITABLE FOR YOUR USE. TO THE FULL EXTENT ALLOWED BY APPLICABLE LAW, PALMSOURCE, INC. ALSO EXCLUDES FOR ITSELF AND ITS SUPPLIERS ANY LIABILITY, WHETHER BASED IN CONTRACT OR TORT (INCLUDING NEGLIGENCE), FOR DIRECT, INCIDENTAL, CONSEQUENTIAL, INDIRECT, SPECIAL, EXEMPLARY OR PUNITIVE DAMAGES OF ANY KIND ARISING OUT OF OR IN ANY WAY RELATED TO THIS TECHNICAL DOCUMENTATION, INCLUDING WITHOUT LIMITATION DAMAGES FOR LOST REVENUE OR PROFITS, LOST BUSINESS, LOST GOODWILL, LOST INFORMATION OR DATA, BUSINESS INTERRUPTION, SERVICES STOPPAGE, IMPAIRMENT OF OTHER GOODS, COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR OTHER FINANCIAL LOSS, EVEN IF PALMSOURCE, INC. OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR IF SUCH DAMAGES COULD HAVE BEEN REASONABLY FORESEEN.

PalmSource, Palm OS, Palm Powered, and certain other trademarks and logos are trademarks or registered trademarks of PalmSource, Inc. or its affiliates in the United States, France, Germany, Japan, the United Kingdom, and other countries. These marks may not be used in connection with any product or service that does not belong to PalmSource, Inc. (except as expressly permitted by a license with PalmSource, Inc.), in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits PalmSource, Inc., its licensor, its subsidiaries, or affiliates. All other product and brand names may be trademarks or registered trademarks of their respective owners.

IF THIS TECHNICAL DOCUMENTATION IS PROVIDED ON A COMPACT DISC, THE SOFTWARE AND OTHER DOCUMENTATION ON THE COMPACT DISC ARE SUBJECT TO THE LICENSE AGREEMENTS ACCOMPANYING THE SOFTWARE AND OTHER DOCUMENTATION.

Introduction to Palm OS Developer Suite

Document Number 3122-003

June 3, 2005

For the latest version of this document, visit

<http://www.palmos.com/dev/support/docs/>.

PalmSource, Inc.

1188 East Arques Avenue

Sunnyvale, CA 94085

USA

www.palmsource.com

Table of Contents

About This Book	vii
What This Book Contains	viii
Additional Resources	viii
Eclipse Workbench Documentation	ix
C/C++ Development Toolkit (CDT) Documentation	x
PRC-Tools Documentation.	x
Cygwin Documentation.	x
Palm OS 68K API Documentation	xi
Palm OS Protein API Documentation	xi
Palm OS Developer Suite Documentation	xii
 1 Palm OS Developer Suite Overview	 1
Palm OS Application Development Tasks	2
Design Considerations	2
Process for Building Palm OS Applications.	3
Process for Building Palm OS 68K Applications	4
Process for Building PACE Native Objects	5
Process for Building Palm OS Protein Applications	7
Components of Palm OS Developer Suite	11
Workbench Integration	11
Palm OS Compiler Tools	11
PRC-Tools.	12
Cygwin Packages	12
Palm OS Resource Tools.	13
Testing and Debugging Tools	14
Palm OS Package Builder	15
 2 Eclipse Workbench Integration	 17
Using the Palm OS C/C++ Development Perspective	18
Setting the Workbench Perspective	18
Setting Workbench Preferences.	19
Using Projects to Manage Application Development	21
Using Workspaces	21
Using Makefiles	21

Creating a New Project	21
Select an Application Type.	22
Select Standard Make or Managed Make	23
Use the New Project Wizard to Create a Project.	25
PACE Native Object Projects	33
Changing Project Settings for PNO Resources	33
Shared Library Projects	35
Importing Files into a Project.	36
Importing 68K Source Files	36
Fixing Line Delimiters	38
Importing Palm OS Resource Files	39
Importing Palm OS Sample Applications	42
Editing Project Source Files	42
Building a Project.	43
Building a Standard Make Project	43
Building a Managed Make Project	44
Modifying Tool Properties	47
GNU Compiler Preferences	47
Palm OS Protein Compiler Preference.	48
Palm OS Resources Preferences	48
Target Environment Settings Preferences	48
Launching Other Palm OS Tools	50
Palm OS Garnet Simulator.	50
Palm OS Cobalt Simulator.	50
Palm OS Virtual Phone	51
Palm OS Debugger	51
Palm OS Emulator	51
Palm OS Reporter	52
Palm OS Resource Editor	52
Updating Palm OS Developer Suite	52
Features.	52
Prerequisites.	53
Verifying Features	53
Obtaining New Features	54
Obtaining Updates to Existing Features	54

Managing Features	55
Getting More Information	57
3 Palm OS Compiler Tools	59
Using Compiler Tools with the Workbench.	59
Setting Compiler Options	60
Running the Compiler Tools	61
Using the Palm OS Protein C/C++ Compiler Tools Independently	
61	
pacc, paasm, palink: Compiler Tools	62
palib: Librarian	63
elfdump: Diagnostic Tool	63
Getting More Information	63
4 Palm OS Resource Tools	65
Resource File Overview	66
Importing a Resource File	66
Creating an XRD File	66
Using the Resource Tools with the Workbench	67
Setting Resource Compiler (PalmRC) Options	67
Setting Resource Linker (PRCMerge) Options	69
Using the Resource Tools Independently.	72
GenerateXRD: Resource Migration Tool	74
Palm OS Resource Editor: Resource Editing Tool	74
PalmRC and PRCMerge: Resource Building Tools	74
PRCCompare: Resource Utility Tool	74
hOverlay: Application Localization Tool.	74
PRCSign and PRCCert: Application Security Tools	74
Getting More Information	75
5 Palm OS Testing Tools	77
Using Palm OS Simulator	78
Palm OS Garnet Simulator.	78
Palm OS Cobalt Simulator.	78
Creating a Palm OS Simulator Run Target	78
Using Your Palm OS Simulator Run Target	79

Palm OS Simulator User Interface	80
Using Palm OS Reporter	82
Using Reporter with 68K Applications	82
Using Reporter with Palm OS Protein Applications	83
Displaying Trace Information in Reporter	83
Using Palm OS Emulator	84
Creating a Palm OS Emulator Run Target	84
Using Your Palm OS Emulator Run Target	85
Using Virtual Phone	86
Launching Virtual Phone from the Workbench	87
Virtual Phone User Interface	87
Getting More Information	88
6 Palm OS Debugging Tools	89
Using Integrated Debugging	90
Integrated Debugging Overview	90
Setting a Breakpoint in an Application	91
Creating a Debug Configuration	91
Using Palm OS Debugger	93
Launching Palm OS Debugger from the Workbench	93
Palm OS Debugger User Interface	95
Getting More Information	97
7 Palm OS Package Builder	99
Palm OS Package Builder Overview	99
Using Palm OS Package Builder	100
A Hints and Tips	103
Build Issues	103
gcc Compiler Issues	105
gdb Debugger Issues	105
68K Applications	106
Index	107

About This Book

Introduction to Palm OS Developer Suite is a conceptual introduction to the Palm OS® application developer tools.

This book describes the general process for developing a Palm OS application, and provides an overview of the Palm OS developer tools.

This book covers all of the tools delivered with Palm OS Developer Suite. Palm OS Developer Suite is an integrated development environment that enables you to create the following types of Palm OS applications:

- 68K applications, for all Palm OS releases
- 68K applications with PACE Native Objects (PNO), for Palm OS Garnet and later releases
- Palm OS Protein applications, for Palm OS Cobalt and later releases

What This Book Contains

This book has the following organization:

- [Chapter 1, “Palm OS Developer Suite Overview,”](#) on page 1, describes the process you follow to create a Palm OS application, and explains how each of the tools are used in the process.
- [Chapter 2, “Eclipse Workbench Integration,”](#) on page 17, introduces the Eclipse Workbench integrated development environment. This chapter describes the developer tasks supported by the Eclipse Workbench.
- [Chapter 3, “Palm OS Compiler Tools,”](#) on page 59, introduces the Palm OS Protein C/C++ Compiler and associated tools: the compiler, the assembler, and the linker.
- [Chapter 4, “Palm OS Resource Tools,”](#) on page 65, introduces the Palm OS resource tools, which allow you to create user interface objects and other resources for your application.
- [Chapter 5, “Palm OS Testing Tools,”](#) on page 77, introduces the Palm OS testing tools Palm OS Emulator, Palm OS Garnet Simulator, Palm OS Cobalt Simulator, and Palm OS Virtual Phone.
- [Chapter 6, “Palm OS Debugging Tools,”](#) on page 89, introduces the Palm OS debugging tools.
- [Chapter 7, “Palm OS Package Builder,”](#) on page 99, describes how to use Palm OS Package Builder to create PalmSource Installer (PSI) files.
- [Appendix A, “Hints and Tips,”](#) on page 103, provides some additional information on using Palm OS Developer Suite.

Additional Resources

- **Integrated Documentation**
Most of the documentation described in this introduction is integrated into the Eclipse help system. To view titles from within Eclipse, select **Help > Help Topics** to open the Eclipse help browser.

- **Documentation Web Site**
PalmSource publishes its latest versions of documents for Palm OS developers at
<http://www.palmos.com/dev/support/docs/>
- **Training**
PalmSource and its partners host training classes for Palm OS developers. For topics and schedules, check
<http://www.palmos.com/dev/training>
- **Knowledge Base**
The Knowledge Base is a fast, web-based database of technical information. Search for frequently asked questions (FAQs), sample code, white papers, and the development documentation at
<http://www.palmos.com/dev/support/kb/>

Eclipse Workbench Documentation

A general understanding of the Eclipse Workbench will make it easier for you to use the Palm OS tools that are integrated with Eclipse. The *Eclipse Workbench User Guide* can give you more information about Eclipse:

- *Getting Started*
Includes step-by-step tutorials on how to use the Eclipse Workbench.
- *Concepts*
Describes the concepts behind most of the components in the Workbench, with cross references to related tasks and reference information.
- *Tasks*
Provides step-by-step detailed information on how to perform specific tasks in the Workbench.
- *Reference*
Documents reference information for the components of the Eclipse Workbench.

C/C++ Development Toolkit (CDT) Documentation

C/C++ Development Toolkit (CDT) provides a C/C++ editor, outlining and indexing functions, a search facility, integration hooks for a 68K compiler (`gcc`), a 68K debugger (`gdb`), and build support for standard make and managed make. *C/C++ Development User Guide* provides the following information:

- *Tutorial*

Includes step-by-step descriptions of how to create projects, source files, and makefiles, plus information on building and debugging with the CDT.

- *Concepts*

Provides background information that may help you to complete specific tasks.

- *Tasks*

Documents the procedural instructions for completing CDT-specific tasks.

- *Reference*

Provides reference information for elements of the C/C++ perspective.

PRC-Tools Documentation

PRC-Tools is a collection of tools supporting C and C++ programming. Palm OS Developer Suite uses components of PRC-Tools to compile 68K applications.

For more information about PRC-Tools, see the documentation at PRC-Tools web site (prc-tools.sourceforge.net/doc/).

Cygwin Documentation

Palm OS Developer Suite uses Cygwin components to compile and debug Palm OS Protein applications targeted for Palm OS Simulator:

- `gcc`, the C compiler
- `gdb`, the GNU Debugger

For more information about Cygwin packages, see the documentation at the Cygwin web site (www.cygwin.com).

Palm OS 68K API Documentation

If you are interested in developing 68K applications that work through PACE and that also run on earlier Palm OS releases, you can read the Palm OS 68K API documentation:

- *Palm OS Programmer's API Reference*
An API reference document that contains descriptions of all of the Palm OS 68K function calls and important data structures.
- *Palm OS Programmer's Companion*
A multi-volume guide to application programming for Palm OS 68K applications. This guide contains conceptual and "how to" information that complements *Palm OS Programmer's API Reference*.
- *Palm OS User Interface Guidelines*
A guide describing how to design applications for Palm Powered™ handhelds so that they conform to PalmSource's user interface guidelines for 68K applications.
- *Testing with Palm OS Garnet Simulator*
A guide describing how to use Palm OS Garnet Simulator to test your applications.
- *Virtual Phone User's Guide*
A guide describing how to use Virtual Phone to test telephony applications.

Palm OS Protein API Documentation

The Palm OS Protein API documentation is called the *Exploring Palm OS* series. Together, the books in this series document and explain how to use the APIs exposed to third-party developers by the fully ARM-native versions of Palm OS, beginning with Palm OS Cobalt. Each of the books in the *Exploring Palm OS* series explains one aspect of the Palm operating system, and each contains both conceptual and reference documentation for the pertinent technology.

About This Book

Additional Resources

As of this writing, the complete *Exploring Palm OS* series consists of the following titles:

- *Exploring Palm OS: Programming Basics*
- *Exploring Palm OS: Memory, Databases, and Files*
- *Exploring Palm OS: User Interface*
- *Exploring Palm OS: System Management*
- *Exploring Palm OS: Text and Localization*
- *Exploring Palm OS: Input Services*
- *Exploring Palm OS: High-Level Communications*
- *Exploring Palm OS: Low-Level Communications*
- *Exploring Palm OS: Telephony and SMS*
- *Exploring Palm OS: Multimedia*
- *Exploring Palm OS: Security and Cryptography*
- *Exploring Palm OS: Porting Applications to Palm OS Cobalt*
- *Exploring Palm OS: Palm OS File Formats*

Palm OS Developer Suite Documentation

The following tools books are part of the Palm OS Developer Suite package:

Document	Description
<i>Introduction to Palm OS Developer Suite</i>	Provides an overview of all of the Palm OS development tools: <ul style="list-style-type: none">• Compiler Tools• Resource Tools• Testing and Debugging Tools
<i>Palm OS Protein C/C++ Compiler Tools Guide</i>	Describes the tools associated with the Palm OS Protein C/C++ Compiler.
<i>Palm OS Protein C/C++ Compiler Language and Library Reference</i>	Provides reference information about the C language and runtime libraries used with the Palm OS Protein C/C++ Compiler.

Document	Description
<i>Palm OS Debugger Guide</i>	Describes how to use Palm OS Debugger.
<i>Palm OS Resource Editor Guide</i>	Describes how to use Palm OS Resource Editor to create XRD files.
<i>Palm OS Resource Tools Guide</i>	Describes how to use the Palm OS resource tools: <ul style="list-style-type: none">• GenerateXRD - migration tool• Palm OS Resource Editor - XRD editor• PalmRC - building tool• PRCMerge - building tool• PRCCompare - comparison tool• hoverlay - localization tool• PRCSign and PRCCert - code-signing tools
<i>Palm OS Resource File Formats</i>	Describes the XML formats used for XML resource definition (XRD) files. XRD files are used to define Palm OS resources, and are the input files for the Palm OS resource tools.
<i>Palm OS Cobalt Simulator Guide</i>	Describes how to use Palm OS Cobalt Simulator.
<i>Palm OS Virtual Phone Guide</i>	Describes how to use Virtual Phone.

About This Book

Additional Resources

Palm OS Developer Suite Overview

This chapter describes the process to create a Palm OS application, describes the features provided by Palm OS Developer Suite, and discusses how each of the Palm OS developer tools are used in the process.

Palm OS Developer Suite is an integrated tools package based on Eclipse 3.0.1 and the C/C++ Development Toolkit (CDT) feature, version 2.0.2. This integrated environment simplifies the process for building Palm OS applications while providing support for 68K applications, PACE native objects, and Palm OS Protein applications.

If you choose, you can also use the Palm OS developer tools independently or with other development solutions such as CodeWarrior or gcc.

This chapter covers the following topics:

- [“Palm OS Application Development Tasks”](#) on page 2
- [“Process for Building Palm OS Applications”](#) on page 3
- [“Components of Palm OS Developer Suite”](#) on page 11

Palm OS Application Development Tasks

Creating a Palm OS application is similar to creating an application for other operating systems:

Design:

Design the application by determining the functional requirements, the data requirements, and the user interface requirements.

Develop:

Write the program logic and user interface source code.

Build:

Compile the source and build the executable application.

Test:

Run the application, using testing and debugging tools to help identify how the application can be improved.

Package:

Create installable packages of your application and support files.

Design Considerations

The Palm OS platform supports three application types. One of your primary application design considerations is to decide which application type is right for your application.

- **Palm OS 68K Applications**

If your goal is to create applications that run on all Palm OS devices and on all supported Palm OS versions, then you want to create Palm OS 68K applications.

The 68K API is the set of functions that spans all Palm OS releases. The 68K functions allow application code compiled for Dragonball processors to access the capabilities of Palm OS. The 68K API is the native API for Palm OS 4 and earlier Palm OS versions. The 68K API operates in PACE (the

Palm OS Application Compatibility Environment) on Palm OS Garnet and Palm OS Cobalt.

For more information on creating 68K applications, see [“Process for Building Palm OS 68K Applications”](#) on page 4.

- **PACE Native Objects**

If you are an experienced developer accustomed to working with the 68K API but you want to take advantage of the ARM CPUs that power Palm OS Garnet and Palm OS Cobalt handhelds, then you can create PACE native objects.

A PACE native object (PNO) is an ARM-native subroutine that your 68K application calls using the `PceNativeCall()` function.

For more information on creating PACE native objects, see [“Process for Building PACE Native Objects”](#) on page 5.

- **Palm OS Protein Applications**

If you want to take advantage of the Palm OS Protein API functions and the features emerging on new Palm OS Cobalt handhelds, you want to create Palm OS Protein applications.

Palm OS Protein applications can be multithreaded, can access schema and extended databases, and can employ Palm OS Cobalt's multimedia framework. However, Palm OS Protein applications only run on Palm OS Cobalt and later.

For more information about creating Palm OS Protein applications, see [“Process for Building Palm OS Protein Applications”](#) on page 7.

Process for Building Palm OS Applications

This section provides a high-level overview of the processes for building the different types of Palm OS applications. The descriptions include references to where you can get more detailed information.

- [“Process for Building Palm OS 68K Applications”](#) on page 4
- [“Process for Building PACE Native Objects”](#) on page 5
- [“Process for Building Palm OS Protein Applications”](#) on page 7

Process for Building Palm OS 68K Applications

For building Palm OS 68K applications, Palm OS Developer Suite includes PRC-Tools. PRC-Tools includes patched versions of the GNU packages `gcc`, `binutils`, `gdb`, and various post-linker tools.

- Create a 68K C/C++ Project for your source code. You can choose either a standard make project or a managed make project.

For more information about creating projects, see “[Creating a New Project](#)” on page 21.

- Create your C/C++ source files using the C/C++ editor provided by CDT.
- Create your application’s XML resource definition (XRD) files either by using Palm OS Resource Editor or by editing the XML file using the Eclipse text editor. For more information about XRD files and the Palm OS Resource Editor, see “[Palm OS Resource Tools](#)” on page 65.
- Build your project. From Developer Suite, select **Build Project** or **Rebuild Project** to build your 68K application.

When you use Palm OS Developer Suite, you do not need to invoke the individual build tools directly. If you manually build your project, you need to take the following steps:

- Use the `m68k-palmos-gcc` compiler to compile and link your source code.
- Use the Palm OS resource compiler, `PalmRC`, to compile your XRD file into a temporary resource (TRC) file. `PalmRC` is part of the [Palm OS Resource Tools](#) component.
- Use the Palm OS resource builder, `PRCMerge`, to join your code resources, data resource, and temporary resources into a Palm OS application. `PRCMerge` is part of the [Palm OS Resource Tools](#) component.
- Test your application by running it on Palm OS Emulator, Palm OS Simulator, or a Palm Powered™ device. From Palm OS Developer Suite, select **Run > Run** to open the Run dialog box, and use the Target tab to select the run target for your application.

- Debug your application, using the debugger integrated with Palm OS Developer Suite. From Palm OS Developer Suite, select **Run > Debug** to open the Debug dialog box, and use the Target tab to select the debug target for your application.

After your application is built, you can optionally take some additional steps.

- Compare applications using PRCCompare. PRCCompare is part of the [Palm OS Resource Tools](#) component.
- Create localized overlays for your application using PRC2OVL.

For more information about building a project in Palm OS Developer Suite, see “[Building a Project](#)” on page 43 in [Chapter 2](#), “[Eclipse Workbench Integration](#).”

Multi-Segment 68K Application Support

Sample code generated by the 68K application wizard contains two files to help you create multi-segment applications, if your application is sufficiently large to require multiple segments. These files are `Sections.h` and `Sections.def`. You can read the instructions in these files on how to use them.

Process for Building PACE Native Objects

A PACE native object is an ARM-native subroutine that you call from a 68K application. The ARM-native subroutine needs to be compiled with an ARM-based compiler rather than the 68K-based compiler that you use for your 68K application.

- Create a 68K PNO C/C++ Project for your source code. You can choose either a standard make project or a managed make project.

For more information about creating projects, see “[Creating a New Project](#)” on page 21.

- Create the C/C++ source files for your 68K application and the C/C++ source files for your PNO using the C/C++ editor provided by CDT.
- Create your application’s XML resource definition (XRD) files either by using Palm OS Resource Editor or by editing the XML file using the Eclipse text editor. For more

Palm OS Developer Suite Overview

Process for Building Palm OS Applications

information about XRD files and the Palm OS Resource Editor, see “[Palm OS Resource Tools](#)” on page 65.

- Build your project. From Developer Suite, select **Build Project** or **Rebuild Project** to build your 68K PNO application.

When you use Palm OS Developer Suite, you do not need to invoke the individual build tools directly. If you manually build your project, you need to take the following steps:

- Use Palm OS Protein C/C++ Compiler, `pacc`, to compile your PNO’s C/C++ source files into ELF object files for ARM-based devices.

Note: To generate PNO targets for Palm OS Simulator, use the GNU x86 compiler instead of Palm OS Protein C/C++ Compiler. For information, see “[Using Compiler Tools with the Workbench](#)” on page 59.

- Use `m68k-palmos-gcc` to compile and link your 68K application’s source code.
- Use the Palm OS resource compiler, `PalmRC`, to compile your XRD file into a temporary resource (TRC) file. `PalmRC` is part of the Palm OS Resource Tools component.
- Use the Palm OS resource builder, `PRCMerge`, to join your code resources, data resource, and temporary resources into a Palm OS application. The ELF object files from your PNOs are included as code resources when your 68K application is linked together.
- Test your application by running it on Palm OS Simulator or a Palm Powered™ device. From Palm OS Developer Suite, select **Run > Run** to open the Run dialog box, and use the Target tab to select the run target for your application.
- Debug your application, using the debugger integrated with Palm OS Developer Suite. From Palm OS Developer Suite, select **Run > Debug** to open the Debug dialog box, and use the Target tab to select the debug target for your application.

After your application is built, you can optionally take some additional steps.

- Compare applications using `PRCCompare`. `PRCCompare` is part of the Palm OS Resource Tools component.

- Create localized overlays for your application using PRC2OVL.

For more information about building a project in Palm OS Developer Suite, see “[Building a Project](#)” on page 43 in [Chapter 2](#), “[Eclipse Workbench Integration](#).”

Process for Building Palm OS Protein Applications

[Figure 1.1](#) on page 10 shows the detailed steps for building a Palm OS Protein application. Some of these steps may be optional, depending on the application.

- Create a Palm OS Protein C/C++ Project for your source code. You can choose either a standard make project or a managed make project.

For more information about creating projects, see “[Creating a New Project](#)” on page 21.

- Create the C/C++ source files for your Palm OS Protein application using the C/C++ editor provided by CDT.
- Create your application’s XML resource definition (XRD) files either by using Palm OS Resource Editor or by editing the XML file using the Eclipse text editor. For more information about XRD files and the Palm OS Resource Editor, see “[Palm OS Resource Tools](#)” on page 65.
- Create a shared library definition (SLD) file, if necessary, to specify the entry points in your application or shared library. For applications, all code templates already create the required SLD file. But if you use an empty project to import existing source code, you need to create the SLD file.
- Build your project. From Developer Suite, select **Build Project** or **Rebuild Project** to build your Palm OS Protein application.

When you use Palm OS Developer Suite, you do not need to invoke the individual build tools directly. If you manually build your project, you need to take the following steps:

Palm OS Developer Suite Overview

Process for Building Palm OS Applications

- Use Palm OS Protein C/C++ Compiler, `pacc`, to compile your C/C++ source files into ELF object files. `pacc` is part of the [Palm OS Compiler Tools](#) component.

Note: To generate targets for Palm OS Simulator, use the GNU x86 compiler instead of Palm OS Protein C/C++ Compiler. For information, see “[Using Compiler Tools with the Workbench](#)” on page 59.

- Use the Palm OS resource compiler, `PalmRC`, to compile your XRD file into a temporary resource (TRC) file. `PalmRC` is part of the [Palm OS Resource Tools](#) component.
- Use the Palm OS shared library tool, `pslib`, to compile your SLD file into a ELF object file.
- Use the Palm OS linker, `palink`, to link your ELF object files, and use the Palm OS post linker, `pelf2bin`, to prepare the files to be merged into a Palm OS application.
- Use the Palm OS resource builder, `PRCMerge`, to join your code resources, data resource, and temporary resources into a Palm OS application. `PRCMerge` is part of the [Palm OS Resource Tools](#) component.
- Test your application by running it on Palm OS Simulator or a Palm Powered™ device. From Palm OS Developer Suite, select **Run > Run** to open the Run dialog box, and use the Target tab to select the run target for your application.
- Debug your application, using the debugger integrated with Palm OS Developer Suite. From Palm OS Developer Suite, select **Run > Debug** to open the Debug dialog box, and use the Target tab to select the debug target for your application.

After your application is built, you can optionally take some additional steps. (These steps are not shown in [Figure 1.1](#).)

- Compare applications using `PRCCompare`. `PRCCompare` is part of the [Palm OS Resource Tools](#) component.
- Localize your application using the Palm OS localization tool, `hoverlay`. `hoverlay` is part of the [Palm OS Resource Tools](#) component.
- Create and embed a digital signature and associated certificates in your application using the code signing tools,

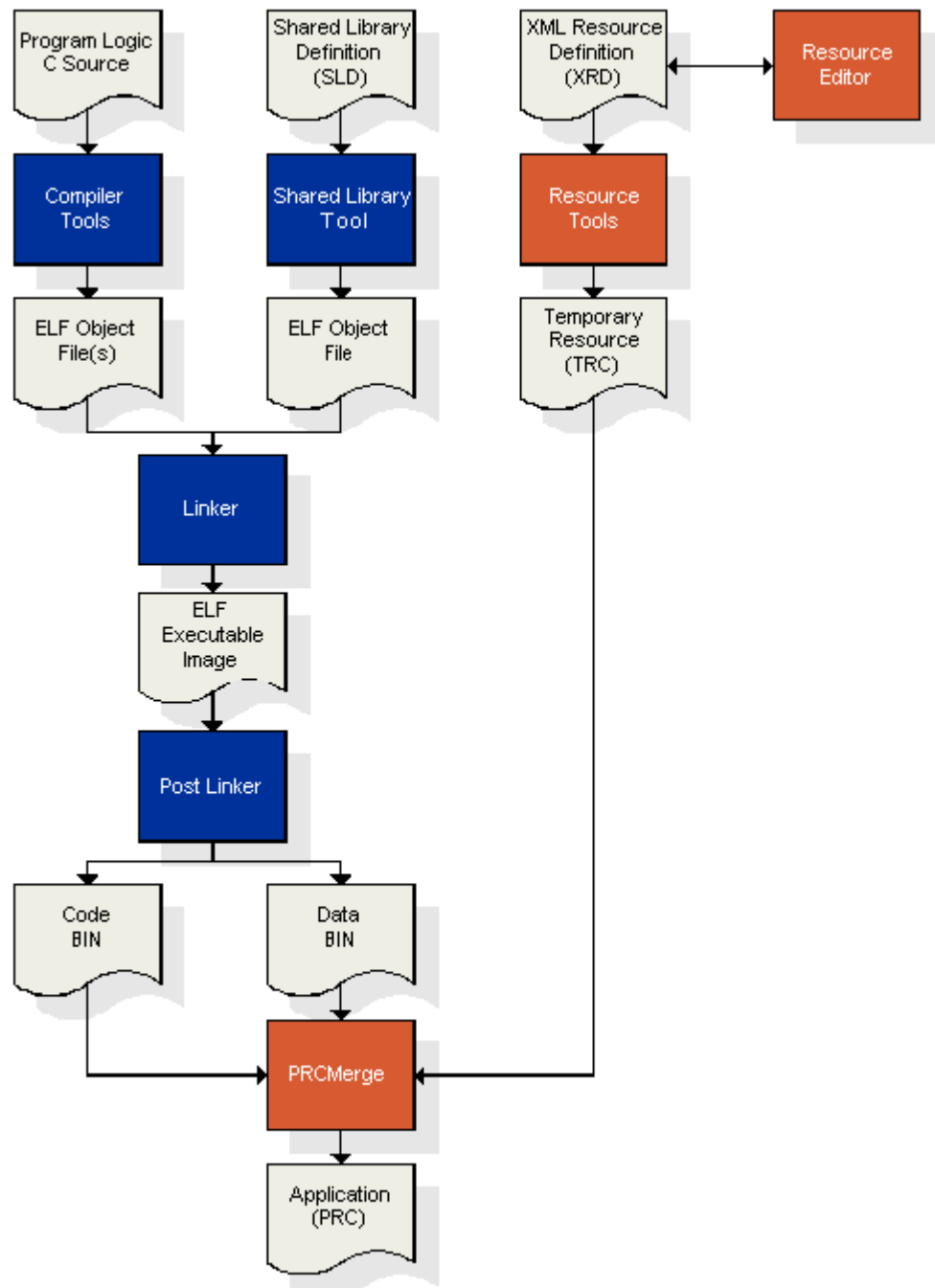
PRCCert and PRCSign. PRCCert and PRCSign are included with the [Palm OS Resource Tools](#) component.

For more information about building a project in Palm OS Developer Suite, see “[Building a Project](#)” on page 43 in [Chapter 2](#), “[Eclipse Workbench Integration](#).”

Palm OS Developer Suite Overview

Process for Building Palm OS Applications

Figure 1.1 Palm OS Protein Application Development Process



Components of Palm OS Developer Suite

The Palm OS Developer Suite package consists of the following components:

- [Workbench Integration](#)
- [Palm OS Compiler Tools](#)
- [PRC-Tools](#)
- [Palm OS Resource Tools](#)
- [Testing and Debugging Tools](#)
- [Palm OS Package Builder](#)

This section provides a brief overview of each of these components. For a more detailed look into each component, see the remaining chapters of this document.

Workbench Integration

The Workbench integration consists of wizards, project files, libraries, and documentation that allow you to use the Palm OS developer tools in an Eclipse-integrated development environment.

For more information about this component, see [Chapter 2, “Eclipse Workbench Integration,”](#) on page 17.

Palm OS Compiler Tools

The Palm OS Compiler Tools component consists of an ANSI C compliant C compiler, a linker, a librarian, and utility tools that you can use to build Palm OS Protein applications:

pacc:

Palm OS Protein C/C++ Compiler. Compiles C/C++ source files and calls the other compiler tools to produce ELF object files.

paasm:

Palm OS assembler. Called by pacc to produce ELF object files.

Palm OS Developer Suite Overview

Components of Palm OS Developer Suite

palink:

Palm OS linker. Called by `pacc` to produce ELF object files.

palib:

Palm OS librarian. Creates libraries of ELF object files.

elfdump:

A utility tool for viewing the contents of ELF object files.

For more information about this component, see [Chapter 3](#), “[Palm OS Compiler Tools](#),” on page 59.

PRC-Tools

PRC-Tools is a collection of tools supporting C and C++ programming for Palm OS application development. Palm OS Developer Suite uses components of PRC-Tools to compile 68K applications.

gcc: The GNU Compiler Collection, `gcc`, which is a collection of compiler front ends and libraries.

Palm OS Developer Suite specifically uses the `m68k-palmos-gcc` compiler for 68K applications.

as: The GNU assembler, `as`, which is a component of GNU `binutils`.

ld: The GNU linker, `ld`, which is a component of GNU `binutils`.

For more information about this component, see the documentation at PRC-Tools web site (<http://prc-tools.sourceforge.net/doc/>).

Cygwin Packages

Cygwin includes many packages for developing Windows applications. Palm OS Developer Suite uses Cygwin components to compile and debug Palm OS Protein applications targeted for Palm OS Simulator.

gcc: The GNU Compiler Collection, `gcc`, which is a collection of compiler front ends and libraries.

Palm OS Developer Suite specifically uses the `i686-pc-cygwin-gcc-3.3.1` compiler for Palm OS Protein applications targeted for Palm OS Simulator.

as: The GNU assembler, `as`, which is a component of GNU `binutils`.

ld: The GNU linker, `ld`, which is a component of GNU `binutils`.

pxgdb: The GNU Project symbolic debugger for Windows, `pxgdb`, which supports debugging of Palm OS applications that have been compiled to run on Palm OS Simulator.

Palm OS Resource Tools

NOTE: XML resource definition files are called XRD files. For information about XRD files, see the book *Palm OS Resource File Formats*.

The Palm OS Resource Tools component includes tools for editing, migrating, building, comparing, localizing, and securing resources for Palm OS applications:

Palm OS Resource Editor:

A visual builder for designing Palm OS user interfaces. Palm OS Resource Editor creates and edits XML resource definition files.

GenerateXRD:

Converts 68K-based PRC files and Macintosh RSRC files into XML that can be compiled using `PalmRC`.

PalmRC:

Compiles XRD files into TRC files that can be merged into applications using `PRCMerge`.

Palm OS Developer Suite Overview

Components of Palm OS Developer Suite

PRCMerge:

Combines TRC files with code and data resources to make Palm OS applications.

PRCCompare:

Compares two Palm OS applications (PRCs, BPRCs, OPRCs, or TRCs).

hoverlay:

Creates overlays for localized versions of Palm OS Protein applications.

PRC2OVL:

Creates overlays for localized versions of 68K applications.

PRCCert:

Creates RSA key pairs and digital certificates for Palm OS applications.

PRCSign:

Digitally signs applications or embeds digital signatures in applications.

For more information about the resource tools, see [Chapter 4](#), “[Palm OS Resource Tools](#),” on page 65.

Testing and Debugging Tools

Palm OS Developer Suite includes a debugger, a device simulator, and a mobile telephone simulator.

Palm OS Emulator:

A hardware emulator program for the Palm Powered™ platform. Palm OS Emulator emulates device hardware in software, providing you with the ability to test and debug Palm OS software on a desktop computer.

You can use Palm OS Emulator to test 68K applications; Palm OS Emulator does not run ARM-native code.

Palm OS Simulator:

Palm OS recompiled to run on Windows. Palm OS Simulator is simulation tool that allows you to test your Palm OS applications as if they were running on a handheld device. By using Palm Reporter with Palm OS Simulator, you can perform real time trace analysis of your application.

Palm OS Garnet Simulator is Palm OS Garnet recompiled to run on Windows; **Palm OS Cobalt Simulator** is Palm OS Cobalt recompiled to run on Windows.

You can use Palm OS Cobalt Simulator to test Palm OS Protein applications, but the code must be recompiled to run on Windows.

Palm OS Reporter:

Palm OS Reporter is a trace utility that can be used with Palm OS Simulator and Palm OS Emulator. As an application runs on Simulator or Emulator, the application sends information in real time to Reporter. This information can help pinpoint problems that might be hard to identify when executing code step-by-step or when specifying breakpoints.

Virtual Phone:

A mobile telephone simulator that allows you to test Palm OS telephony applications. You can use Virtual Phone with Palm OS Emulator, Palm OS Garnet Simulator, and Palm OS Cobalt Simulator.

Palm OS Debugger:

A full-function, source-level debug tool that you can use to debug Palm OS Protein and 68K applications and shared libraries.

For more information about these components, see [Chapter 6](#), “[Palm OS Debugging Tools](#),” on page 89.

Palm OS Package Builder

Palm OS Developer Suite includes support for building PalmSource Installer (PSI) files using Palm OS Package Builder. Palm OS Package Builder is a developer tool that can automatically create

Palm OS Developer Suite Overview

Components of Palm OS Developer Suite

packages for “over-the-air” delivery of Palm OS applications. PalmSource Install packages can specify device-specific features, such as screen sizes or language version, as well as creating a full package for download to the user's desktop. The full package includes all languages and all screen sizes, plus any desktop components or other files too big for OTA delivery.

For more information about Palm OS Package Builder integration, see [Chapter 7, “Palm OS Package Builder,”](#) on page 99.

Eclipse Workbench Integration

This chapter describes how to use the Palm OS tools in the Eclipse Workbench integrated development environment.

Using the Eclipse Workbench, you create a new project using a wizard, edit source files using the code and text editors, and build applications from the standard development environment menu. You can even launch Palm OS Resource Editor and the Palm OS Simulators from the development environment.

- [“Using the Palm OS C/C++ Development Perspective”](#) on page 18
- [“Using Projects to Manage Application Development”](#) on page 21
- [“Creating a New Project”](#) on page 21
- [“Importing Files into a Project”](#) on page 36
- [“Editing Project Source Files”](#) on page 42
- [“Building a Project”](#) on page 43
- [“Modifying Tool Properties”](#) on page 47
- [“Launching Other Palm OS Tools”](#) on page 50
- [“Updating Palm OS Developer Suite”](#) on page 52

Using the Palm OS C/C++ Development Perspective

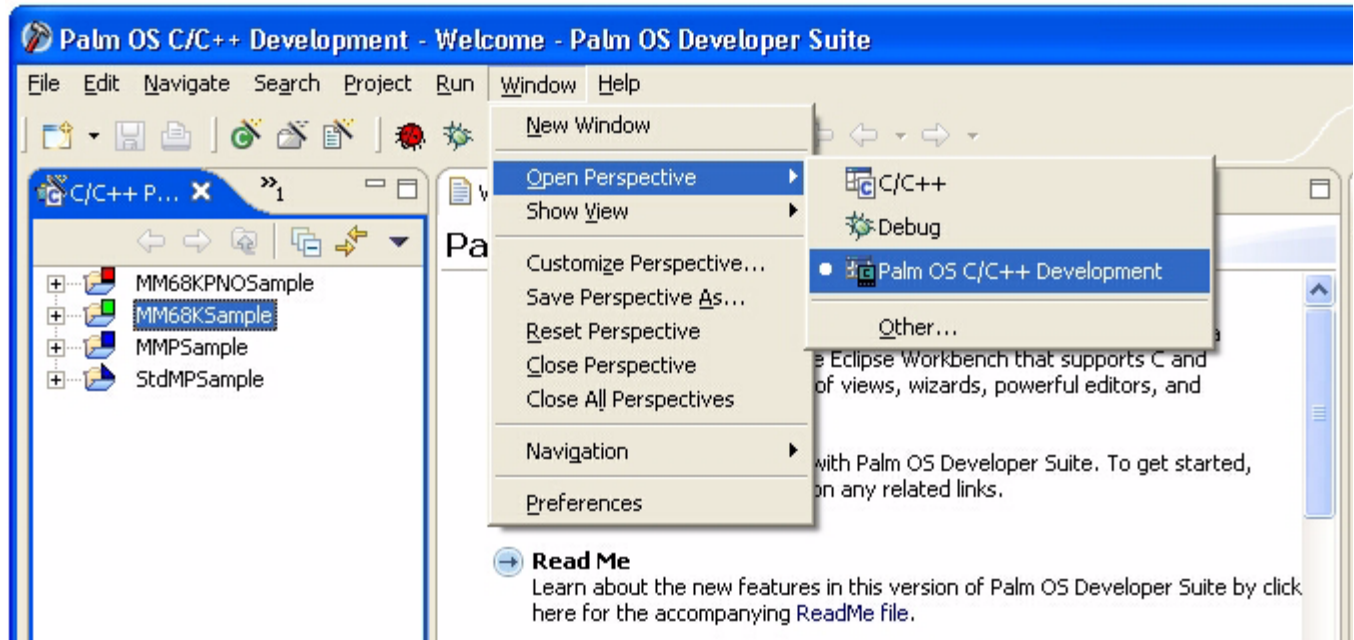
Developer Suite uses the Eclipse Workbench as its integration platform. The Eclipse Workbench uses **perspectives** to define many of the attributes and functions available in the Workbench window. For example, perspectives determine the initial layout of windows, what functions appear in certain menus and toolbars, and what editors and tools are integrated.

The functions provided by Developer Suite are provided by the Palm OS C/C++ Development perspective. The Palm OS C/C++ Development perspective gives you access to all of the components described in [Chapter 1, “Palm OS Developer Suite Overview,”](#) on page 1.

Setting the Workbench Perspective

When you first install Developer Suite, the perspective is set to Palm OS C/C++ Development perspective for you. To ensure that this perspective is the selected perspective, select **Window > Open Perspective > Palm OS C/C++ Development** as shown in [Figure 2.1](#) on page 19.

Figure 2.1 Palm OS C/C++ Development perspective



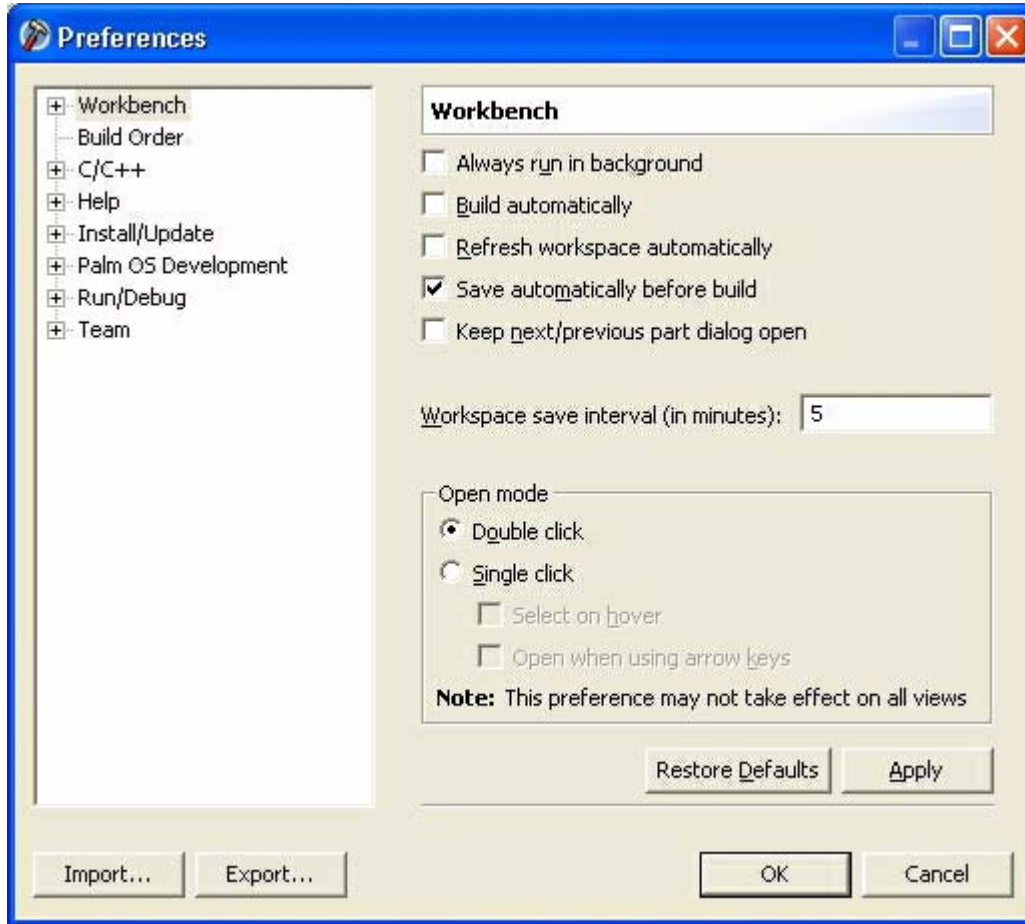
Setting Workbench Preferences

Before starting your development work, you may want to start by reviewing the default Workbench preferences. Select **Window > Preferences** and click Workbench to view the Workbench preferences, as shown in [Figure 2.2](#) on page 20.

Eclipse Workbench Integration

Using the Palm OS C/C++ Development Perspective

Figure 2.2 Workbench Preferences



Build automatically:

By default, the option Build automatically is selected, which means that every time you save a source file or a resource file, the build system is automatically invoked to build the project. If you like to save your files frequently and do not want the build system invoked every time you save, you may prefer to turn this setting off.

Save automatically before build:

In most cases, you want to turn this setting on. The build system builds the most recently saved versions of the source. If you have this setting turned off and invoke the build system without saving your source files, then the build

system does not include the changes you have made since the last save.

Using Projects to Manage Application Development

Developer Suite uses projects to manage most aspects of application development, including source file creation and application building, running, and debugging.

Using Workspaces

By default, the Eclipse Workbench stores project files (projects, folders, and files) in a subdirectory of the location of where you installed the Workbench. This sub-directory is called the “workspace” subdirectory.

When you create a project, you can select a different directory. However, the Eclipse Workbench assumes that the directory you specify is an Eclipse-generated directory. It is recommended that you use the default location as your workspace. Specifically, it is recommended that you do not create projects on the desktop.

Rather than changing the project’s workspace directory, you should accept the default location and import your files into the project, which moves them to the workspace location.

Using Makefiles

Developer Suite projects rely on make files. Either use **Managed Make**, which generates makefiles, or use **Standard Make**, which requires you to write makefiles yourself or to modify the makefiles generated when a Palm OS Standard Make project is created. For more information about makefiles, see “[Select Standard Make or Managed Make](#)” on page 23.

Creating a New Project

To create a new project, follow these steps:

- [Select an Application Type](#)

- [Select Standard Make or Managed Make](#)
- [Use the New Project Wizard to Create a Project](#)

Select an Application Type

Before you create a project, you first consider which type of application you want to create.

Palm OS 68K Application

If your goal is to create applications that run on all Palm OS devices and on all supported Palm OS versions, then you want to create Palm OS 68K applications.

The 68K API is the set of functions that spans all Palm OS releases. The 68K functions allow application code compiled for Dragonball processors to access the capabilities of Palm OS. The 68K API is the native API for Palm OS 4 and earlier Palm OS versions. The 68K API operates in PACE (the Palm OS Application Compatibility Environment) on Palm OS Garnet and Palm OS Cobalt.

Palm OS 68K Application with PACE Native Objects

If you are an experienced developer accustomed to working with the 68K API but you want to take advantage of the ARM CPUs that power Palm OS Garnet and Palm OS Cobalt handhelds, then you can create PACE native objects.

A PACE native object (PNO) is an ARM-native subroutine that your 68K application calls using the `PceNativeCall()` function.

Palm OS Protein Application

If you want to take advantage of the Palm OS Protein API functions and the features emerging on new Palm OS Cobalt handhelds, you want to create Palm OS Protein applications.

Palm OS Protein applications can be multithreaded, can access schema and extended databases, and can employ Palm OS Cobalt's multimedia framework. However, Palm OS Protein applications only run on Palm OS Cobalt and later.

Select Standard Make or Managed Make

Next, decide whether you want to use standard make or managed make to build your application.

Standard Make

In the standard make project wizards, Developer Suite provides a generic set of makefiles that you can modify and tailor for your specific application build. The standard make makefiles are user-defined and user-controlled. You have to manually update when you make changes to your project (adding files, removing files, changing the output, etc.). Developer Suite essentially invokes make to build the standard make projects.

You also have the option of not using the Developer Suite supplied makefiles, but the you must modify your makefiles to meet certain Developer Suite requirements. Developer Suite requires that the following variables be defined in the makefile:

ARTIFACT_NAME

Represents the root name of the target of the build, without any extensions. For example, if you are building the application `sample.prc`, then `ARTIFACT_NAME` must have a value `sample`.

DEBUG_OR_RELEASE

The valid values for this variable are `Debug` or `Release`.

This variable indicates whether the current build is a debug build or a release build.

TARGET_PLATFORM

The valid values for this variable are `Device` or `Simulator`.

Standard make projects build applications, shared libraries, and static libraries for two targets: device and simulator. To select the target for your project build, edit the make file provided with the project and change the `TARGET_PLATFORM` variable to either `Device` (actual hardware device or emulator), or `Simulator` (Palm OS Garnet or Cobalt Simulator).

Device builds and Simulator builds produce output in different files and in different formats; Developer Suite must be able to differentiate the two.

Eclipse Workbench Integration

Creating a New Project

`DEBUG_DEVICE_OUTPUT_DIR`

Indicates the directory relative to the project's directory that contains the final output for a debug build targeting a device platform.

`RELEASE_DEVICE_OUTPUT_DIR`

Indicates the directory relative to the project's directory that contains the final output for a release build targeting a device platform.

`DEBUG_SIMULATOR_OUTPUT_DIR`

Indicates the directory relative to the project's directory that contains the final output for a debug build targeting a simulator platform.

`RELEASE_SIMULATOR_OUTPUT_DIR`

Indicates the directory relative to the project's directory that contains the final output for a release build targeting a simulator platform.

`SDK_LOCATION`

Specifies the location of the Palm OS SDK used for this project.

`TOOLS_DIR`

Specifies the location of the Palm OS tools used for this project.

Whenever you build a project, the Palm OS C/C++ Development perspective dynamically determines the location of the Palm OS SDKs and tools used for the project build. These locations are included in the generated file called `auto-generated.mk`, which is placed in the project's root directory. This `auto-generated.mk` file is used by the default makefiles generated with Palm OS projects. You may include this file in your own make files if you want to determine the values of `SDK_LOCATION` and `TOOLS_DIR` at compile time.

The variables in makefiles must be direct assignments of values with no variable substitution. For example,

```
ARTIFACT_NAME = a
```

```
ARTIFACT_NAME := b
```

are valid in a Developer Suite makefile, but

```
ARTIFACT_NAME = $(SOME_OTHER_VARIABLE)
```

is not valid.

You can review the sample code produced by Developer Suite's wizards for more examples of these variables in makefiles.

Managed Make

Managed Make dynamically generates your makefile based on the contents of your project folders. Managed make “watches” your project and automatically updates the makefile when you add files or remove files in your project. To make additional project updates, you use the provided preference panels.

To select build targets (Device or Simulator targets) for managed make projects, select a project and select **Project > Properties**. In the Properties dialog box, select the C/C++ Build settings group, and change the Active configuration setting appropriately.

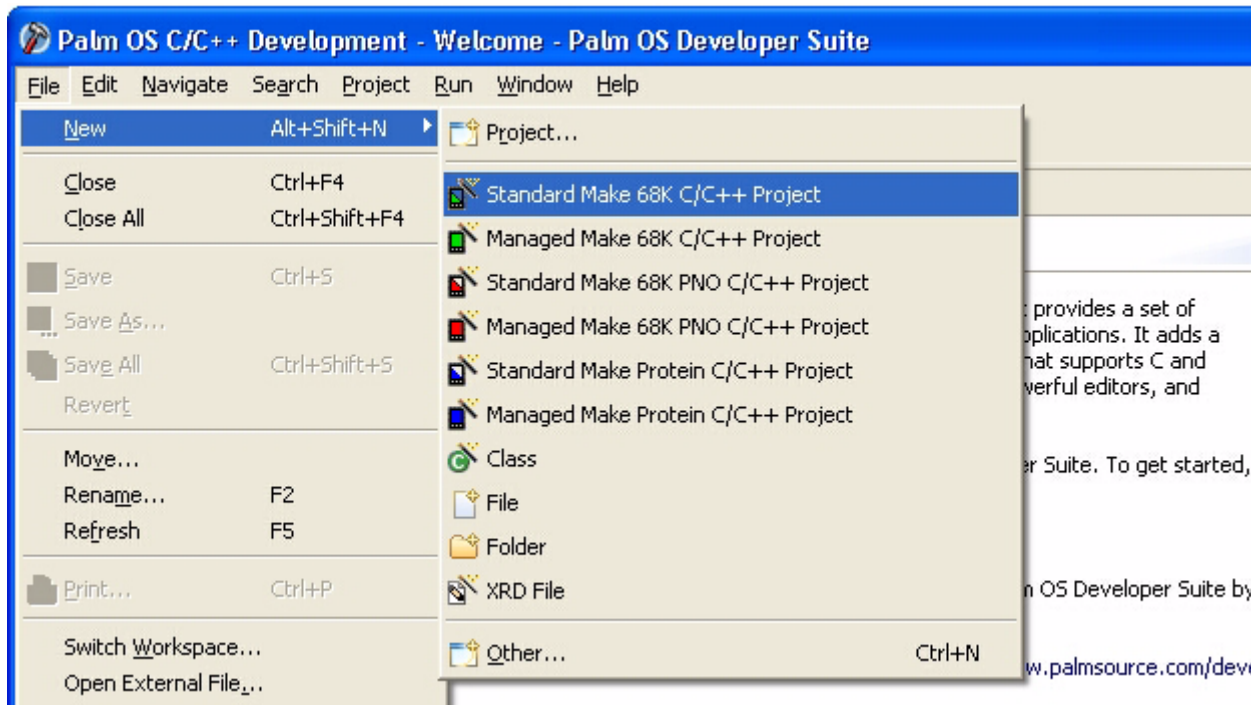
Use the New Project Wizard to Create a Project

To create a new Palm OS project in the Workbench, select **File > New** and the application type you want to create, as shown in [Figure 2.3](#) on page 26.

Eclipse Workbench Integration

Creating a New Project

Figure 2.3 Creating a new project



- **Standard Make 68K C/C++ Project**
This is a project for a Palm OS 68K application, written in C/C++, that use a Standard Make makefile.
- **Managed Make 68K C/C++ Project**
This is a project for a Palm OS 68K application, written in C/C++, that uses a Managed Make makefile.
- **Standard Make 68K PNO C/C++ Project**
This is a project for a Palm OS 68K application with an ARM-based PNO, written in C/C++, that uses a Standard Make makefile.
- **Managed Make 68K PNO C/C++ Project**
This is a project for a Palm OS 68K application with an ARM-based PNO, written in C/C++, that uses a Managed Make makefile.
- **Standard Make Protein C/C++ Project**
This is a project for a Palm OS Protein application, written in C/C++, that uses a Standard Make makefile.

- Managed Make Protein C/C++ Project

This is a project for a Palm OS Protein application, written in C/C++, that uses a Managed Make makefile.

Eclipse Workbench Integration

Creating a New Project

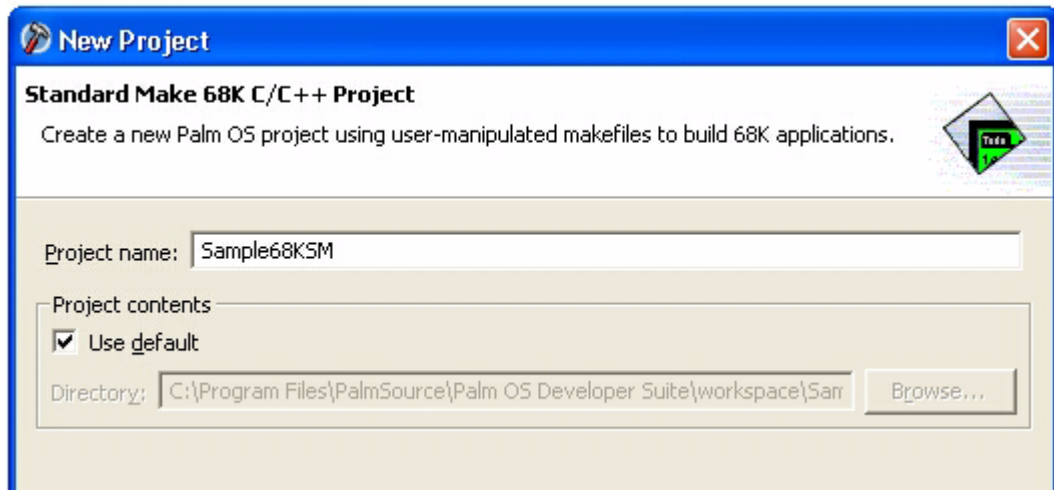
Enter Project Name and Location

When you select your application type, the New Project wizard opens, as shown in [Figure 2.4](#).

Enter a project name and select the location for your project files. (As described in “[Using Workspaces](#)” on page 21, it is recommended that you accept the default location.)

For each page of the wizard, click **Next** to move to the next page.

Figure 2.4 New Project wizard project name and location



Enter Palm OS Settings

Enter Palm OS settings, as shown in [Figure 2.5](#).

Figure 2.5 New Project wizard Palm OS settings

New Project

Palm OS Settings

Creator ID should be changed from the default starter value.

Test

Project Output:

- ☒ Application (prc)
- ☐ Shared Library
- ☐ Static Library
- ☐ Database (pdb)

Output Name without extension: Sample68K5M

Database Type: appl

Database Name: Sample68K5M

Database Version: 1

Creator ID: STRT

Note: Register your creator ID at <http://www.palmos.com/dev/creatorid>

Database Settings:

- ☒ Allow Backup on Sync
- ☐ Prevent Copy
- ☐ Hidden
- ☐ Reset on Install
- ☐ Bundle its Databases

Eclipse Workbench Integration

Creating a New Project

Project Output Type: Select the output type for the project.

Application - A Palm OS application. For information, see *Exploring Palm OS: Programming Basics*.

Shared Library - An executable module that is compiled and linked separately. For more information, see *Exploring Palm OS: System Management*, Chapter 6 “Shared Libraries.”

Static Library - A collection of object files that you can link into another Palm OS application.

Database (pdb) - A PDB is a record database generally used to store data for an application. For information, see *Exploring Palm OS: Palm OS File Formats*.

According to the normal Eclipse build paradigm, a project produces only one output file. Creating Database projects gives you a way to add PDB files to an application’s debug session by using the **Files to install** field in the Debug dialog box for the Palm OS Application configuration.

Note that the **Database (pdb)** selection is supported for standard make projects only.

Output Name without extension: The name for your project and related output files. This field is set to the project name entered in the **Name** field shown in [Figure 2.4](#) on page 28.

Database Type: For Palm OS applications (PRC databases), this field has the value `appl`.

For Palm OS shared libraries, this field has the value `libr`.

Database Version: The application-specific version of the database layout.

Database Name: The database name stored in the PRC database header. This is normally the name of the application.

Creator ID: Each Palm OS application is uniquely identified by a combination of its name and a four-byte creator ID. Enter the creator ID that you have registered with PalmSource. For information on creator IDs, see *Exploring Palm OS: Programming Basics*.

Database Settings: Select how you want the database attribute constants set for your database. See *Palm OS Programmer’s*

API Reference for more information on the database attribute constants described.

Allow Backup on Sync - Select to set the `dmHdrAttrBackup` constant.

Hidden - Select to set the `dmHdrAttrHidden` constant.

Bundle its Databases - Select to set the `dmHdrAttrBundle` constant.

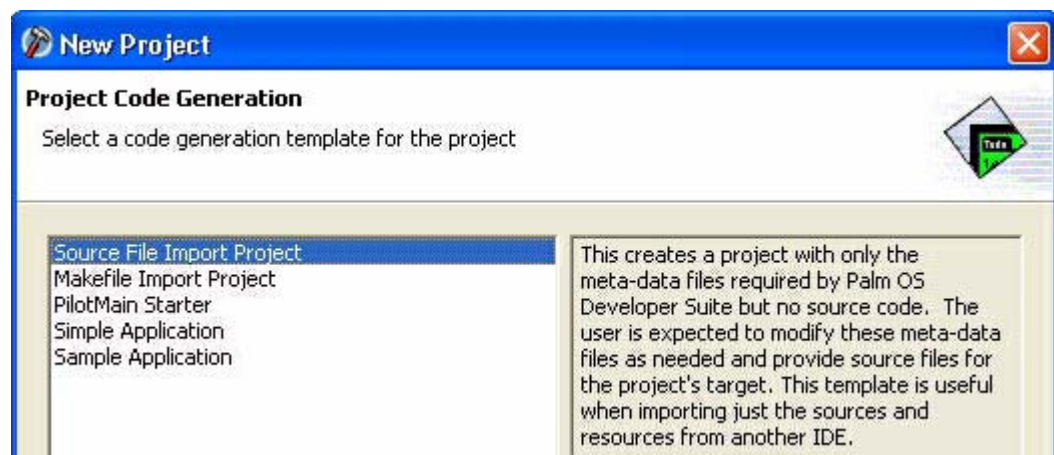
Prevent Copy - Select to set the `dmHdrAttrCopyPrevention` constant.

Reset on Install - Select to set the `dmHdrAttrResetAfterInstall` constant.

Project Code Generation

Select what code you want generated for your project, as shown in [Figure 2.6](#).

Figure 2.6 New Project wizard code generation



NOTE: The code generation template items listed are dependent on the type of project you are creating.

Source File Import Project: Provides meta-data files, such as a makefile and `make-engine.mk` file, that you can use as a base for your project. This template is useful for importing existing source files.

Eclipse Workbench Integration

Creating a New Project

Make File Import Project:

Creates an empty project without the meta-data files. This selection is useful for importing and modifying existing Palm OS makefile based projects such as PRC-tools projects, when you don't want a new makefile created.

PilotMain Starter: Provides a single source file with the required `PilotMain()` function.

Simple: Provides a simple Hello World application with `PilotMain()`, a frame, and event loop handling.

Sample: Provides a fully functioning sample application, a Puzzle game.

Additional Project Settings

For managed make, you can choose to reference an existing project, as shown in [Figure 2.7](#) on page 32. These are inherited from the CDT component, though they have additional values from the Palm OS C/C++ Perspective. For information on the CDT settings, see the C/C++ Development Toolkit (CDT) documentation.

If you don't want to change the build settings, you can click **Finish** to take the default build settings.

Figure 2.7 New Project wizard build settings



PACE Native Object Projects

In this version of Developer Suite, the PNO projects require a very specific directory structure, as created by the PNO project wizards. You should not alter the directory structure that the New Project wizard creates.

In addition, Developer Suite does not support PACE Native Objects with global data. If you attempt to use global data in a PNO, you receive a warning message from the Palm OS post linker, and the execution of the PNO can have unpredictable results.

Changing Project Settings for PNO Resources

By default, a resource type of ARMC is used for PNO resources. The default IDs of PNO resources start at hex value 1000. These values are the normal convention for PNOs and are used as the defaults for project creation.

For device target builds, you can change the PNO resource type and IDs. (For simulator target builds, the PNO code is compiled into a DLL, so this change is not pertinent to simulator target builds.)

Standard Make

For PNO standard make projects and makefiles provided by Palm OS Developer Suite, you change the resource type and ID specified in the makefile. See the `makefile` for instructions on how to specify the PNO resource values in the appropriate sections.

Managed Make

For PNO managed make projects, change the resource type and ID through project properties.

Changing the resource type

To change the resource type, make two modifications:

1. Rename the directory name containing the PNO resource code to the new resource type. By default, the directory is named `NativeCode/ARMC1000`.

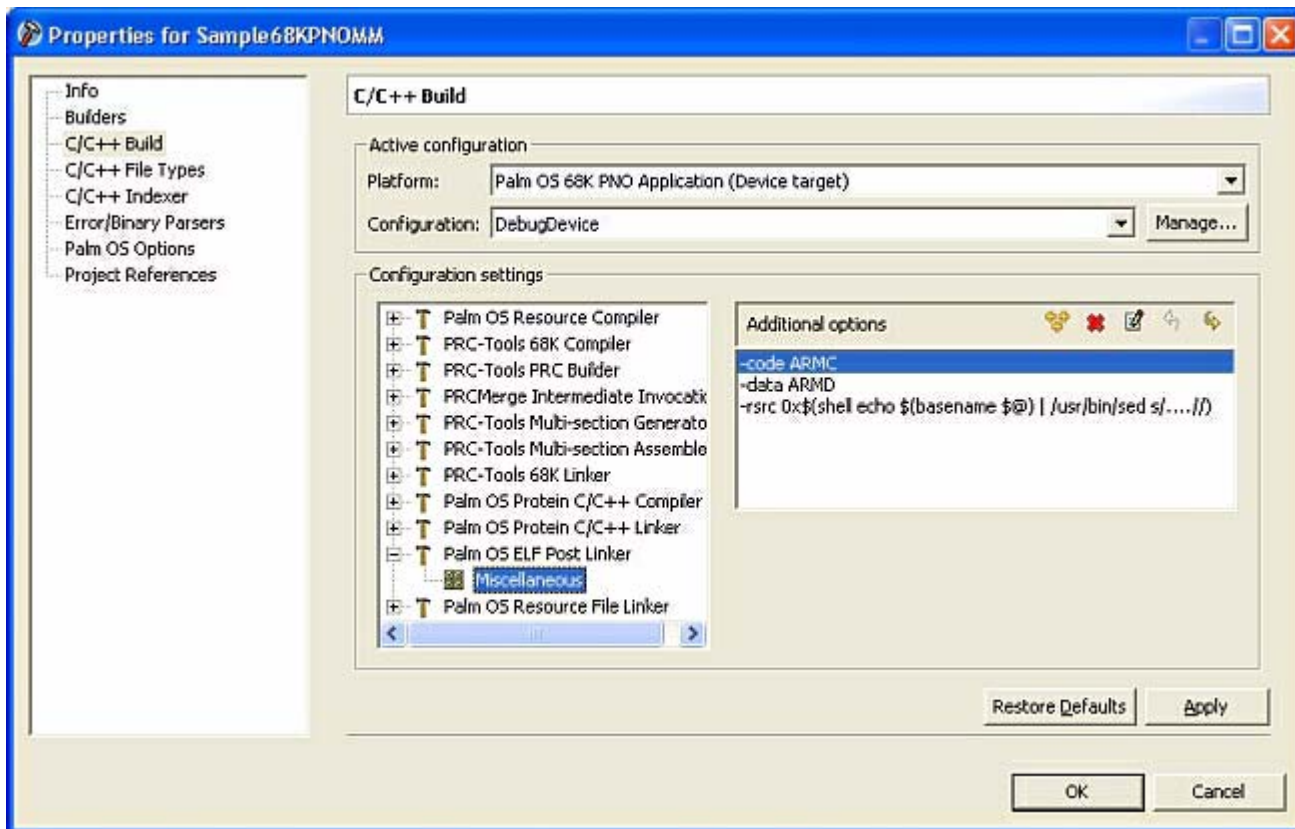
For example, to change to a resource type of `WXYZ`, rename the directory to `NativeCode/WXYZ1000`.

Eclipse Workbench Integration

PACE Native Object Projects

2. Change the project properties for the Palm OS Elf Post Linker by doing the following:
 - Open the Properties dialog box by selecting the project and then selecting **Project > Properties**. The Properties dialog box opens, as shown in [Figure 2.8](#) on page 34.

Figure 2.8 Managed make PNO Properties dialog box



- Select C/C++ Build.
- For Platform, select Palm OS 68K PNO Application (Device target).
- In the Configuration Settings, expand Palm OS ELF Post Linker and select Miscellaneous.

- Change the setting for `-code` to match the new resource type. By default, this setting is this:

`-code ARMC`

For example, to change the resource type to `WXYZ`, change the setting to this:

`-code WXYZ`

- Change the setting for `-data` to match the new resource type. By default, this setting is this:

`-data ARMD`

For example, to change the resource type to `WXYD`, change this setting to this:

`-data WXYD`

- Change the `-rsrc` option to match the new resource type. By default, this setting is

`-rsrc 0x$(subst ARMC, , $*)`

- For example, to change the resource type to `WXYZ`, change this setting to this:

`-rsrc 0x$(subst WXYZ, , $*)`

Changing the resource ID

To change the resource ID, simply rename the directory name containing the PNO resource code to the new resource ID. For example, to change to a resource ID of hex 2000, rename the directory to `NativeCode/ARMC2000`.

Shared Library Projects

Creating a shared library project is similar to creating an application project. However, in the New Project wizard:

- In the second page of the wizard, select Shared Library for the Project output setting.
- In the third page of the wizard, select Shared Library Starter. For 68K shared libraries, this ensures that the DEF file gets created; for Palm OS Protein shared libraries, this ensures that the SLD file gets created.

When you create a project that uses a shared library, you define your shared library's project as a Project Reference for your application.

Importing Files into a Project

If you have existing C/C++ application code, you can create a new project and import your code into the project. Developer Suite uses the Workbench's import support to allow you to add files to a project. This means that you can add files to a project by:

- dragging and dropping from the file system into the Navigator view, or
- copying and pasting from the file system into the Navigator view, or
- using the Import wizard.

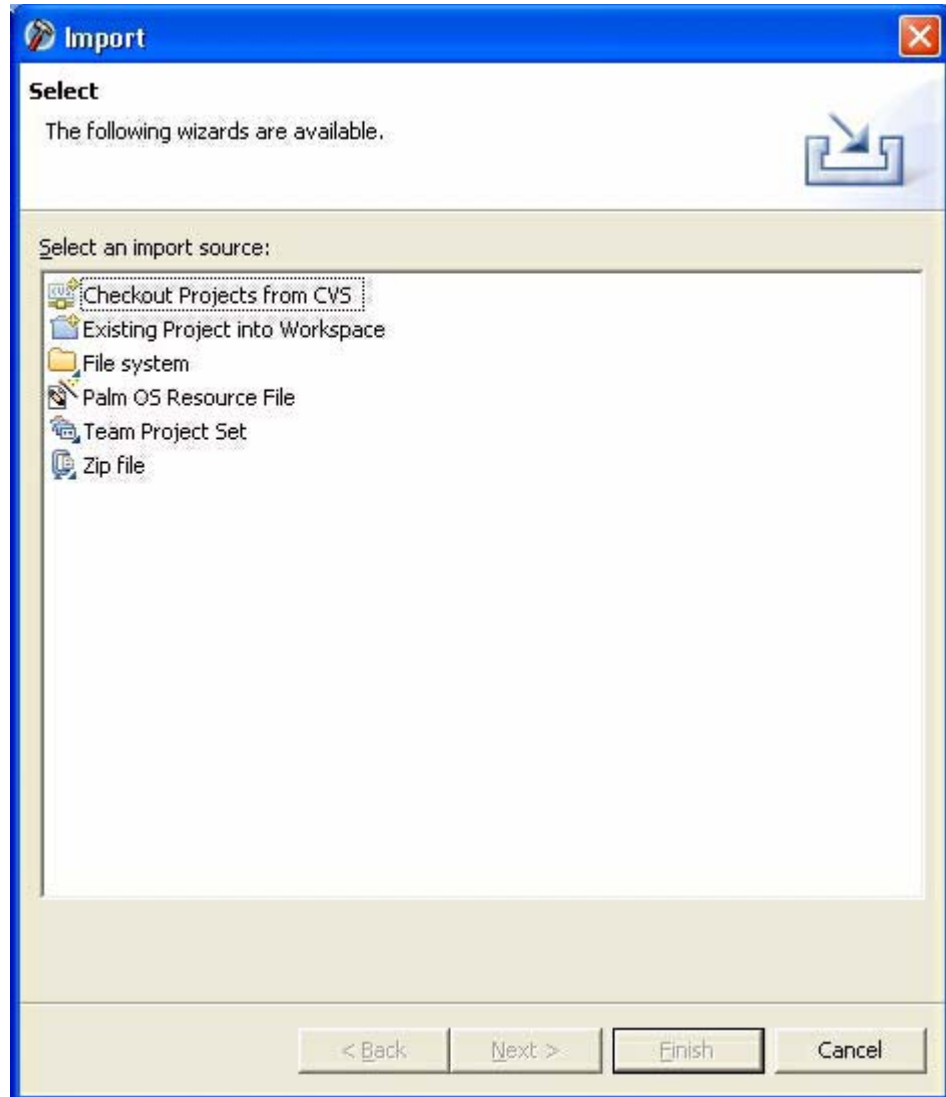
The Palm OS C/C++ Development perspective inherits the Workbench-implemented support for importing existing projects, files from the file system, team project sets, and zip files. The Palm OS C/C++ Development perspective adds a wizard for importing Palm OS resource files.

Importing 68K Source Files

Importing source code from either CodeWarrior projects or from PRC-Tools is relatively straightforward.

- Start by creating a Make File Import Project, as described in the section "[Use the New Project Wizard to Create a Project](#)" on page 25.
- Right-click on the project's icon in the C/C++ Projects view and select Import to open the Import Wizard, shown in [Figure 2.9](#) on page 37.

Figure 2.9 Import Wizard

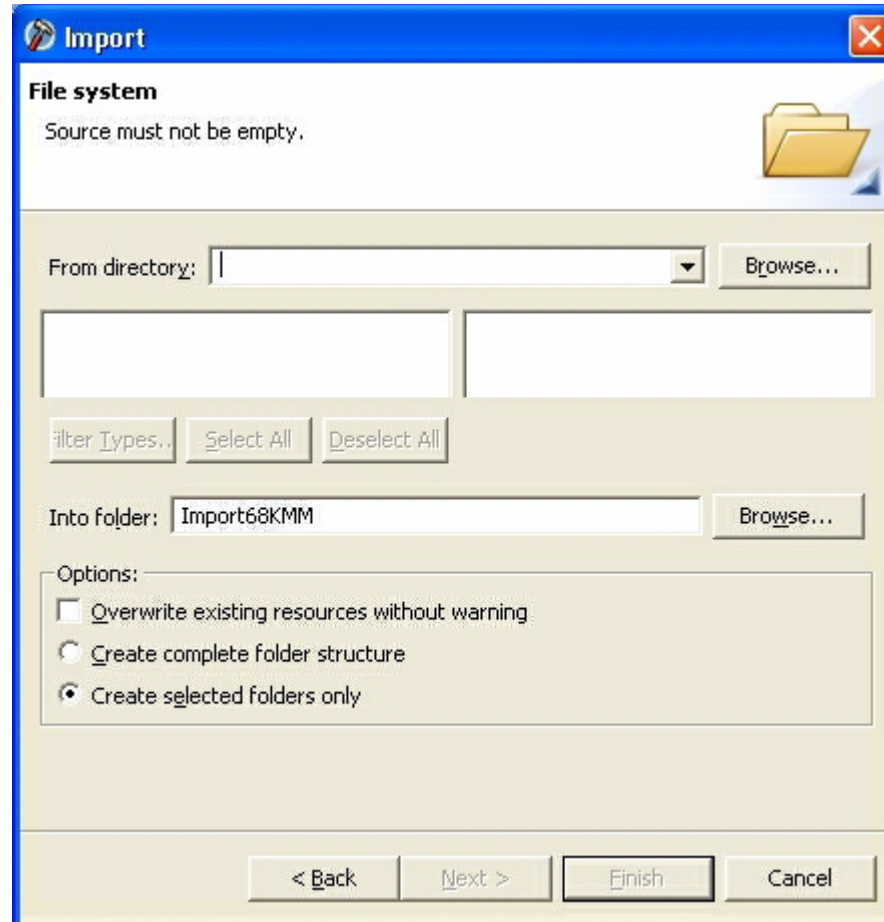


- Select File System and then click **Next** to open the File system Import dialog box, shown in [Figure 2.10](#) on page 38.

Eclipse Workbench Integration

Importing Files into a Project

Figure 2.10 File system Import



- Click Browse to select the folder where the existing application source files are located. Click the check boxes to select which files you want to import.
- Review the Import Options. For most projects, the default option **Create selected folders only** shown be correct.

Do not import any resource files or resource definition files; follow the instructions in section [“Importing Palm OS Resource Files”](#) on page 39 for these types of files.

Fixing Line Delimiters

Text files come with three varieties of end-of-line character sequences: Windows, Mac OS X/UNIX/Linux, and Classic Mac

OS). You may not be aware of which line delimiters your source files have, because most development environments automatically handle (and preserve) these values. However, the gcc tool chain on Windows requires that all source files end with Windows line delimiters (CRLF), and as a result can generate errors when compiling otherwise valid code.

Many sample code projects, CodeWarrior stationery projects, and book samples were created on Macintosh systems. As a result, some of the files in your old projects might have non-Windows line delimiters.

To convert sources to Windows line delimiters:

- Double-click the source file's icon in the C/C++ Projects view to open it in the editor view.
- Make sure the file's editor view is selected and choose **Convert Line Delimiters To -> (CRLF) Windows**. If you see that the file has been changed (there will be an asterisk next to the file's name in its editor tab), the Workbench has changed the delimiters to the proper value.
- Save the file.
- Repeat for each file in your project.

Importing Palm OS Resource Files

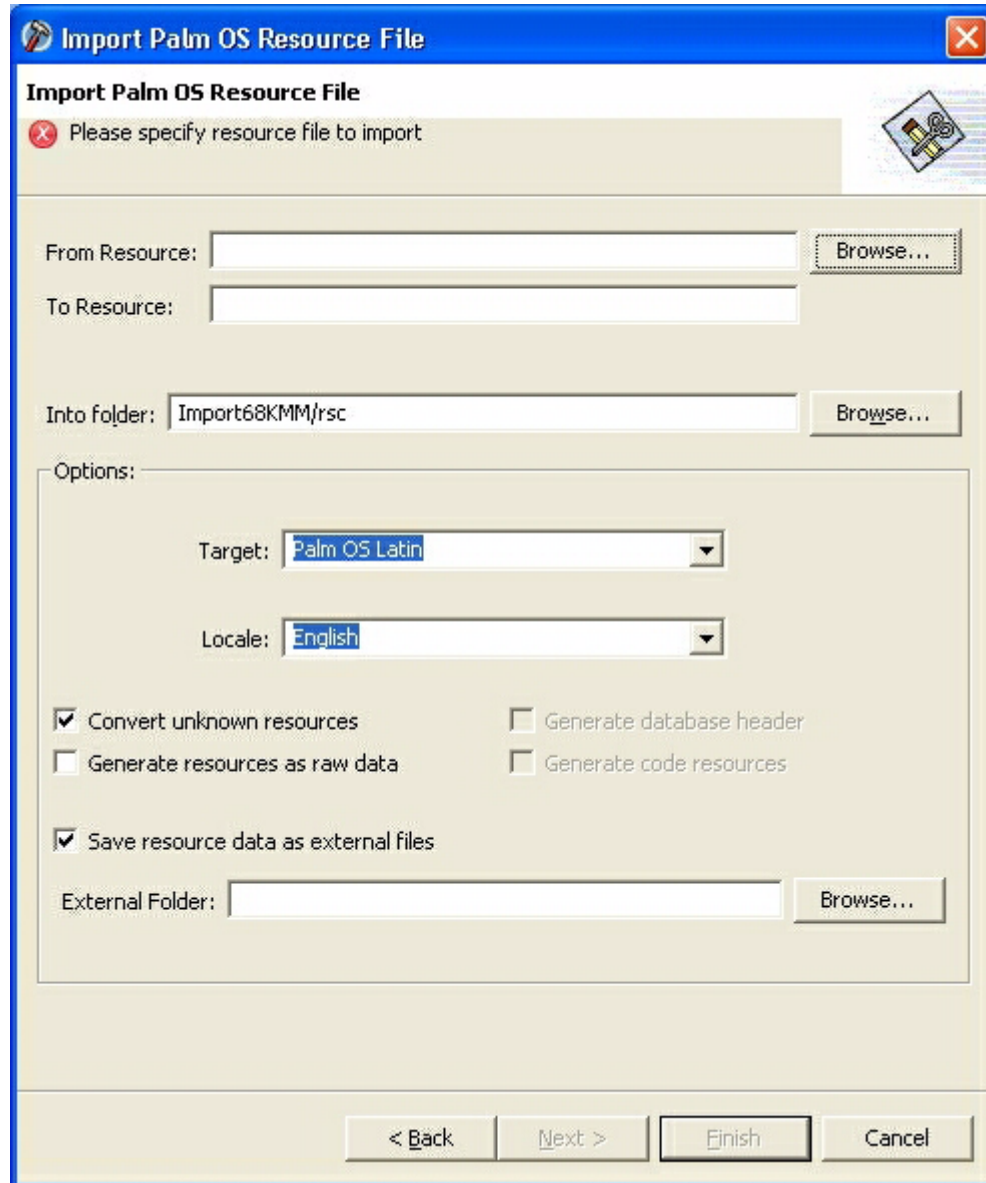
To import a Palm OS resource file, select File > Import to open the Import wizard, as shown in [Figure 2.9](#).

Select Palm OS Resource File and then click **Next**. The Import Palm OS Resource File wizard, shown in [Figure 2.11](#), opens.

Eclipse Workbench Integration

Importing Files into a Project

Figure 2.11 Import Palm OS Resource File wizard



- Select the PRC, RSRC, or XRD file that you want to import.
- Enter a name for the XRD file that you want added to your project. (If your input file is a PRC or a RSRC file, this file is the XRD file that will be generated from your input file.)
- Select the project folder into which you want this XRD file to be imported.

- Set additional options for the GenerateXRD tool, if you are importing from a PRC or RSRC file.

GenerateXRD options:

- Target
Specifies the text encoding of the input resources.
- Locale
Specifies the locale used to tag output resources with a `LOCALE` attribute.
- Convert unknown resource
Specifies that resource types that are not recognized are included in the output XRD file.
- Generate resources as raw data
Specifies that all resources that are output to the XRD are output as `RAW_RESOURCE` elements that preserve the binary input data exactly.
- Generate database header
This option is for importing PRC files. It specifies that the attributes of the input PRC database header will be stored as a `DATABASE_HEADER` element in the output XRD file.
- Generate code resources
This option is for testing and debugging. It specifies that executable code resources or support resources are output to the XRD file, usually as `RAW_RESOURCE` elements.
- Save resource data as external files
This option determines whether binary data should be included as inline binary data or as an external file reference.

Click **Finish** to import the resource file. Developer Suite calls the GenerateXRD tool to convert your PRC or RSRC file, if necessary, and updates your make file if you are using a managed make file.

For more information about GenerateXRD, see *Palm OS Resource Tools Guide*.

Importing Palm OS Sample Applications

Each SDK sample already includes a Developer Suite project. To build the samples from the SDK, first import the project into your workspace. Select **File > Import > Existing Project into Workspace**, and select the folder for the sample application. Then build the project within Eclipse.

NOTE: You can also use the command line to build the sample applications. In order to build the applications, invoke the make utility with the "`-f`" parameter by passing the name of the makefile. The makefile builds not only the target application but also all the dependent libraries. For example, to build the Address Book sample application, use the command `make -f makeaddr` at the command prompt.

When you import a project using **Import > Existing Project into Workspace**, Eclipse does not move your files but rather references the files in their existing location. If you want to retain an unchanged version of an existing project, you should make a copy of the project before importing it.

Editing Project Source Files

Developer Suite uses the Workbench's support of unique editors associated with different types of files.

- You edit C/C++ source files, header files, and shared library definition (SLD) files with the text editor. Simply double-click on a filename in the Navigator view, and the text editor opens the file in the editor view.
- You edit XML resource definition (XRD) files with Palm OS Resource Editor. Double-click on an XRD filename to open Palm OS Resource Editor.

Building a Project

The process for building a project is dependent on the type of project. See the appropriate section for your project:

- [“Building a Standard Make Project”](#)
- [“Building a Managed Make Project”](#) on page 44

Building a Standard Make Project

A standard make project uses the generated `makefile` to build the project. To change the settings for your standard make project, you edit this generated `makefile`.

For example, to specify whether you want a Debug build or a Release build of a standard make project, you edit this line in the `makefile`:

```
DEBUG_OR_RELEASE=Debug
```

To specify your run target for a standard make project, you edit this line in the `makefile`:

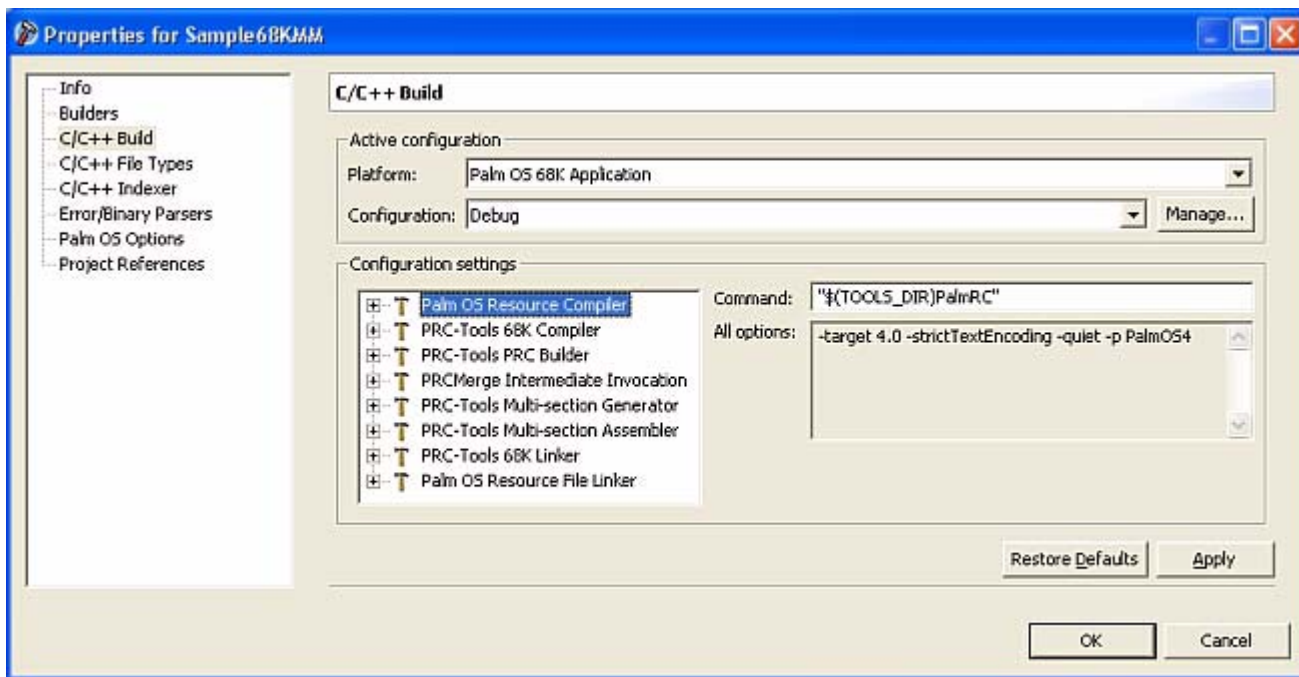
```
TARGET_PLATFORM=Device
```

After making any necessary changes to the project's `makefile`, you can build the standard make project by selecting **Project > Rebuild All**.

Building a Managed Make Project

Before you build your project, check the project's properties by selecting **Project > Properties**, which opens the Properties dialog box, as shown in [Figure 2.12](#).

Figure 2.12 Properties dialog box for managed make projects



From the Properties dialog box, you can view the current project properties, and change specific settings before building the project.

Active Configuration Settings

The Active Configuration setting is a combination of the Platform setting and the Configuration setting. Developer Suite supports the following active configuration settings.

- **Palm OS 68K Application with Debug configuration**

Builds the 68K application for execution on Palm OS Emulator, Palm OS Simulator, or a device, with the compiler's debug level set to the default debug level, `-g`.

This build uses the Cygwin compiler for 68K-based Palm OS devices (`m68k-palmos-gcc`) to build the application.

- **Palm OS 68K Application with Release configuration**

Builds the 68K application for execution on Palm OS Emulator, Palm OS Simulator, or a device, with the compiler's debug level set to None.

This build uses the Cygwin compiler for 68K-based Palm OS devices (`m68k-palmos-gcc`) to build the application.

- **Palm OS 68K PNO Application (Device target) with DebugDevice configuration**

Builds the 68K application with a PNO for execution on a Palm Powered device with additional debugging settings specified.

This build uses the Cygwin compiler for 68K-based Palm OS devices (`m68k-palmos-gcc`) to build the 68K application, and uses Palm OS Protein C/C++ Compiler to build the PNO component.

- **Palm OS 68K PNO Application (Device target) with ReleaseDevice configuration**

Builds the 68K application with a PNO for execution on a Palm Powered device.

This build uses the Cygwin compiler for 68K-based Palm OS devices (`m68k-palmos-gcc`) to build the 68K application, and uses Palm OS Protein C/C++ Compiler to build the PNO component.

- **Palm OS 68K PNO Application (Simulator target) with DebugSim configuration**

Builds the 68K application with a PNO for execution on Palm OS Simulator with additional debugging settings specified.

This build uses the Cygwin compiler for 68K-based Palm OS devices (`m68k-palmos-gcc`) to build the 68K application, and uses the Cygwin C/C++ compiler for Windows (`i686-pc-cygwin-gcc-3.3.1`) to build the PNO component.

- **Palm OS 68K PNO Application (Simulator target) with ReleaseSim configuration**

Builds the 68K application with a PNO for execution on Palm OS Simulator.

This build uses the Cygwin compiler for 68K-based Palm OS devices (`m68k-palmos-gcc`) to build the 68K application,

Eclipse Workbench Integration

Building a Project

and uses the Cygwin C/C++ compiler for Windows (i686-pc-cygwin-gcc-3.3.1) to build the PNO component.

- **Palm OS Protein Application (Device target) with DebugDevice configuration**

Builds the Palm OS Protein application for execution on a Palm Powered device with additional debugging settings specified.

This build uses Palm OS Protein C/C++ Compiler to build the application to run on an ARM-based device.

- **Palm OS Protein Application (Device target) with ReleaseDevice configuration**

Builds the Palm OS Protein application for execution on a Palm Powered device.

This build uses Palm OS Protein C/C++ Compiler to build the application to run on an ARM-based device.

- **Palm OS Protein Application (Simulator target) with DebugSim configuration**

Builds the Palm OS Protein application for execution on Palm OS Simulator with additional debugging settings specified.

This build uses the Cygwin C/C++ compiler for Windows (i686-pc-cygwin-gcc-3.3.1) to build the application.

- **Palm OS Protein Application (Simulator target) with ReleaseSim configuration**

Builds the Palm OS Protein application for execution on Palm OS Simulator.

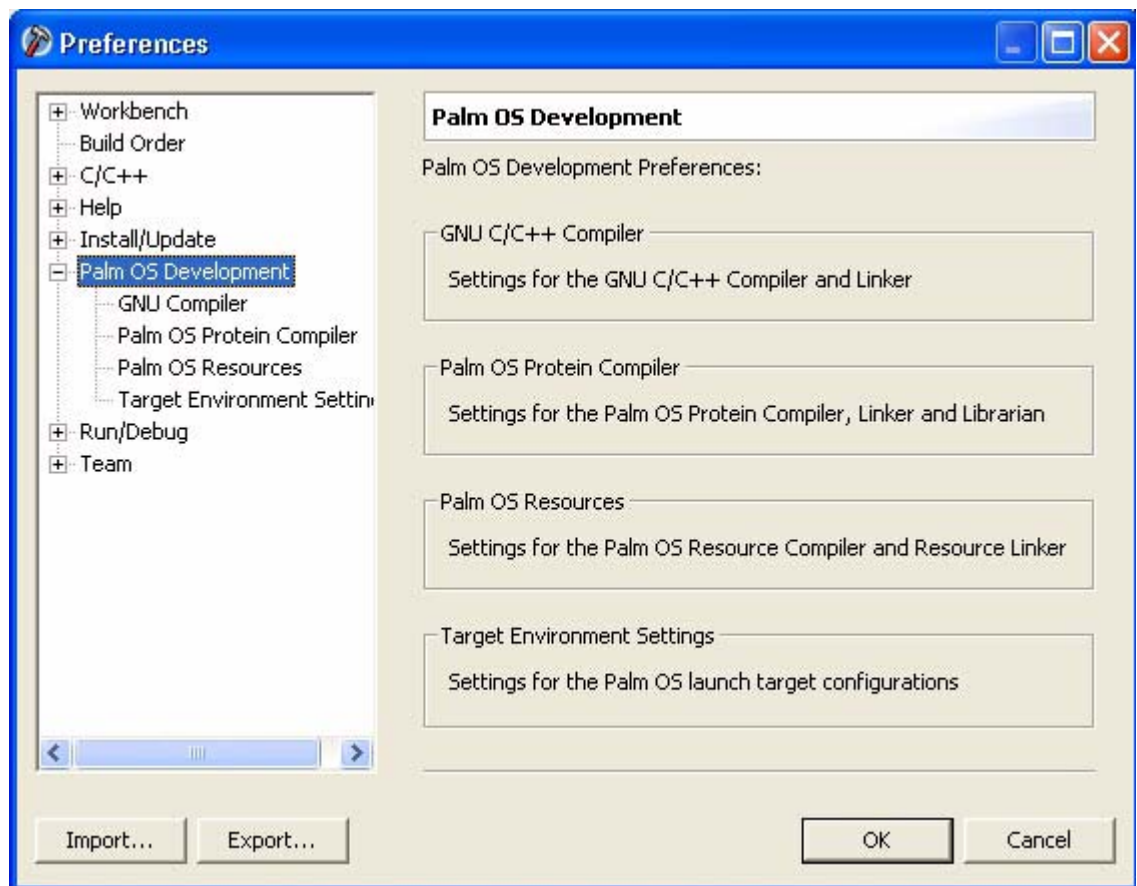
This build uses the Cygwin C/C++ compiler for Windows (i686-pc-cygwin-gcc-3.3.1) to build the application.

Once you have set the build configuration settings, to build a project, select the project in the C/C++ Projects pane. Then select **Project > Build Project**.

Modifying Tool Properties

To modify the properties for the compilers, resource tools, and target environment settings, you use the Palm OS Development Preferences dialog box, shown in [Figure 2.13](#).

Figure 2.13 Preferences dialog box



To open the Preferences dialog box, select **Window > Preferences**. Then select **Palm OS Development**.

GNU Compiler Preferences

Use this pane to set preferences for the GNU compilers.

Eclipse Workbench Integration

Modifying Tool Properties

- Select the **68K** tab to change preferences for the Cygwin compiler for 68K-based Palm OS devices (`m68k-palms-gcc`). This is the compiler used to build 68K applications.
- Select the **x86** tab to change the preferences for the Cygwin C/C++ compiler for Windows (`i686-pc-cygwin-gcc-3.3.1`). This is the compiler used to build applications targeted for Palm OS Simulator.

For information about GNU compiler options, see the documentation at the PRC-Tools web site (<http://prc-tools.sourceforge.net/doc/>).

Palm OS Protein Compiler Preference

Use this pane to set preferences for the Palm OS Protein C/C++ Compiler's associated compiler tools: `pacc`, `palink`, and `palib`.

- Select the **Palm OS C/C++ Compiler** tab to set options for `pacc`.
- Select the Palm OS Linker tab to set options for `palink`.
- Select the Palm OS Librarian tab to set options for `palib`.

For more information about the compiler tools, see the book *Palm OS Protein C/C++ Compiler Tools Guide*.

Palm OS Resources Preferences

Use this pane to set preferences for the Palm OS resource tools.

- Select the **Resource Compiler** tab to set options for `PalmRC`.
- Select the **PRCMerge** tab to set options for `PRCMerge`.

For more information about the resource tools, see the book *Palm OS Resource Tools Guide*.

Target Environment Settings Preferences

Use this pane to set preference for the execution targets used for running and debugging applications.

- Select **Device** to set preferences for using Palm Powered devices as run targets.

- Select **Emulator** to set preferences for using Palm OS Emulator as a run target. Palm OS Developer Suite comes with one Emulator target pre-defined. You can change this target's settings, or create additional Emulator targets.
 - Use the Browse button to select a different Emulator executable.
 - Use the Options entry field to specify command line arguments for Emulator.
 - To add a new Emulator target, use the New button.

For more information about Palm OS Emulator settings, see the book *Using Palm OS Emulator*.

- Select **Simulator** to set preferences for using Palm OS Garnet Simulator and Palm OS Cobalt Simulator as run targets. Palm OS Developer Suite comes with one Palm OS Garnet Simulator target pre-defined, and with one Palm OS Cobalt Simulator target pre-defined. You can change these targets' settings, or create additional Simulator targets.
 - Use the Browse button to select a different Simulator executable.
 - Use the Options entry field to specify command line arguments for Simulator.

For example, Palm OS Cobalt Simulator skips the Language Picker application at startup if you specify the `-preferredLocale` setting as a command line argument.
 - To add a new Simulator target, use the New button.

For information about Palm OS Garnet Simulator, see the book *Testing with Palm OS Garnet Simulator*. For information about Palm OS Cobalt Simulator, see the book *Palm OS Cobalt Simulator Guide*.

Launching Other Palm OS Tools

You can launch the following standalone tools from Developer Suite:

- [Palm OS Garnet Simulator](#)
- [Palm OS Cobalt Simulator](#)
- [Palm OS Virtual Phone](#)
- [Palm OS Debugger](#)
- [Palm OS Emulator](#)
- [Palm OS Reporter](#)
- [Palm OS Resource Editor](#)

Palm OS Garnet Simulator

In general, you launch Palm OS Garnet Simulator as a run target, using the **Run** menu. For information on using Palm OS Garnet Simulator as a run target, see “[Using Palm OS Simulator](#)” on page 78.

In addition, you can launch Palm OS Garnet Simulator directly by selecting **Start > Programs > PalmSource > Palm OS Garnet SDK Tools > Palm OS Garnet Simulator**.

For more information about Palm OS Garnet Simulator, see *Testing with Palm OS Garnet Simulator*.

Palm OS Cobalt Simulator

In general, you launch Palm OS Cobalt Simulator as a run target, using the **Run** menu. For information on using Palm OS Cobalt Simulator as a run target, see “[Using Palm OS Simulator](#)” on page 78.

In addition, you can launch Palm OS Cobalt Simulator editor directly by selecting **Start > Programs > PalmSource > Palm OS Cobalt SDK Tools > Palm OS Cobalt Simulator**.

For more information about Palm OS Cobalt Simulator, see *Palm OS Cobalt Simulator Guide*.

Palm OS Virtual Phone

There are two versions of Virtual Phone included in the Developer Suite package: one included from the Palm OS Garnet SDK and one include from the Palm OS Cobalt SDK. You can launch Palm OS Virtual Phone directly by selecting either:

- **Start > Programs > PalmSource > Palm OS Cobalt SDK Tools > Palm OS Virtual Phone**
- **Start > Programs > PalmSource > Palm OS Garnet SDK Tools > Palm OS Virtual Phone**

For additional information on Virtual Phone, see *Palm OS Virtual Phone Guide*.

Palm OS Debugger

Palm OS Debugger is integrated into the Palm OS C/C++ Development perspective. To launch Palm OS Debugger from Developer Suite, select **Run > Palm OS Debugger**.

In addition, you can launch Palm OS Debugger directly by selecting **Start > Programs > PalmSource > Tools > Palm OS Debugger**.

For additional information on Palm OS Debugger, see *Palm OS Debugger Guide*.

Palm OS Emulator

In general, you launch Palm OS Emulator as a run target, using the **Run** menu. For information on using Palm OS Emulator as a run target, see "[Using Palm OS Emulator](#)" on page 84.

In addition, you can launch Palm OS Emulator directly by selecting **Start > Programs > PalmSource > Tools > Palm OS Emulator**.

For additional information about Palm OS Emulator, see *Using Palm OS Emulator*.

Palm OS Reporter

To autostart Reporter with Palm OS Simulator, edit the `PalmSim.ini` file and set the value of the `AutoStartReporter` option:

```
AutoStartReporter=1
```

In addition, you can launch Palm OS Reporter directly by selecting **Start > Programs > PalmSource > Tools > Palm OS Reporter**.

Palm OS Resource Editor

To launch Palm OS Resource Editor from Developer Suite, double-clicking an XRD source file in your project. Palm OS Resource Editor starts and loads the selected XRD file.

In addition, you can launch Palm OS Resource editor directly by selecting **Start > Programs > PalmSource > Tools > Palm OS Resource Editor**.

Updating Palm OS Developer Suite

The Eclipse Workbench offers a simple mechanism for discovering, downloading, and installing upgrades to Palm OS Developer Suite. You can obtain new components and install them without exiting your development environment.

The Update Manager provides this functionality by means of a wizard that connects to update sites, displays a list of available updates, and then automatically downloads and installs the ones you select.

Features

Eclipse-based products such as Palm OS Developer Suite have a modular architecture, made up of many functional components called *plug-ins*. Related plug-ins are organized into features. A **feature** is the smallest package of code that can be downloaded and installed as a unit. Using features, you can upgrade Palm OS Developer Suite without the cost in time and bandwidth of downloading an entirely new version.

Palm OS Developer Suite consists of the following features:

- Palm OS Developer Suite
- Cygwin/PRC-Tools for Palm OS Developer Suite
- Palm OS Developer Suite—Japanese Simulator ROMs
- Palm OS Developer Suite—Chinese Simulator ROMs.

Some of these features contain sub-features. For instance, Palm OS Developer Suite has sub-features as indicated below:

- Palm OS Developer Suite
 - Cygwin/PRC-Tools Modified Components
 - Eclipse C/C++ Development Tools (patched)

The entire feature, however, must be downloaded, installed, uninstalled, or reverted as a unit.

New and Updated Features

The Update Manager wizard lists only features that are new or have later version numbers than those already installed on your computer. Plug-ins with version numbers that have not changed do not appear on the list. If you accidentally damage some components of Palm OS Developer Suite, you should download a fresh distribution of the version of Palm OS Developer Suite you are using and re-install it.

Prerequisites

Make sure you have Palm OS Developer Suite 1.2 installed before trying to use the Update Manager. Currently, the updating functionality does not work with earlier versions of Palm OS Developer Suite or with a generic version of Eclipse.

Verifying Features

You can check what features you have installed and identifying information about them at any time. Select **Help > About Palm OS Developer Suite**, and then click the **Feature Details** button. A table is displayed, each row listing installed features and their subfeatures: columns display the name of the feature, the

organization providing it, the feature's current version number, and the feature ID.

Obtaining New Features

To obtain a new feature, select **Help > Software Updates > Find and Install**. The Find and Install wizard opens.

- In the Feature Updates panel, select **Search for new features to install**.
- In the Update sites to visit panel of the wizard, select **Palm OS Developer Suite Update Site** by clicking on the white box next to the name.
 - If you click on the plus-sign, you will see a list of categories of content available on the update site. On a larger site, you can select individual categories of content. This choice determines which feature names appear on the next panel.
- In the Password Required dialog, enter the user name and password you registered at the Palm OS Developer Pavilion. If you don't have an account, go to the following web site and create one:
<https://www.developerpavilion.com>
- In the Search Results panel, select the features you wish to download by clicking on the white box next to each one. Then click **Finish**.

Obtaining Updates to Existing Features

You can obtain updates to features that are already installed on your computer. Select **Help > Software Updates > Find and Install**. The Find and Install wizard opens. Then proceed as described in "[Obtaining New Features](#)" on page 54.

- In the Feature Updates panel, select **Search for updates of the currently installed features**.
- In the Update sites to visit panel of the wizard, select **Palm OS Developer Suite Update Site** by clicking on the white box next to the name.

- If you click on the plus-sign, you will see a list of categories of content available on the update site. On a larger site, you can select individual categories of content. This choice determines which feature names appear on the next panel.
- In the Password Required dialog, enter the user name and password you registered at the Palm OS Developer Pavilion. If you don't have an account, go to the following web site and create one:
<https://www.developerpavilion.com>
- In the Search Results panel, select the Palm OS Developer Suite feature. Then click **Finish**.

Managing Features

You can use the Managing Configuration panel to manage your features—either individually or globally.

Managing Individual Features

You can disable individual features and then enable them again. You can revert to a previous version of the feature, or completely uninstall a feature. To do these tasks,

1. Select **Help > Software Updates > Manage Configuration**,
2. Open the tree to the appropriate level, and click on the feature.
3. In the right-hand panel, click one of the following:

Replace with Another Version: The Replace with Another Version dialog box appears. Select the version you desire to make active.

Enable or disable, depending on the current state of this version of the feature: This link toggles between the two states of the version.

Uninstall: Removes this version of the feature from your computer.

Show Properties: Displays properties of this version of the feature.

Managing Features as a Configuration

You can also operate globally, returning all installed features to their state at a particular moment in the past. To do so, you choose from a list of past configurations. A **configuration** is the state of all features at the moment when a particular activity was completed. Activities can include downloading/installing updates or managing installed features by disabling, re-enabling, or reverting them. A configuration is identified by the date and time of the last activity or group of activities before that configuration was created.

To revert to a configuration:

1. Select **Help > Software Updates > Manage Configuration**.
2. In the Product Configuration dialog box, click on the root of the tree—**Palm OS Developer Suite**.
3. In the right-hand panel, click on **Revert to previous**.
4. In the Revert to a previous configuration dialog box, click on a date-time in the Past Configurations list.
5. Examine the table titled Activities that caused the creation of this configuration.

This is the complete list of activities that resulted in the selected configuration. Note that one activity, such as disabling a particular version of a feature, may be negated by a succeeding activity, such as enabling that same version.

6. Click the Finish button after you verify that this is the configuration to which you wish to revert.

Consequences of Reverting a Feature

Reverting from a version of a feature causes that version to be disabled; it is not uninstalled. You will not be able to access this version again through the update mechanism. The version will not show up in the Find and Install dialog box, since it is already installed on your computer.

To access a version that was disabled via reverting:

1. Select **Help > Software Updates > Manage Configuration**.
2. In the tree, select the feature in question.
3. In the right-hand panel, click **Revert to Another Version**.

4. In the Replace With Another Version dialog box, select the version you want to make active.
5. Click the Finish button.

Getting More Information

The Workbench provides much integrated documentation in the Help system. To view all of the integrated documentation, select **Help > Help Contents**.

The Workbench component itself is described in detail in the *Workbench User Guide*.

Eclipse Workbench Integration

Getting More Information

Palm OS Compiler Tools

This chapter describes how to use the Palm OS compiler tools:

- [“Using Compiler Tools with the Workbench”](#) on page 59
- [“Using the Palm OS Protein C/C++ Compiler Tools Independently”](#) on page 61

Using Compiler Tools with the Workbench

The Palm OS compiler tools are fully integrated with the Workbench (described in [Chapter 2, “Eclipse Workbench Integration,”](#) on page 17).

The Palm OS C/C++ Perspective uses different compiler chains to build projects, dependent on the application types and target types.

Table 3.1 Compiler chains use by Palm OS Developer Suite

Compiler	When Used
PRC-Tools GNU 68K compiler, m68K-palmos-gcc.exe	Palm OS 68K applications, shared libraries, and static libraries targeted for Device and Simulator
GNU x86 compiler, i686-pc-cygwin-gcc-3.3.1.exe	Palm OS Protein applications, shared libraries, and static libraries targeted for Palm OS Cobalt Simulator; PNOs targeted for either Palm OS Garnet Simulator or Palm OS Cobalt Simulator.
Palm OS Protein C/C++ Compiler, pacc.exe	Palm OS Protein applications, shared libraries, and static libraries targeted for Device; PNOs targeted for Device.

Palm OS Compiler Tools

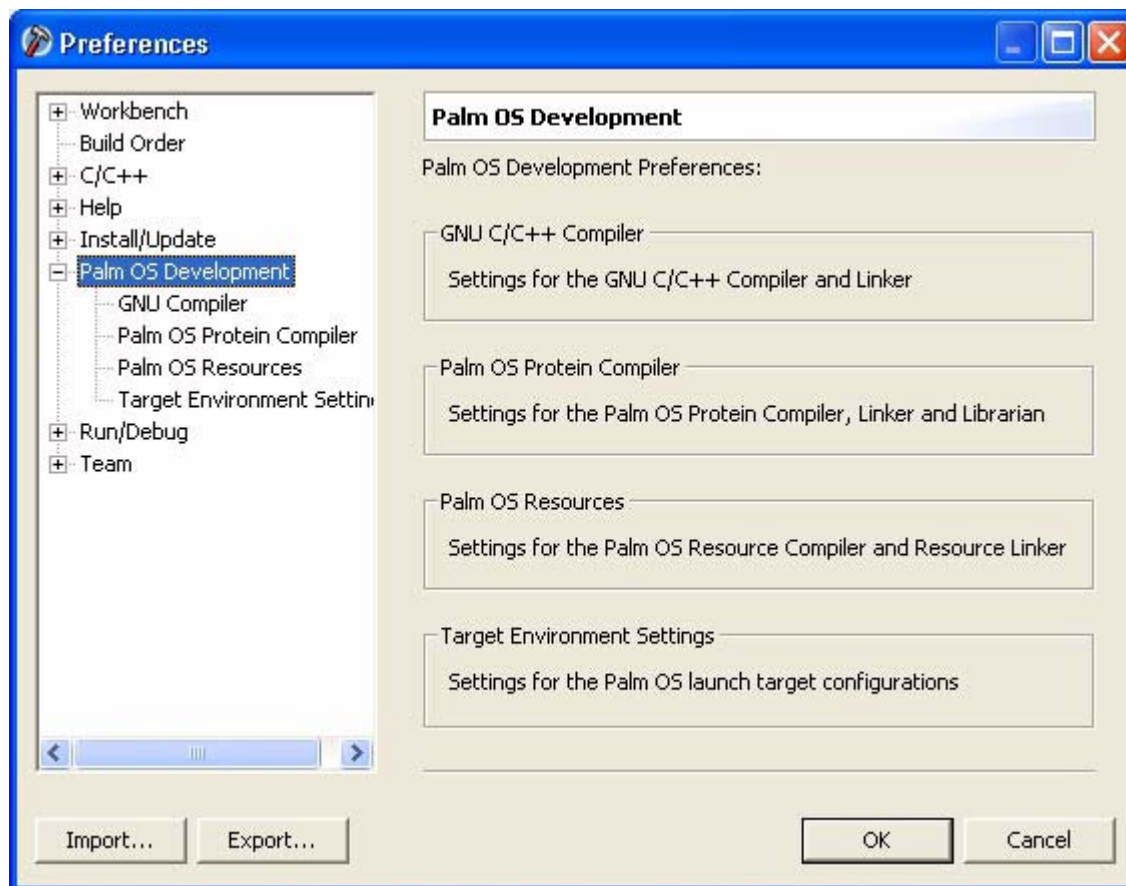
Using Compiler Tools with the Workbench

When you use the development environment's menu to build a project, the correct compiler chain is called to compile your application.

Setting Compiler Options

To set compiler options in the development environment, you use the Eclipse Workbench's Preferences dialog box. Select **Window > Preferences** to open the Preferences dialog box, shown in [Figure 3.1](#).

Figure 3.1 Palm OS Development Preferences



GNU Compiler Options

To set compiler options for the GNU C/C++ compilers (gcc), select **GNU Compiler**. You can set options for both the 68K and the x86 compilers on this page.

For information on the GNU compiler options, see the documentation at PRC-Tools web site (<http://prc-tools.sourceforge.net/doc/>).

Palm OS Protein C/C++ Compiler Options

To set compiler options for Palm OS Protein C/C++ compiler (`pacc`), select **Palm OS Protein Compiler**. You can also set options for Palm OS Linker (`palink`) and Palm OS Librarian (`palib`) tools on this page.

For information on the Palm OS Protein C/C++ compiler options, see the book *Palm OS Protein C/C++ Compiler Tools Guide*.

Running the Compiler Tools

To run the compiler in the development environment, select one of the menu items from the **Build** menu. For example, select **Build > Build Solution** to use all of the Palm OS tools to build your Palm OS application.

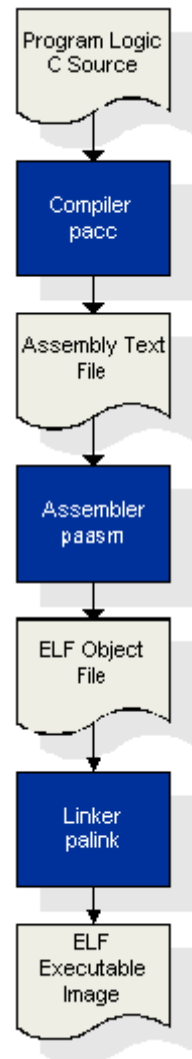
Using the Palm OS Protein C/C++ Compiler Tools Independently

As is common with command line compilers, Palm OS Protein C/C++ Compiler, `pacc`, acts as a driver. `pacc` invokes all of the commands necessary to produce linked files from source code.

Palm OS Compiler Tools

Using the Palm OS Protein C/C++ Compiler Tools Independently

Figure 3.2 Compiler Overview



pacc, paasm, palink: Compiler Tools

- pacc compiles the source files into assembly
- pacc calls the assembler, paasm, to produce object code
- pacc calls the linker, palink, to generate the ELF executable image.

palib: Librarian

`palib` lets you create and manage a collection of ELF object files. With `palib`, you can:

- Create a new library.
- Add files to the library.
- Delete files from the library.
- Replace files in a library.
- Extract files from a library.

elfdump: Diagnostic Tool

`elfdump` lets you place the contents of an ELF object file into a text file. With `elfdump`, you can:

- Disassemble executable bytecode sections
- Disassemble data sections as code.
- Disassemble for a given instruction set architecture
- Show only segment and section summaries
- Show specific sections, such as code, data, debug information or symbols.

Getting More Information

The Palm OS compiler tools are described in detail in the book *Palm OS Protein C/C++ Compiler Tools Guide*.

Palm OS Compiler Tools

Getting More Information

Palm OS Resource Tools

This chapter describes how to use the Palm OS resource tools.

- “[Resource File Overview](#)” on page 66
- “[Using the Resource Tools with the Workbench](#)” on page 67
- “[Using the Resource Tools Independently](#)” on page 72

Resource File Overview

The source code for Palm OS resources are stored in a platform-independent, XML text file format. This file format is called an XML resource description file, or **XRD file**.

This XRD file is an XML implementation for defining application resources. For more information on the format of XRD files and XML tag descriptions, see the book *Palm OS Resource File Formats*.

Importing a Resource File

If you have an exiting PRC or RSRC file that you want to add to a project in Palm OS Developer Suite as an XRD file, use the Palm OS Resource File import wizard.

- Select **File > Import** to open the Import dialog box.
- Select **Palm OS Resource File**. Follow the wizard steps to select the PRC or RSRC file, to select the project into which you want to import the resource file, and to set other options.

The import wizard calls the `GenerateXRD` tool to create an XRD file from your resource file.

Creating an XRD File

To add a new XRD file to a project:

- First select the project.
- Then select **File > New > XRD File** to create an empty XRD file in the selected project.

You can then edit the XRD file using Palm OS Resource Editor. For more information on Palm OS Resource Editor, see the book *Palm OS Resource Editor Guide*.

Using the Resource Tools with the Workbench

The Palm OS resource tools for building applications are fully integrated with the Workbench (described in [Chapter 2, “Eclipse Workbench Integration,”](#) on page 17).

When you use the development environment’s menu to build your solution configuration, the Palm OS resource tools are called to compile your XML resource definition file and merge the output with your code and data resources to produce a Palm OS application.

NOTE: The Palm OS resource tools `PalmRC` and `PRCMerge` are used to build your application both for Device configurations and for Simulator configurations.

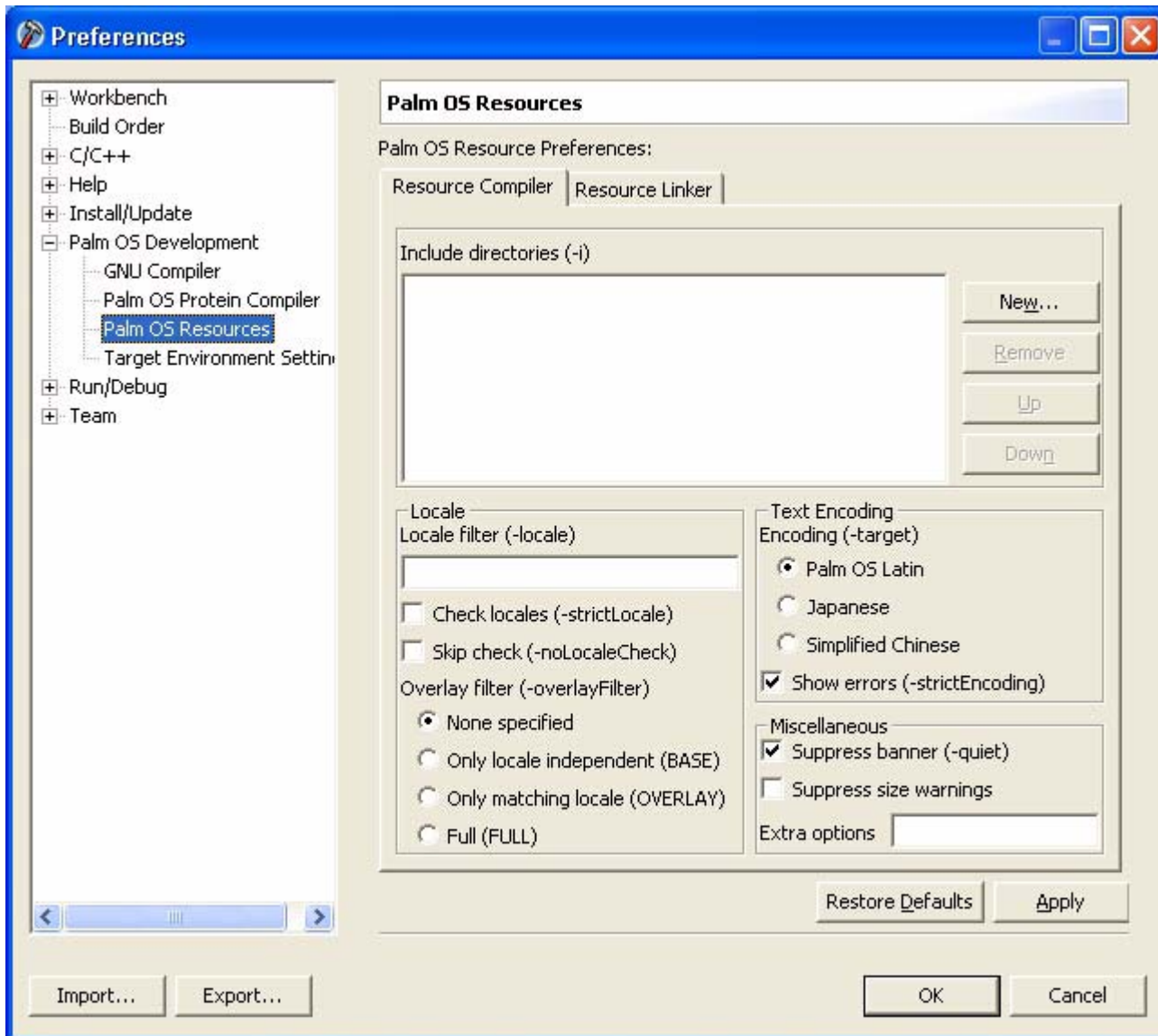
Setting Resource Compiler (PalmRC) Options

- To set the default resource compiler options in the development environment, you use the Eclipse Workbench’s Preferences dialog box. Select **Window > Preferences** to open the Preferences dialog box, shown in [Figure 3.1](#) on page 60. Then select **Palm OS Resources** to open the page shown in [Figure 4.1](#) on page 67.

Figure 4.1 Palm OS Resource Preferences

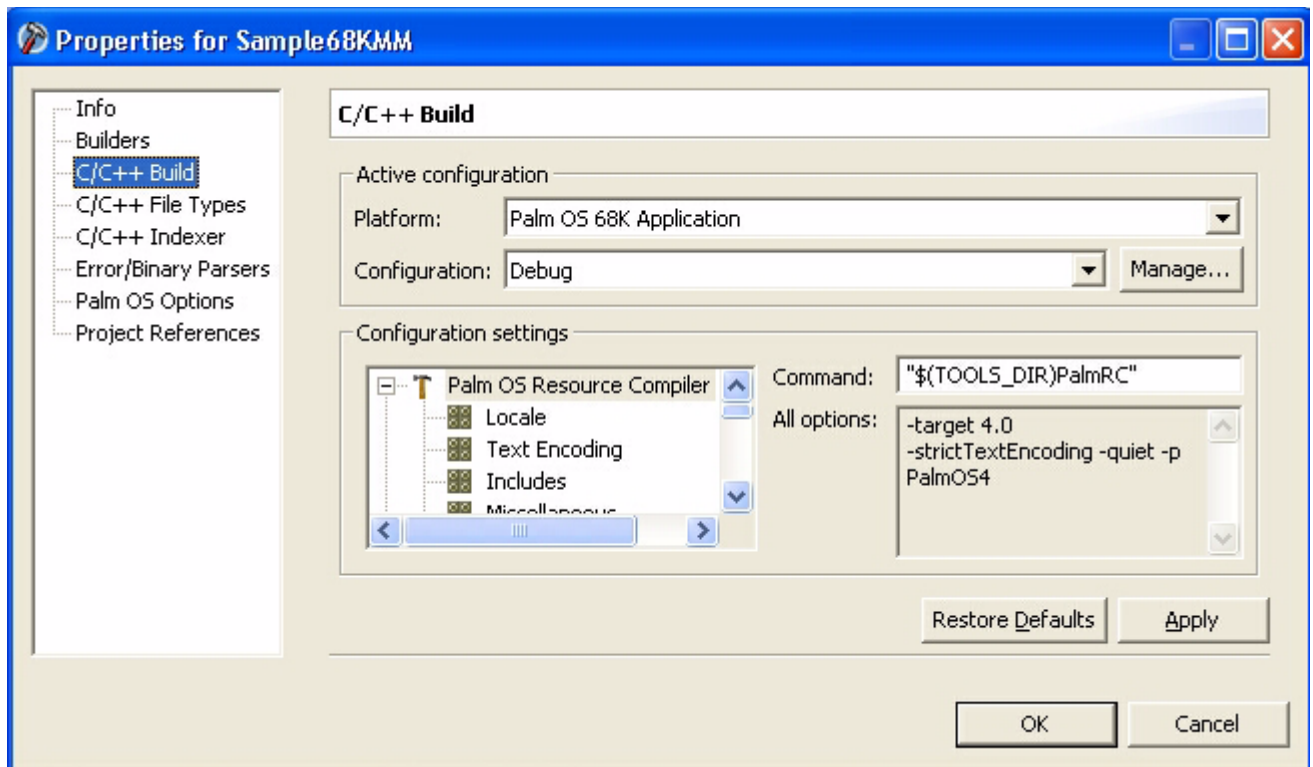
Palm OS Resource Tools

Using the Resource Tools with the Workbench



- To set the specific resource compiler options for a single project, you use the project's properties dialog box. Select the project, then select **Project > Properties** to open the Properties dialog box, shown in [Figure 4.2](#). Select the **C/C++ Build** group in the left pane, and then select **Palm OS Resource Compiler** in the Configuration settings list.

Figure 4.2 PalmRC Properties dialog box



For more information about the PalmRC and the Resource Compiler options, see the book *Palm OS Resource Tools Guide*.

Setting Resource Linker (PRCMerge) Options

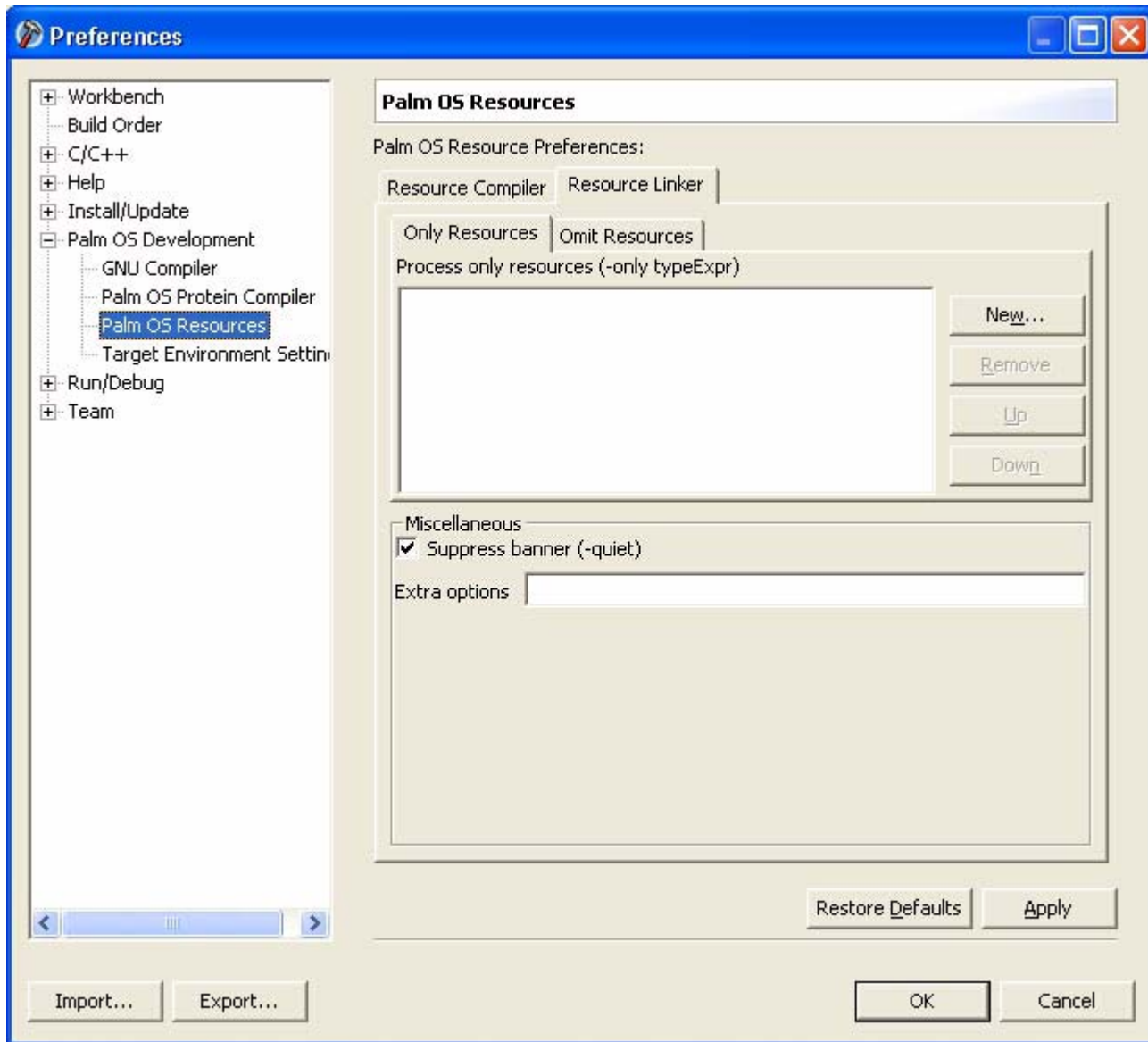
- To set the default PRCMerge options in the development environment, you use the Eclipse Workbench's Preferences dialog box. Select **Window > Preferences** to open the Preferences dialog box, shown in [Figure 3.1](#) on page 60. Then select **Palm OS Resources** to open the page shown in [Figure](#)

Palm OS Resource Tools

Using the Resource Tools with the Workbench

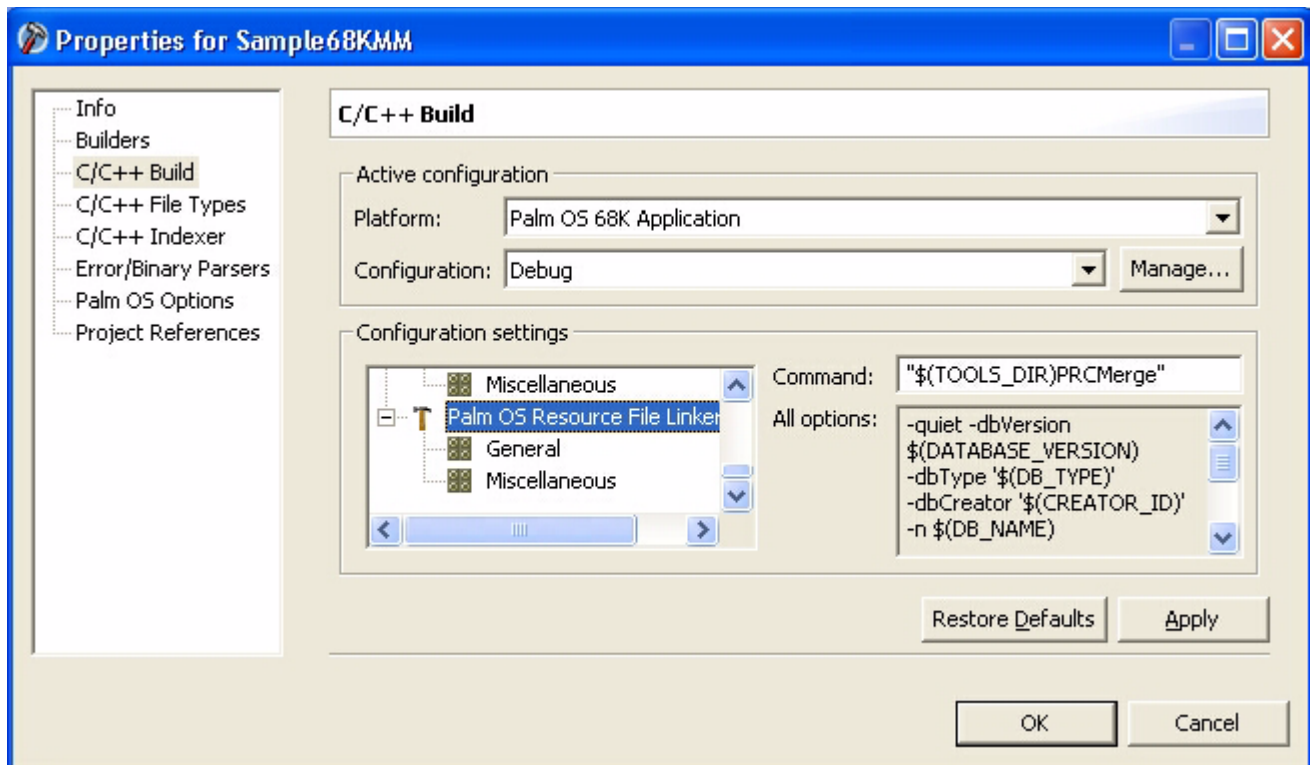
[4.1](#) on page 67. Then select **Resource Linker** to display the page shown in [Figure 4.3](#) on page 70.

Figure 4.3 PRCMerge options



- To set the specific resource linker options for a single project, you use the project's properties dialog box. Select the project, then select **Project > Properties** to open the Properties dialog box, shown in [Figure 4.4](#). Select the **C/C++ Build** group in the left pane, and then select **Palm OS Resource File Linker** in the Configuration settings list.

Figure 4.4 PRCMerge Properties dialog box



For more information about PRCMerge and PRCMerge options, see the book *Palm OS Resource Tools Guide*.

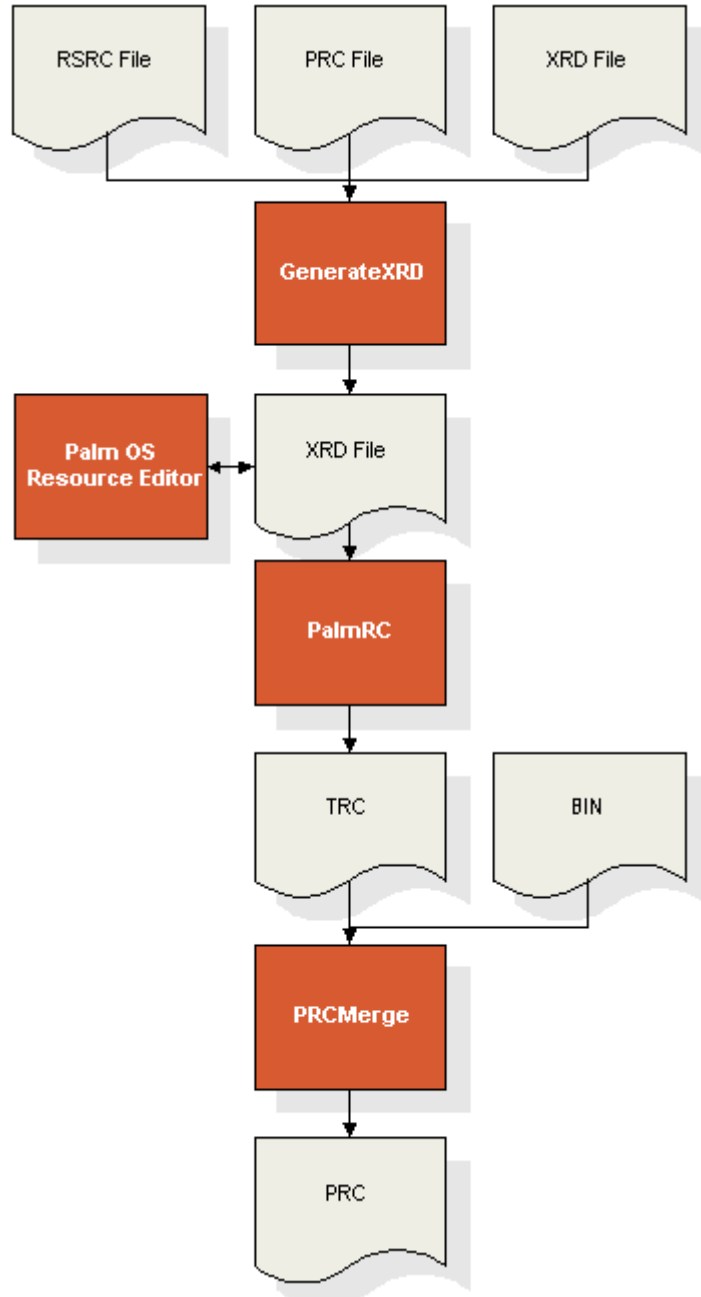
Using the Resource Tools Independently

Palm OS resource tools fall into the following categories:

- Migration tools
- Editing tools
- Building tools
- Utility tools
- Localization tools
- Security tools

PalmSource's toolset provides resource tools in each of these categories, as described in the following descriptions.

Figure 4.5 Basic Resource Tools



GenerateXRD: Resource Migration Tool

GenerateXRD provides a way to conversion from existing resource files to the new resource file format. If you currently have resources in Macintosh RSRC files, you can convert them to the new XRD format without loss of information. You can also “de-compile” resources from PRC files into the new XRD resource file format.

Palm OS Resource Editor: Resource Editing Tool

Palm OS Resource Editor allows you to create and edit the new XML resource definition (XRD) files. With Palm OS Resource Editor, you can edit XML tags directly, or you can use a graphical form editor, dragging and dropping user interface controls from a catalog of user interface elements.

PalmRC and PRCMerge: Resource Building Tools

Build tools are resource file compilers and linkers that build resources into Palm OS applications (PRC files). The resource file compiler is called PalmRC; the linker is called PRCMerge.

PRCCompare: Resource Utility Tool

PRCCompare is a utility tool that shows you the difference between two Palm OS binary resource database files.

hOverlay: Application Localization Tool

Localization tools allow you to create locale-specific versions of your Palm OS application. hOverlay is a tool that helps you create overlay PRCs with national language strings that work with your original base PRC.

PRCSign and PRCCert: Application Security Tools

Security tools provide a way for securing and authenticating a Palm OS application. PRCSign and PRCCert allow you to create and

embed a digital signature and associated certificates in your Palm OS application (PRC file).

Getting More Information

The Palm OS Resource Tools are described in detail in the book *Palm OS Resource Tools Guide*.

Palm OS Resource Tools

Getting More Information

Palm OS Testing Tools

This chapter describes how to use the Palm OS testing tools.

The Palm OS Developer Suite package provides the following testing tools:

- “[Using Palm OS Simulator](#)” on page 78 describes how to use the device simulator, Palm OS Simulator.
- “[Using Palm OS Reporter](#)” on page 82 introduces how to use Palm OS Reporter with Palm OS Simulator or Palm OS Emulator.
- “[Using Palm OS Emulator](#)” on page 84 describes how to use the 68K device emulator, Palm OS Emulator.
- “[Using Virtual Phone](#)” on page 86 describes how to use the mobile telephone simulator, Virtual Phone.

Using Palm OS Simulator

Palm OS Simulator is Palm OS recompiled for a desktop machine processor. Palm OS Simulator combines the following into a single execution environment:

- Palm OS applications
- Palm Application Compatibility Environment (PACE)
- Palm OS system code

Palm OS Simulator includes all of the built-in Palm OS applications, such as Address Book, Date Book, Memo Pad, and To Do List. The built-in Palm OS applications are included in the Simulator ROM file. You test your own applications by adding them to a Simulator session.

Palm OS Garnet Simulator

Palm OS Garnet Simulator is Palm OS Garnet recompiled to run on Windows. The Developer Suite package includes the most recent build of Palm OS Garnet Simulator, which corresponds to Palm OS Garnet release 5.4.

For more information on Palm OS Garnet Simulator, see the book *Testing with Palm OS Garnet Simulator*.

Palm OS Cobalt Simulator

Palm OS Cobalt Simulator is Palm OS Cobalt recompiled to run on Windows. The Developer Suite package includes the most recent build of Palm OS Cobalt Simulator, which corresponds to Palm OS Cobalt release 6.0.1.

For more information on Palm OS Cobalt Simulator, see the book *Palm OS Cobalt Simulator Guide*.

Creating a Palm OS Simulator Run Target

Developer Suite comes with two default Palm OS Simulator run targets already created:

- The debug version of Palm OS Cobalt Simulator version 6.0.1
- The debug version of Palm OS Garnet Simulator version 5.4

To create additional Simulator run targets, follow these steps:

- From the Developer Suite interface, select **Window > Preferences**. This opens the Preferences dialog box.
- Open the category **Palm OS Development**, and select **Target Environment Settings**.
- Select the **Simulator** item in the list, and click **New** to show the Simulator preferences.
- Enter a name for your Simulator Run Target.
- Click **Browse** to select a Simulator executable that you want to use for this Simulator Run Target.
- Enter any additional command line options in the **Options** entry field. (Command line options are listed in the Palm OS Simulator books.)
- Click **OK** to create your Simulator Run Target.

Using Your Palm OS Simulator Run Target

Now that you have created your Simulator Run Target, you can use it to test your application.

Testing 68K Applications using Palm OS Simulator

Palm OS Simulator runs 68K applications in PACE, the Palm OS Compatibility Environment.

- Compile your 68K application normally, using the `m68k-palmos-gcc` compiler.
- In Developer Suite, select the 68K project or PRC file you want to run, and select **Run**. This opens the Run dialog box.
- In the notebook control, select the **Target** tab.
- In the **Device** list, select the name that you specified when you created the Simulator Run Target.
- Click **Run** to run your application on the Simulator Run Target.

Palm OS Testing Tools

Using Palm OS Simulator

Testing 68K Applications with PNOs using Palm OS Simulator

Although Palm OS Simulator runs 68K applications in PACE, the PNO component needs to be compiled into a DLL file using the `gcc` compiler targeted for Windows.

Testing Palm OS Protein Applications using Palm OS Simulator

Palm OS Simulator does not run Palm OS Protein applications that have been compiled for ARM-based devices. To run a Palm OS Protein application on Palm OS Simulator, you need to compile the application into a DLL file using the `gcc` compiler targeted for Windows.

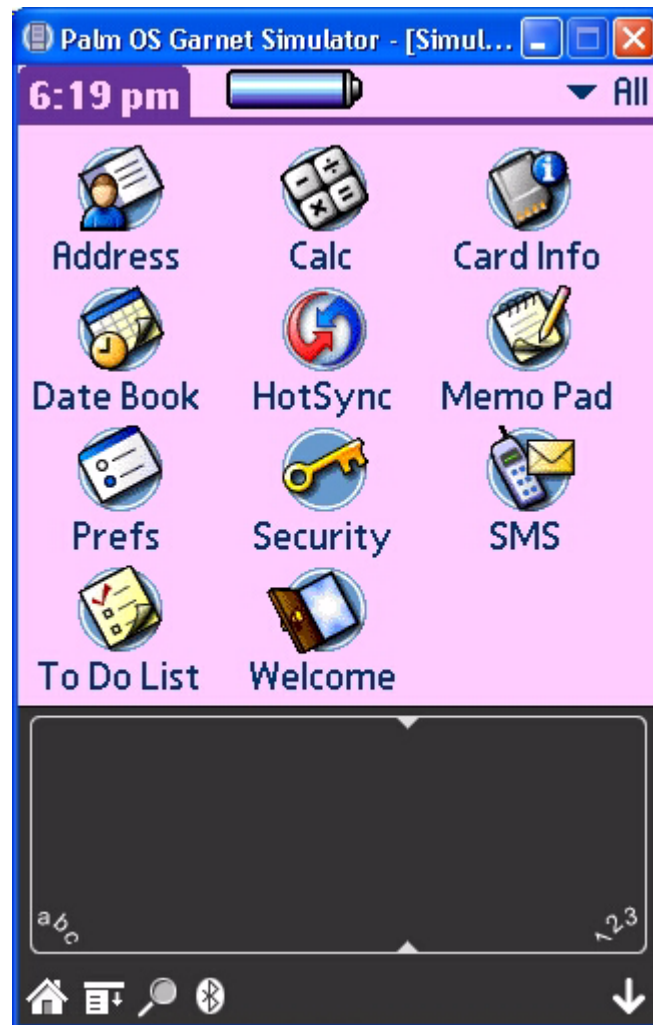
Palm OS Simulator User Interface

The Palm OS Simulator display looks very much like a real Palm Powered handheld device, as shown in [Figure 5.1](#). You can use your mouse to perform actions that you perform with the stylus on handheld devices, and you can use the menus to access Palm OS Simulator functions.

For more information on the user interface features, see the appropriate simulator book:

- *Testing with Palm OS Garnet Simulator*
- *Palm OS Cobalt Simulator Guide*

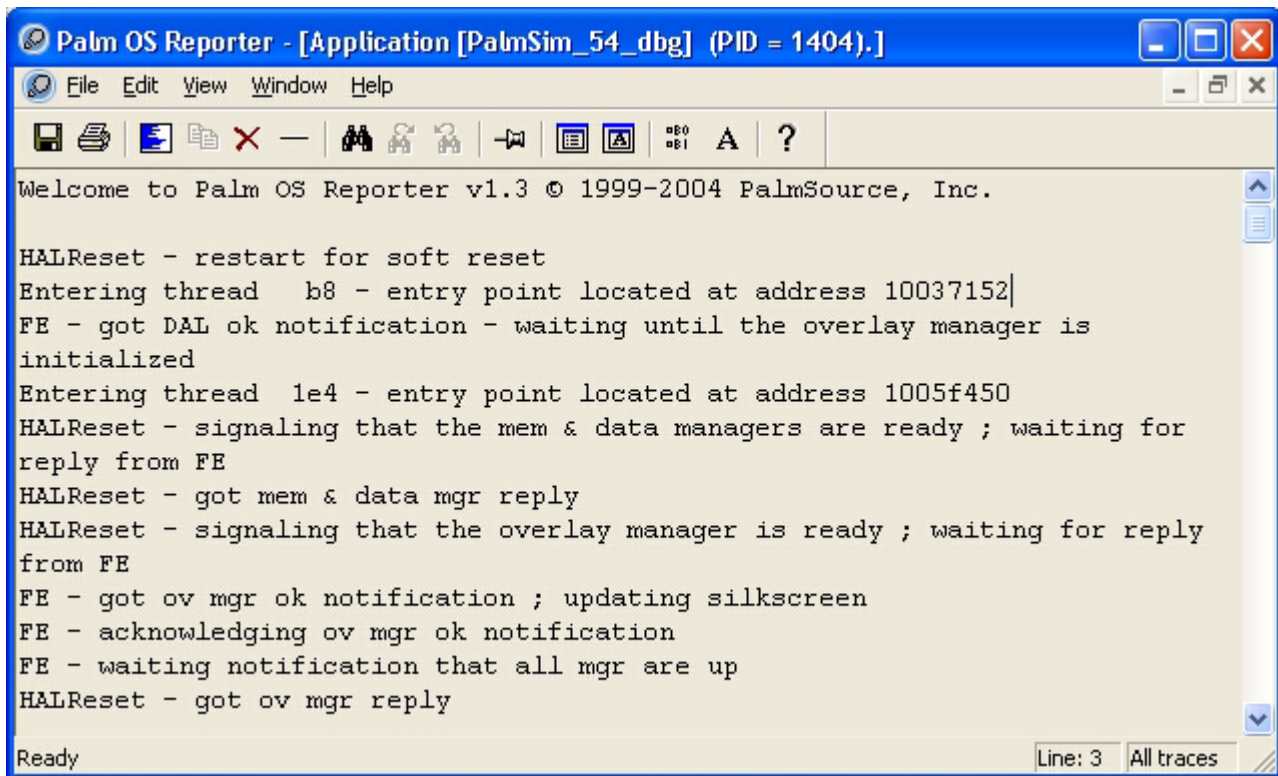
Figure 5.1 Palm OS Simulator user interface



Using Palm OS Reporter

Palm OS Reporter is a trace utility that can be used with Palm OS Simulator and with Palm OS Emulator. As an application runs on Simulator or on Emulator, the application can send information in real time to Reporter.

Figure 5.2 Palm OS Reporter main window



Using Reporter with 68K Applications

To use Reporter with 68K applications, add 68K trace calls to your application. These trace calls are listed in `hostcontrol.h`, which is part of the Palm OS SDK. For more information about the Host Control API, see the book *Exploring Palm OS: System Management*.

Using Reporter with Palm OS Protein Applications

To use Reporter with Palm OS Protein applications, add Trace Manager calls to your application. These Trace Manager calls are listed in `TraceMgr.h`, which is part of the Palm OS SDK. For more information about the Trace Manager API, see the book *Exploring Palm OS: System Management*.

Displaying Trace Information in Reporter

To view trace information in Palm Reporter, you need to do the following:

- Add trace calls to your application and build your application.
- Start a Palm Reporter session.
- If your run target is Simulator:
 - Start a Simulator session.
 - Redirect the Simulator's SocketLib calls to the host machine's TCP/IP stack. (Select the Simulator menu **Settings>Communications> Redirect SocketLib Calls to Host TCP/IP.**)
- If your run target is Emulator:
 - Start an Emulator session.
 - Redirect the Emulator's NetLib calls to the host machine's TCP/IP stack. (Select **Redirect NetLib calls to host TCP/IP** in the Emulator Properties panel.)
- Install your trace-enabled application in the Simulator or Emulator session.
- Run your trace-enabled application in the Simulator or Emulator session.

For more information about Palm OS Reporter, see *Testing with Palm OS Garnet Simulator* or *Palm OS Cobalt Simulator Guide*.

Using Palm OS Emulator

Palm OS Emulator is a hardware emulator program for the Palm Powered™ platform, which means that it emulates device hardware in software, providing you with the ability to test and debug Palm OS software on a desktop computer.

When you run a Palm OS application with Palm OS Emulator on your desktop computer, Palm OS Emulator fetches instructions, updates the handheld screen display, works with special registers, and handles interrupts in exactly the same manner as does the processor inside Palm Powered handhelds. The difference is that Palm OS Emulator executes these instructions in software on your desktop computer.

NOTE: Palm OS Emulator can run 68K applications; however, it cannot run applications that use PNOs or Palm OS Protein applications.

Creating a Palm OS Emulator Run Target

Developer Suite comes with a default Palm OS Emulator run target already created: Palm OS Emulator version 3.5. When you first run Palm OS Emulator, you select a ROM file. Developer Suite comes with two ROM files:

- PalmOS412_FullRel_EZ_enUS.rom, which is a Palm OS 4 release ROM file
- PalmOS412_FullDbg_EZ_enUS.rom, which is a Palm OS 4 debug ROM file

You can get other ROM files on the Palm OS developer web site:

http://www.palmos.com/dev/dl/dl_tools/dl_emulator/

To create additional Emulator run targets, follow these steps:

- From the Developer Suite interface, select **Window > Preferences**. This opens the Preferences dialog box.
- Open the category **Palm OS Development**, and select **Target Environment Settings**.

- Select the **Emulator** item in the list, and click **New** to show the Emulator preferences.
- Enter a name for your Emulator Run Target.
- Click **Browse** to select an Emulator executable that you want to use for this Emulator Run Target.
- Enter any additional command line options in the **Options** entry field. (Command line options are listed in the book *Using Palm OS Emulator*.)
- Click **OK** to create your Emulator Run Target.

Using Your Palm OS Emulator Run Target

Now that you have created your Emulator Run Target, you can use it to test your 68K application.

- In Developer Suite, select the project or PRC file you want to run, and select **Run**. This opens the Run dialog box.
- In the notebook control, select the **Target** tab.
- In the **Device** list, select the name that you specified when you created the Emulator Run Target.
- Click **Run** to run your application on the Emulator Run Target.

Palm OS Emulator User Interface

The Palm OS Emulator display looks very much like a real Palm Powered handheld device, as shown in [Figure 5.3](#) on page 86.

Figure 5.3 Palm OS Emulator user interface



You can use your mouse to perform actions that you perform with the stylus on handheld devices, and you can use the menus to access Palm OS Emulator functions. For more information on the user interface features, see *Using Palm OS Emulator*.

Using Virtual Phone

Virtual Phone is a tool which simulates a mobile telephone. Virtual Phone can help you develop and test applications which use the Telephony Manager API. Virtual Phone recognizes Telephony Manager AT commands and responds exactly the same as a mobile phone. Virtual Phone is also capable of simulating events like incoming voice calls and SMS messages.

Virtual Phone is designed to test applications which communicate with a mobile telephone. Virtual Phone is based upon the functioning of a standard GSM default phone driver. This means that any functions not supported by a standard GSM phone driver are not supported by Virtual Phone.

Launching Virtual Phone from the Workbench

Virtual Phone is included in the Developer Suite package, but to launch it directly from the Workbench, you need to define it as an external tool.

- Select **Run > External Tools > External Tools...** to open the External Tools dialog box.
- Select the **Program** category and click **New** to display the tool options.
- Enter a name for your tool configuration (replacing the text `New_configuration`).
- Click **Browse File System** and select the `VirtualPhone.exe` file you want to use, either the version in the 68K SDK (`sdk-5r4`) or the Palm OS Protein SDK (`sdk-6`).
- Click **Apply** to save the tool configuration.
- Click **Run** to run Virtual Phone.

Virtual Phone User Interface

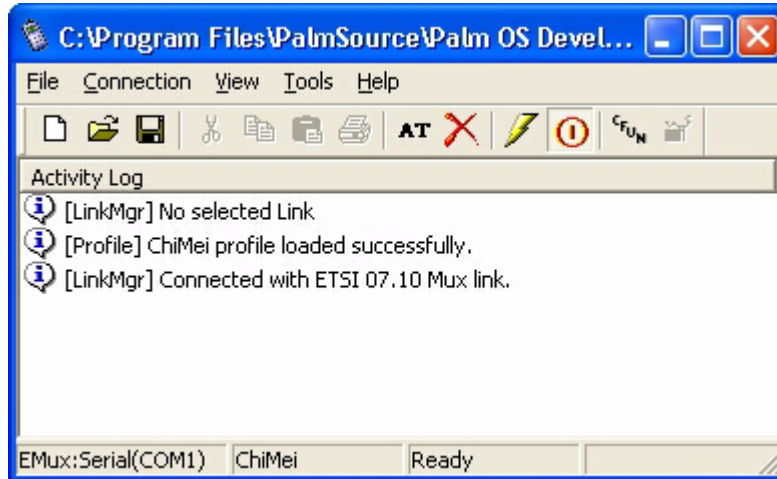
Virtual Phone works with Palm OS Simulator to test telephony applications. Every time a Palm OS application calls a Telephony Manager function, the Telephony Manager issues one or more AT commands which are then sent to Virtual Phone. When Virtual Phone receives these AT commands it responds exactly like a real phone.

The Virtual Phone window, as shown in [Figure 5.4](#), displays the text equivalent of the original Telephony Manager function, while the actual AT commands or traces are stored in a log file.

Palm OS Testing Tools

Getting More Information

Figure 5.4 Virtual Phone user interface



Getting More Information

Palm OS Garnet Simulator is described in detail in the book *Testing with Palm OS Garnet Simulator*; Palm OS Cobalt Simulator is described in detail in the book *Palm OS Cobalt Simulator Guide*.

Palm OS Emulator is described in detail in the book *Using Palm OS Emulator*.

Virtual Phone is described in detail in the book *Palm OS Virtual Phone Guide*.

Palm OS Debugging Tools

This chapter describes how to use the Palm OS debugging tools.

The Palm OS Developer Suite package provides the following debugging tools:

- “[Using Integrated Debugging](#)” on page 90 describes how to use the debugging features that are integrated in Palm OS Developer Suite.
- “[Using Palm OS Debugger](#)” on page 93 describes how to use the full-feature debugger, Palm OS Debugger.

Using Integrated Debugging

Palm OS Developer Suite provides integrated debugging support for all of the following:

- Application types:
 - 68K applications compiled using the GNU 68K compiler
 - 68K applications with PACE Native Objects
 - Palm OS Protein applications, including applications compiled using the GNU x86 compiler to run on Simulator targets, and applications compiled using the Palm OS Protein C/C++ Compiler to run on devices targets
- Target types: applications running on Palm OS Emulator, Palm OS Garnet Simulator, Palm OS Cobalt Simulator.
- Device communications connections: 68K applications can be debugged over serial or USB connections; Palm OS Protein applications require devices connected over USB, including:
 - Palm m500, m505, m125
 - PalmOne Tungsten T, T2, T3, C
 - PalmOne Zire 21, 71
 - PalmOne Treo 600 and 650
 - Handspring Visor series and Treo 180, 270
 - AlphaSmart Dana, Dana Wireless

Integrated Debugging Overview

Integrated debugging makes full use of the functions provided by the C/C++ Development Toolkit (CDT 2.0) that is integrated with Developer Suite. This support provided by CDT includes functions like:

- Specifying execution arguments
- Setting environment variables
- Working with breakpoints and watchpoints
- Step execution of an application
- Watching variables, expressions, registers, and memory

This section provides an overview of some of the commonly used functions. For more information on these integrated tasks, see *C/C++ Development Toolkit User Guide*.

Setting a Breakpoint in an Application

You set a breakpoint on an executable line of a program. If the breakpoint is enabled when you debug, the execution suspends before that line of code executes.

To add a breakpoint point, double click the marker bar located in the left margin of the C/C++ Editor beside the line of code where you want to add a breakpoint. A dot is displayed in the marker bar and in the Breakpoints view, along with the name of the associated file.

Creating a Debug Configuration

To debug your application using Palm OS Emulator or a Simulator target, you first create a debug configuration.

To create a debug configuration:

1. Select your project's icon in the C/C++ Projects view.
2. Select **Run > Debug** to open the Debug dialog box.
3. Click on the **Palm OS Application** icon and click New to create a configuration for the current project.
4. By default, the configuration's name will be the same as your project's.
5. Click the **Project** tab and be sure that your project is selected in the **Files to Install** list.
6. Click on the **Target** tab to choose the debugger target for this configuration. The dropdown lists include the defined targets. (To create additional targets, see the section "[Target Environment Settings Preferences](#)" on page 48.)
 - a. For 68K projects, there is a single dropdown list that includes all of the defined targets that can run 68K applications (device targets, Emulator targets, and Simulator targets).
 - b. For Palm OS Protein projects, there are two dropdown lists. The first is for device targets (applications compiled

Palm OS Debugging Tools

Using Integrated Debugging

with Palm OS Protein C/C++ Compiler, `pacc.exe`), and the second is for Simulator targets (applications compiled with GNU x86 compiler).

The reason for the differentiation is that Simulator builds create two files: a DLL that holds the code for your application (compiled as x86 code), and a PRC that holds your application's resources. For device (ARM) builds, only one file is created: the PRC, which holds everything, as with 68K applications.

7. Click **Debug** to start the debugging process.
8. The Workbench switches to the Debug perspective and stops at a temporary breakpoint inside `PilotMain()`. Click on the green Run arrow to begin execution. You can use the Step Over, Step Into and Step Out Of icons to control your debugging session (there are also commands in the Run menu, as well as keyboard equivalents). Note that the exact sequence of events that occur when you begin debugging depends on the target type (whether Simulator is already running, whether you are using Palm OS Garnet or Palm OS Cobalt Simulator, and so on).
9. When you are finished with the debugging session, click on the Palm OS Debugger line in the Debug view, then click the red square to stop the session.
10. Once you've created a debug configuration, just click the debug button in the toolbar to start another session.

Using Palm OS Debugger

Palm OS Debugger is a full-function debug tool that you can use to debug your Palm OS applications and shared libraries. Palm OS Debugger provides support for multiple symbolic file debugging.

Palm OS Debugger provides the following features:

- Full assembly and source level debugging
- Support for shared library debugging in ARM environments
- Support for debugging with multiple symbolic files
- Fully customizable short cut keys
- Variable and memory watch
- Setting and viewing breakpoints
- Support for ELF/DWARF 1.1 and 2.0

Palm OS Debugger is a tool for debugging both 68K-based and ARM-based Palm OS applications.

- **Debugging 68K-Based Applications:** Palm OS Debugger supports connecting to the 68K debug nub of a 68K-based device and to the 68K debug nub built into the PACE component of an ARM-based device. Palm OS Debugger also connects to the 68K debug nub of Palm OS Emulator and Palm OS Simulator.
- **Debugging ARM-Based Applications:** Palm OS Debugger supports connecting to the ARM debug nub of an ARM-based device.

Launching Palm OS Debugger from the Workbench

Palm OS Debugger is integrated into the Developer Suite package. To launch Palm OS Debugger, select **Run > Palm OS Debugger**. This opens the Palm OS Debugger's main window, as shown in [Figure 6.3](#) on page 96.

Setting Palm OS Debugger Preferences

Palm OS Debugger preferences are dependent on the run target selected. As a result, you can view and change Palm OS Debugger

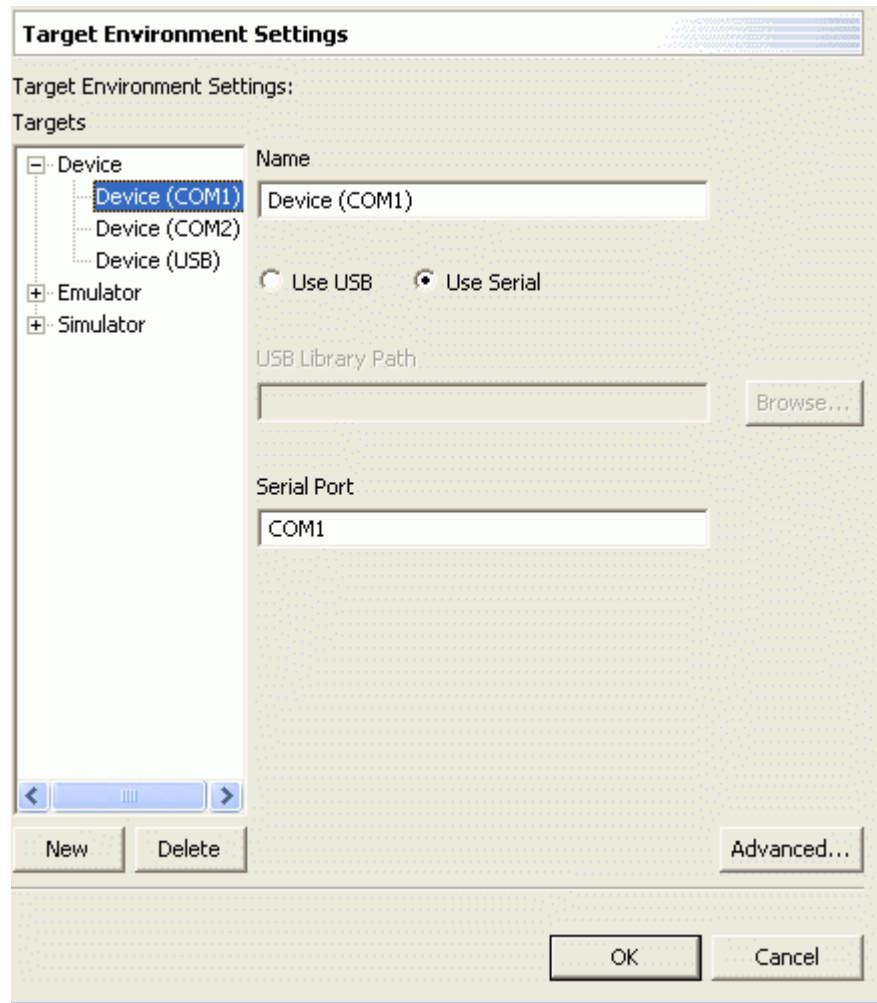
Palm OS Debugging Tools

Using Palm OS Debugger

preferences from the same dialog box where you set the run target preferences.

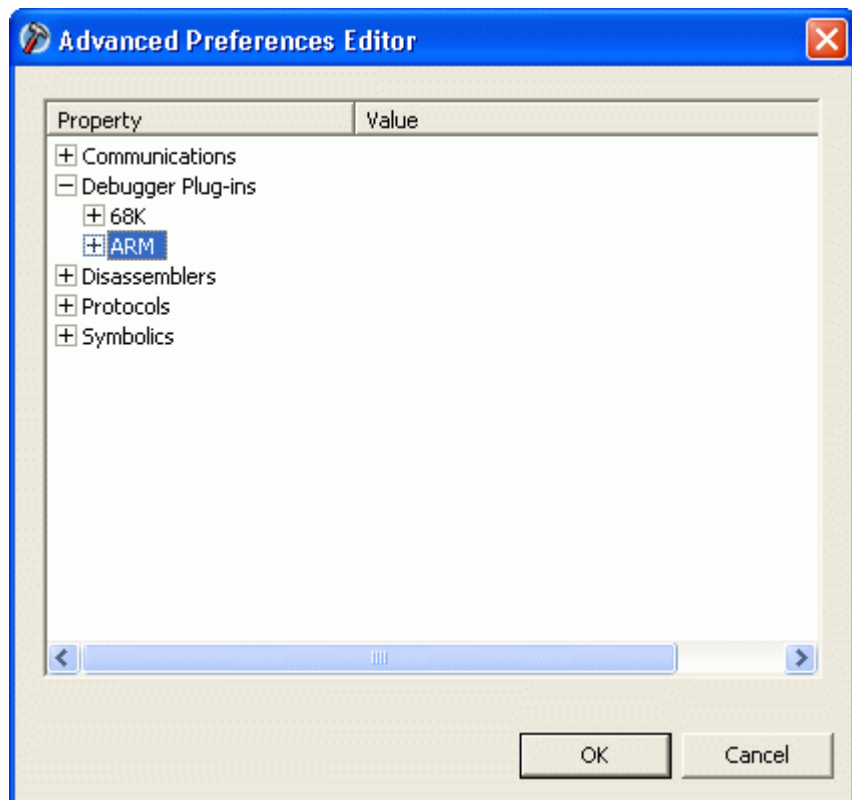
- Select **Window > Preferences** to open the Preferences dialog box.
- Expand the **Palm OS Development** category and select **Target Environment Settings**.
- Expand the category that matches the run target you are using, and select the specific run target. For example, in [Figure 6.1](#) below, the run target **Device (COM1)** is selected.

Figure 6.1 Target Environment Settings dialog box



- Click **Advanced** to open the Advanced Preferences Editor dialog box, as shown in [Figure 6.2](#).

Figure 6.2 Advanced Preferences Editor dialog box



You can open each of the properties listed in the Advanced Preferences Editor to view or change a specific Palm OS Debugger setting. For more information about the Palm OS Debugger settings, see the book *Palm OS Debugger Guide*.

Palm OS Debugger User Interface

You can use Palm OS Debugger to:

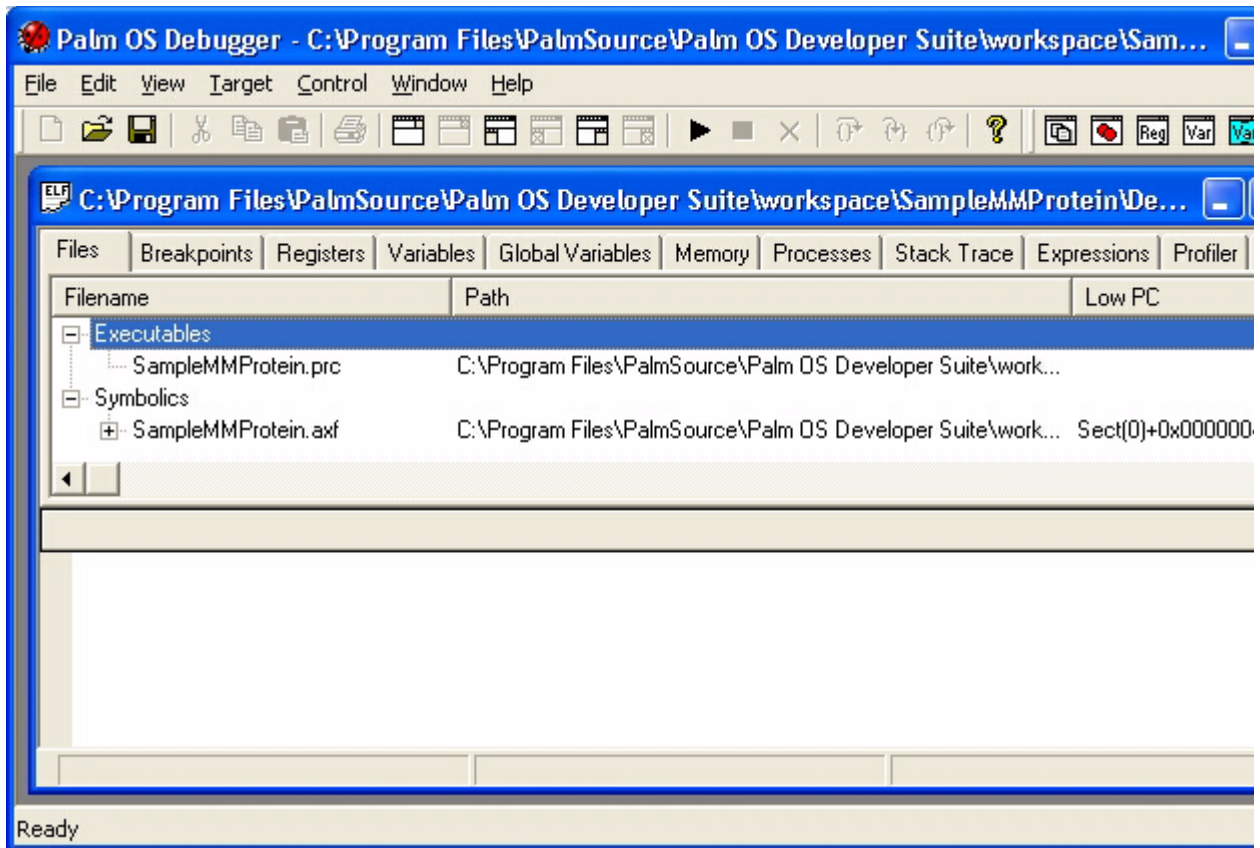
- View information about applications and their source files.
- Set breakpoints.
- Display a list of registers and their values.
- View information about variables.

Palm OS Debugging Tools

Using Palm OS Debugger

- View information about memory address locations and their contents.
- View processes that Palm OS Debugger knows about.
- View the stack trace.
- Evaluate C expressions.
- View standard input and output operations (such as debug messages) in an STDIO console.
- Interact with your debugging session in a command line environment.

Figure 6.3 Palm OS Debugger user interface



You can open an instance any of the Palm OS Debugger tabs in its own separate window. To do this, select Windows in the Palm OS Debugger menu bar, and then select a view from the list of views.

Getting More Information

Palm OS Debugger is described in detail in the book *Palm OS Debugger Guide*.

Palm OS Debugging Tools

Getting More Information

Palm OS Package Builder

This chapter describes how to use the integrated version of Palm OS® Package Builder to create PalmSource Install (PSI) files.

Palm OS Package Builder Overview

Palm OS Package Builder is a component of PalmSource Installer family of tools. PalmSource Installer is designed to streamline the online and over-the-air (OTA) download and install process of Palm OS applications onto Palm Powered™ smartphones and other wireless devices.

Palm OS Package Builder is a developer utility for packaging application files into a single, compressed PalmSource Installer (PSI) file. This package file facilitates the downloading and installation of the application and support files.

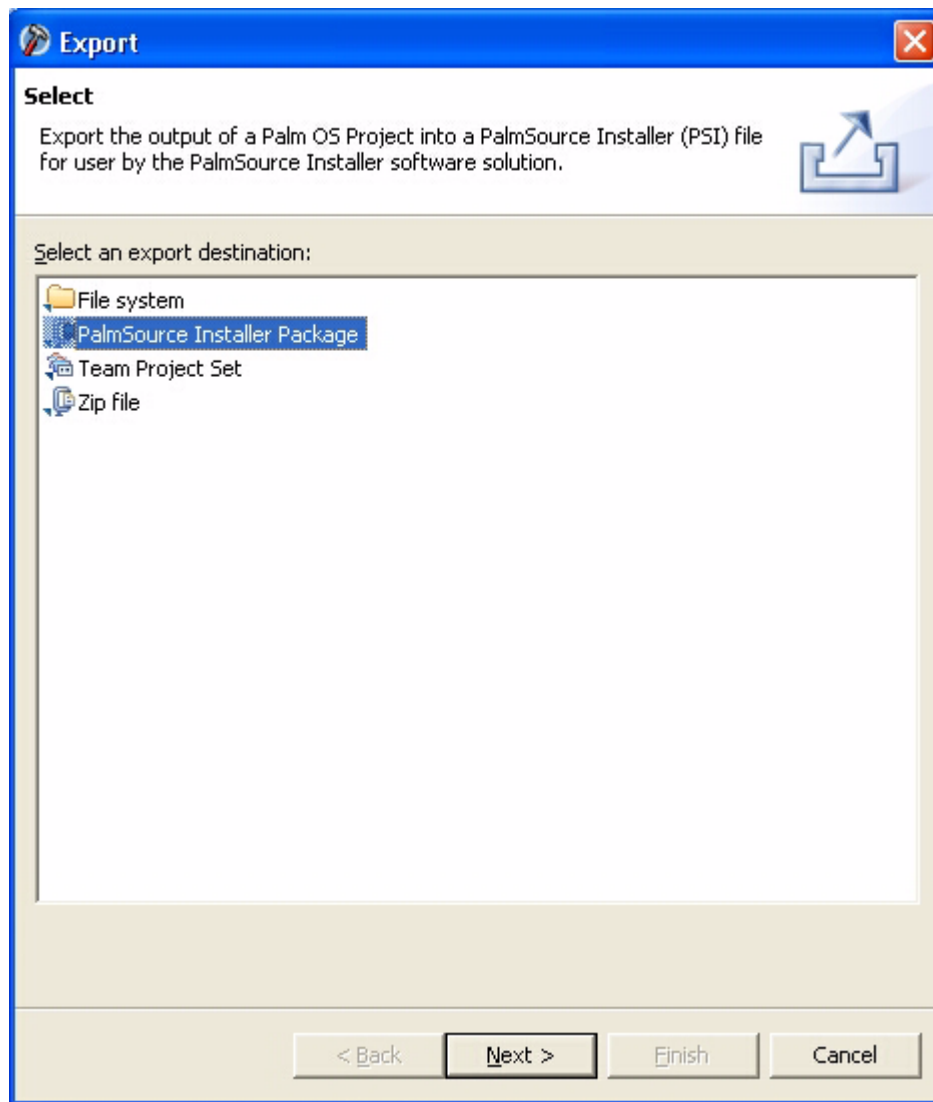
Using the provided XML schema and starter file, you can create a definition file that specifies which files go into the package (including desktop components). Additional OTA files are created for each language and screen combination for direct downloading to specific device environments.

Using Palm OS Package Builder

Palm OS Package Builder is integrated with Developer Suite as an Export function. To use Palm OS Package Builder, follow these steps.

- Select **File > Export** to open the Export dialog box, shown in [Figure 7.1](#) on page 100.

Figure 7.1 Export dialog box



- Select **PalmSource Installer Package**.
- In next page of the Export wizard, select the project you want to export as a package installer file.

When you click **Finish**, Palm OS Package Builder creates the following files:

- `package.psm1`, an XML file that defines the package.
- a PSI file, which is the PalmSource Installer package file.
- a `filename-ota.PSI` file, which is the component that can be installed “over-the-air” to a Palm Powered handheld.

The version of Palm OS Package Builder that is installed as an Export function uses many default settings.

Palm OS Package Builder can also be used as a command line tool, in case you need to use non-default settings. For additional information on Palm OS Package Builder and other PalmSource Installer components, see the book *PalmSource Installer Developer Guide*.

Palm OS Package Builder

Using Palm OS Package Builder

Hints and Tips

This appendix includes some trouble-shooting information that may help you answer questions about Palm OS Developer Suite.

Build Issues

#include Search Behavior

In the current Workbench implementation, `#include` statements of the form

```
#include <header.h>
```

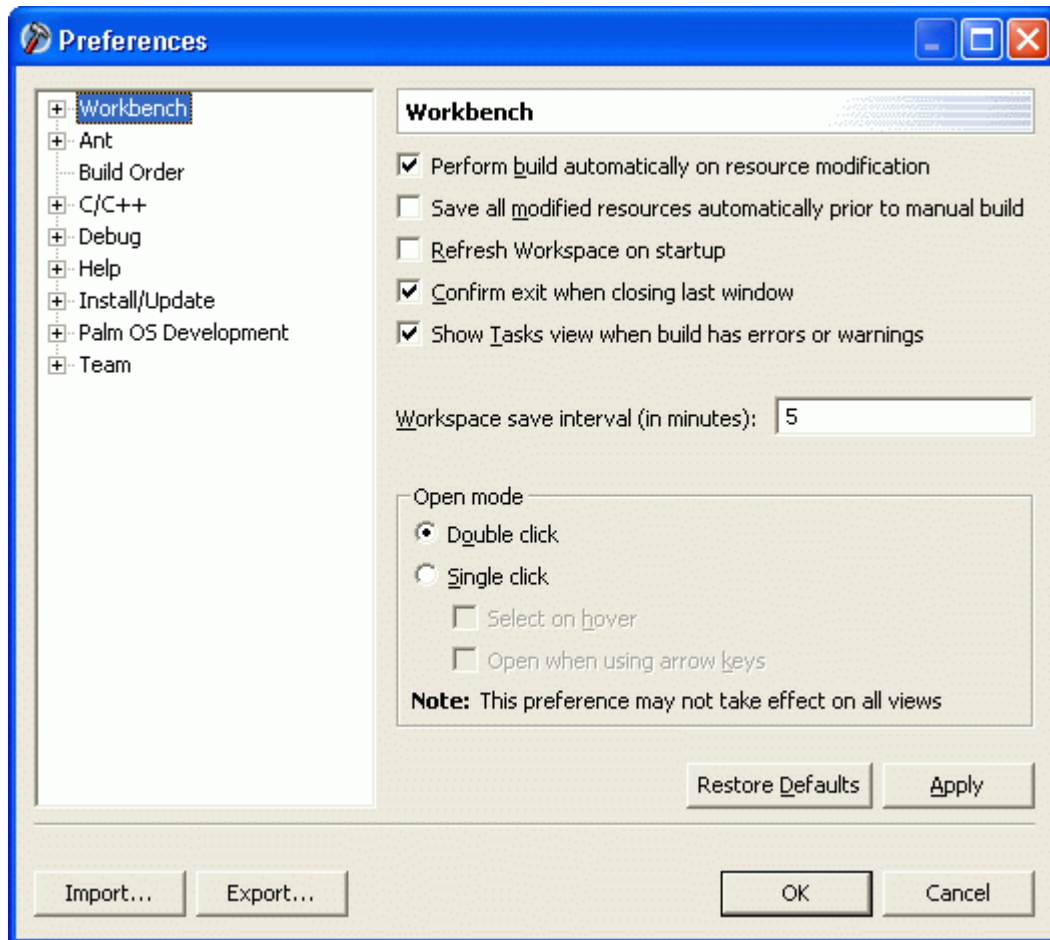
search both the standard directories, and the user's include directories.

Automatically Saving Changes

If you want all of your sources automatically saved prior to a build, you need to set an Eclipse Workbench preference.

- Select **Window > Preferences** to open the Preferences dialog box, as shown in [Figure A.1](#) on page 104.

Figure A.1 Workbench Preferences dialog box



- Select **Workbench** to display the Workbench Preferences page.
- Select **Save all modified resources automatically prior to manual build**.

gcc Compiler Issues

Exception Handling

Because of differences between the gcc compiler and the Palm OS Protein C/C++ Compiler, exception handling—using `try()`, `catch()`, and `throw()`—does not work in Palm OS Protein C++ code that is compiled for the Palm OS Cobalt Simulator.

In addition, the functions `setjmp()` and `longjmp()` from the header file `ErrTryCatch.h` produce linking errors if used in Palm OS Protein applications. To work around this limitation, add the following `#define` statements to get your code to compile and link:

```
#define longjmp ErrLongJump
#define jmp_buf ErrJumpBuf
```

Long Double Data Type

Because of differences between compiler implementations of the `long double` data type, you should not use the `long double` data type in code that is compiled by the gcc compiler.

For example, gcc implements `long double` data as 12-byte data, but Palm OS Simulator treats `long double` data as 8-byte data.

In addition, the gcc option `-m128bit-long-double` treats `long double` data as 16-byte data rather than 12-byte data.

gdb Debugger Issues

Breakpoints on Opening Braces

Because of a gdb debugger issue, if you place a breakpoint on an opening brace, you may not be able to use the **Step Over** function when debugging code running on Palm OS Cobalt Simulator.

To correct this issue, move the breakpoint to a subsequent line, or remove the breakpoint on the open brace after hitting it and before using the **Step Over** function.

Hints and Tips

68K Applications

Resume At Line Function

Because of a known issue in the interaction between CDT and the debugger, you should not use the **Resume At Line** function when debugging code running on Palm OS Cobalt Simulator.

68K Applications

Debugging Shared Libraries

Palm OS Debugger does not support the Step Into function for shared library code with 68K shared library calls.

68K shared library calls are implemented as system traps that contain the parameters on the stack, and one of those parameters is a function index to be called from a function table. The code that implements shared library function tables is in the ROM, which prevents stepping into 68K shared library functions.

Index

A

- advanced preferences editor 95
- application development process 1
- as 12, 13
- assembler
 - as 12, 13

B

- book organization viii
- building a project 44

C

- code signing tool 74
- CodeWarrior
 - importing files 36
- compiler
 - gcc 12, 13
 - m68k-palmos-gcc 4
 - Palm OS Protein C/C++ compiler 6, 11
- compiler options 60
- compiler tools component 59
- compiler tools overview 11
- components
 - compiler tools 11
 - Eclipse integration 11
 - Palm OS Debugger 15
 - Palm OS Simulator 15
 - resource tools 13
 - Virtual Phone 15
- configuration
 - device target 45
 - Palm OS Protein application 46
 - PNO 45
 - simulator target 45
- creating a Palm OS project 25

D

- debug targets 45
- debugger 93
 - gdb 13
- debugging tools 89
- developer suite components 11
- developer tools xii
- developer tools package 11

- device simulator 78
- documentation viii
 - web site ix

E

- Eclipse
 - help system viii
- Eclipse Workbench 11
- elfdump 63

G

- gcc 12, 13
- gdb 13
- generateXRD 74
- GNU assembler 12, 13
- GNU compiler collection 12, 13
- GNU debugger 13
- GNU linker 12, 13

H

- help browser viii

I

- importing CodeWarrior files 36
- importing PRC-Tools files 36

K

- knowledge base ix

L

- ld 12, 13
- linker
 - ld 12, 13

M

- mobile phone simulator 86
- multi-segment application 5

P

- paasm 62
- pacc 62
- palib 63

palink 62
Palm OS Debugger 89, 93
Palm OS Debugger overview 15
Palm OS Developer Suite xii
Palm OS Package Builder 99
Palm OS Protein C/C++ Compiler 6, 11
Palm OS Resource Editor 74
Palm OS Simulator 78
Palm OS Simulator overview 15
palmrc 74
PalmRC options 67, 69
palmsim 78
PalmSource Installer 99
pelf2bin 8
phone simulator 86
post linker 8
prccert 74
prccompare 74
prcmerge 74
prcsign 74
PSI file 99

R

resource build tool 74
resource comparison tool 74
resource compiler options 67, 69
resource editor tool 74
resource migration tool 74

resource security tool 74
resource tools component 65
resource tools overview 13

S

setting compiler options 60
setting debugger options 95
setting resource compiler options 67, 69

T

telephone simulator 86
testing tools 77
tools documentation xi, xii
 Introduction to Palm OS Tools xii
 Language and Library Reference xii
 Palm OS Debugger xiii
 Palm OS Resource Editor xiii
 Resource Tools xiii
 Virtual Phone xiii
tools overview 1
training ix

V

Virtual Phone 86
Virtual Phone overview 15

W

Workbench integration overview 11