Palm® Developer Guide, Palm OS® Platform

Software and Hardware

Rev. J

April 30, 2008

Usage Restrictions

Your right to access and use the following "PDN Materials" is subject to and governed by the following terms and the terms and conditions of the Palm Developer Network Program License Agreement, available at **https://pdn.palm.com/regac/pdn/page?Page_Name=terms**.

Disclaimer and Limitation of Liability

Palm, Inc. ("Palm") provides these PDN Materials as a courtesy to members of its Palm Developer Network ("PDN"). Palm provides the PDN Materials on an "AS IS" basis and does not guarantee nor assume any liability for the accuracy or completeness of these PDN Materials, nor does Palm guarantee that the PDN Materials will be without errors. You assume the entire risk of using the PDN Materials, including any damage or loss resulting from your use of the PDN Materials or any loss or claims by third parties which may arise from your use of the PDN Materials.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE PDN MATERIALS ARE PROVIDED TO YOU ON AN "AS IS" BASIS AND WITHOUT ANY WARRANTY OF ANY KIND OR NATURE. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, PALM AND ITS SUPPLIERS EXPRESSLY DISCLAIM ANY IMPLIED OR STATUTORY WARRANTIES, INCLUDING WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, ACCURACY, CORRESPONDENCE WITH DESCRIPTION, SATISFACTORY QUALITY AND NON-INFRINGEMENT.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, NEITHER PALM NOR ITS SUPPLIERS SHALL BE LIABLE FOR DIRECT, INCIDENTAL, CONSEQUENTIAL, INDIRECT, SPECIAL, OR PUNITIVE DAMAGES OF ANY KIND, LOSS OF INFORMATION OR DATA, LOSS OF REVENUE, LOSS OF BUSINESS OR OTHER FINANCIAL LOSS ARISING OUT OF OR IN CONNECTION WITH THE USE OF THE PDN MATERIALS, WHETHER BASED IN CONTRACT, TORT (INCLUDING NEGLIGENCE), STRICT PRODUCT LIABILITY OR ANY OTHER THEORY, EVEN IF PALM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES AND EVEN IF ANY LIMITED REMEDY IS DEEMED TO HAVE FAILED OF ITS ESSENTIAL PURPOSE. Some countries, states, or provinces do not allow the limitation of incidental or consequential damages so the above provisions may be limited in their application to you.

Copyright and Trademark

© 2004-2008 Palm, Inc. All rights reserved.

Palm, LifeDrive, Zire, Tungsten, Treo, Centro, Blazer, Handspring, Graffiti, HotSync, VersaMail, Palm Powered, and Palm OS by ACCESS are among the trademarks or registered trademarks owned by or licensed to Palm, Inc. Adobe and Acrobat are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. All other brand and product names are or may be trademarks of, and are used to identify products or services of, their respective owners.

The foregoing provisions shall be governed by the laws of the State of California and by the federal laws of the United States, excluding their conflicts of laws provisions. The United Nations Convention on Contracts for the International Sale of Goods (1980) is hereby excluded in its entirety from application to the foregoing provisions. In the event any provision is found to be invalid, illegal or unenforceable, the validity, legality and enforceability of any of the remaining provisions shall not in any way be affected or impaired.

Please read the Palm Developer Network Program License Agreement before using the PDN Materials. If you do not agree to the terms and conditions of the Developer Network Program License Agreement, you may not access or use the PDN Materials. USING OR ACCESSING ANY PART OF THE PDN MATERIALS INDICATES THAT YOU ACCEPT THE TERMS OF THE DEVELOPER NETWORK PROGRAM LICENSE AGREEMENT.

Document Revision History

Date	Revision	Description of Changes
2004 - 2005	A	Documents prior to document revisioning.
May 15, 2006	B	Revisions for the Treo™ 700p smartphone.
June 29, 2006	C	Revisions for Multimedia features.
November 20, 2006	D	Revisions for the Treo™ 700p smartphone Rest of World carrier release.
November 22, 2006	E	Revisions for the Treo™ 680 smartphone.
April 30, 2007	F	Revisions for the Treo™ 755p smartphone.
October 10, 2007	G	Revisions for the Centro™ smartphone, CDMA version.
February 1, 2008	H	Revisions for the Centro™ smartphone, GSM version.
April 30, 2008	J	Revisions for the Centro™ smartphone, CDMA MR.

This page intentionally blank

Contents

What's New . . . . . **xiii**
What's new in Centro™ smartphones xiii

Part I: Introduction

Chapter 1: Overview **17**
1.1 How this guide is organized 17
1.2 Typographical conventions 18
1.3 Additional documentation and resources 19
1.3.1 Palm® Customer Service and Support 19
1.3.2 PDN Knowledge Base 19
1.3.3 ACCESS® documentation 19
1.3.4 Expansion Parts Store 19
1.3.5 Cable and connector drawings 19
1.4 Palm® developer marketing programs 20
1.4.1 Designed for Palm® Products Program 20
1.4.2 Device loaner program 20
1.4.3 Device discount program 20
1.5 Submitting documentation feedback 20

Chapter 2: Quick Start Guide **21**
2.1 Introduction 21
2.2 ACCESS® requirements 22
2.2.1 Join the ACCESS® Developer Program 22
2.2.2 Download the ACCESS® Palm OS® SDK 22
2.3 Development environment 22
2.4 Palm® requirements 23
2.4.1 Join the Palm® Developer Network (PDN) 23
2.4.2 Download the Palm OS® Platform SDK 24

Chapter 3: Product Line Overview **27**
3.1 Centro™ smartphones 27
3.1.1 What's not supported by Centro™ smartphones 28
3.2 Treo™ smartphone product line 29
3.2.1 What's not supported by Treo™ smartphones 30
3.3 Tungsten™ handheld product line 31

3.3.1 What's not supported by Tungsten™ handhelds 31
3.4 LifeDrive™ mobile manager 32
3.4.1 What's not supported by LifeDrive™ mobile managers 32
3.5 Zire™ handheld product line 33
3.5.1 What's not supported by Zire™ handhelds 33
3.6 Hardware feature matrix 34
3.6.1 Palm® smartphone hardware features 34
3.6.2 Palm® handheld and LifeDrive™ mobile manager hardware features 36
3.6.3 Palm® Z22 organizer and Zire™ handhelds hardware features 37
3.7 SDIO support 39
3.8 Software compatibility specifications (Palm® libraries) 40
3.8.1 Palm® smartphones 40
3.8.2 Palm® handhelds, organizers, and mobile managers 42
3.9 sysExternalConnectorAttachEvent and sysExternalConnectorDetachEvent notifications 44
3.10 kPmConnectorClass Notifications 44

Part II: Features and Libraries

Chapter 4: PIM SDK 47
4.1 The PIM SDK 47
4.2 Known issue 47

Chapter 5: Multimedia 49
5.1 Codec Plug-in Manager 50
5.1.1 Codec Plug-in Manager overview 50
5.1.2 Codec wrapping 52
5.1.3 Codec Plug-in Manager process 53
5.1.4 Media codec formats supported by device 56
5.1.5 For more information 58
5.2 Imaging 58
5.2.1 Photo Library 59
5.2.2 LCD Overlay 60
5.2.3 Camera Manager 62
5.2.4 JPEGLib, CameraLib, and ImageLib 64
5.3 Audio 66
5.3.1 Voice recording and sound libraries 66
5.3.2 Tones library 68
5.4 Video playback 72
5.5 Streaming 73
5.5.1 Best practices for encoding video for streaming 73
5.5.2 Known issue 73

Chapter 6: Data Communications 75
6.1 NetPref Library API 75
6.1.1 Loading the library 75
6.1.2 NetPref Library information 77
6.1.3 NetPref panel 77
6.2 NetMaster library API 80
6.2.1 Usage model 81
6.2.2 Loading the library 82
6.2.3 Library information 83
6.3 Email client best practices 84
6.4 HTTP library 85
6.4.1 Architecture 85
6.4.2 Functional highlights 86
6.4.3 httpErrorLibraryAlreadyOpen error message for Treo™ 680 smartphones 87
6.4.4 HTTP library interface to SSL 87
6.4.5 HTTP library use of certificates/public key infrastructure 88
6.4.6 HTTP library implementation 88
6.4.7 General HTTP program information 89
6.5 Net Services API 93
6.5.1 Overview of the Net Services feature 93

Chapter 7: HTML Library 95
7.1 Architecture 95
7.2 Usage model 96
7.3 Image rendering 97
7.4 Debugging 97

Chapter 8: Telephony 99
8.1 Overview of the Telephony API libraries 99
8.1.1 CDMA and GSM libraries 101
8.1.2 Using indicators 102
8.1.3 Using PhnLibRegister 103
8.1.4 Getting the Cell ID on Centro™ smartphones 103
8.1.5 PhnLibGetMMSUAString() 104
8.1.6 PhnMsgBoxDataType 104
8.2 Phone Application 105
8.2.1 Phone Application 2.5 105
8.2.2 Phone Application 3.0 and later 105
8.2.3 Launching the Phone application in a specific view 107
8.2.4 Launching the Phone application in Dial Pad view 108
8.2.5 Launching the Phone application in the Favorites view 110

8.3 Launching the Contacts application with the New Contact window open 111
8.4 EvDO on Palm® smartphones 112
8.4.1 Detecting EvDO vs. 1xRTT 112
8.4.2 Troubleshooting incoming voice calls 113

Chapter 9: SMS 115
9.1 What is the difference between SMS and NBS? 115
9.2 SMS library 115
9.3 What is SMS? 116
9.4 Why use the SMS library? 116
9.5 Understanding the SMS library 116
9.5.1 Incoming SMS messages and message events 117
9.5.2 Outgoing SMS messages 118
9.5.3 Handling the GSM alphabet and Palm OS® alphabet 119
9.5.4 Message segmentation 120
9.5.5 Message database 121
9.6 Launching SMS from the New SMS screen 123

Chapter 10: System Extensions 125
10.1 Transparency API 125
10.2 File Browser API 127
10.3 Smart Text Engine (STE) API 132
10.3.1 STE architecture 133
10.4 REM Sleep API 135
10.4.1 Normal sleep deferral 135
10.4.2 REM sleep mode 136
10.4.3 Detecting REM sleep mode 137
10.4.4 LCD on/off notification 137
10.4.5 Waking up from REM sleep mode 139
10.5 Keyguard API 139
10.6 Option and Shift key APIs 140
10.7 MMS helper functions API 141
10.7.1 MMS usage model 141
10.7.2 MMS sample code 142
10.8 NVFS 144
10.8.1 Differences between NOR and NAND flash memory 147
10.8.2 Database layout on NVFS devices 148
10.8.3 Programming on devices that have NVFS 151
10.8.4 Optimizing your application for Palm® NVFS devices 155
10.9 5-Way Navigator and Keyboard API 157
10.9.1 5-way navigator terminology 157
10.9.2 Overview of 5-way navigator 158

10.9.3 Navigation events 158
10.9.4 Including objects as skipped objects 159
10.9.5 Default navigation 159
10.9.6 Custom navigation 162
10.9.7 Focus treatment 164
10.9.8 Navigational API, Button Mapping, and behavioral differences between Palm smartphones and Tungsten™ T5 handhelds 165
10.9.9 Tips and troubleshooting 178
10.10 Handspring® extensions 180
10.11 Tips and Tutorials 181
10.11.1 Terminology 181
10.11.2 Content 182
10.11.3 Tips and Tutorial structure 182
10.11.4 Converting Tips and Tutorial content in a PRC file 192
10.11.5 Displaying Tips and Tutorial content 193
10.11.6 Graphic element design guidelines 194
10.12 Full-Screen Writing API 196
10.13 Dynamic Input Area (DIA) 197

Chapter 11: Applications 199
11.1 Web Browser API 199
11.1.1 How the web browser works 200
11.1.2 Web browser feature overview 201
11.1.3 Download manager 203
11.1.4 Launching the web browser on Palm™ smartphones 211
11.1.5 Launching the web browser in minimal mode 212
11.2 VersaMail® application API 213
11.2.1 Before using the VersaMail® Device APIs 213
11.2.2 Overview of the VersaMail® Device APIs 213
11.2.3 Adding outgoing email to VersaMail® folders 216
11.2.4 VersaMail® Font library 219
11.2.5 VersaMail® Attachment Plug-ins API 220

Chapter 12: Developing SDIO Applications for Palm® Handhelds 223
12.1 SD, SDIO, and MultiMediaCard specifications 223
12.2 Palm OS® SDK 224
12.3 Software architecture of an SDIO application 224
12.3.1 Expansion Manager 225
12.3.2 VFS Manager 225
12.3.3 SDIO slot driver 226
12.3.4 Notification Manager 226
12.4 Guidelines for SDIO applications 226
12.4.1 Power management 227

12.4.2 Interrupt handling 228
12.4.3 Detecting card insertion and removal 228
12.4.4 Auto Run 229
12.5 Developing the SDIO peripheral 229
12.5.1 Specifications 229
12.5.2 SDIO slot driver 230
12.5.3 SDIO card initialization and identification on Palm OS 230
12.5.4 Code Storage Area (CSA) 231

Part III: Debugging

Chapter 13: Debugging 235
13.1 Overview 235
13.1.1 Hardware requirements 235
13.1.2 Simulator vs. on-device debugging 235
13.1.3 Debugger modes 236
13.2 Simulators and emulators 238
13.2.1 What's the difference? 238
13.2.2 Where can I get them? 238
13.2.3 Palm OS simulators and emulators 239
13.2.4 Release Simulator 239
13.2.5 Debug Simulator 242
13.3 DebugPrefs 243
13.3.1 Checkbox settings 243
13.3.2 Button options 245
13.4 Metrowerks CodeWarrior 246
13.4.1 Simulator debugging using CodeWarrior 247
13.4.2 On-device debugging using CodeWarrior 248
13.4.3 CodeWarrior debugging troubleshooting tips 249
13.5 PalmDebugger 250
13.5.1 Source level debugging 251
13.5.2 Post-crash debugging 251
13.5.3 Common PalmDebugger commands 252
13.5.4 Using PalmDebugger to import and export files 253
13.5.5 PalmDebugger tips 254
13.6 Garnet OS Developer Suite 255
13.7 Resets 256
13.7.1 Soft Reset 256
13.7.2 Warm Reset 256
13.7.3 Hard Reset 256
13.7.4 Factory Reset 257

Part IV: Style Guide

Chapter 14: Style Guide 261
14.1 Designing pages for the Blazer® web browser 261
14.1.1 General rules for web page design 261
14.1.2 Screen resolution 262
14.1.3 Connection speed 262
14.1.4 Content 262
14.1.5 Working with the Blazer® web browser 273
14.1.6 Testing your website 275
14.1.7 International support 275
14.1.8 List of acronyms 276
14.1.9 Palm OS® integration tags 277
14.2 Gadgets 279
14.2.1 Required headers and libraries 279
14.2.2 Overlapping gadgets in Treo™ 680 279
14.2.3 How to include the Battery gadget 279
14.2.4 How to include the Signal gadget 280
14.2.5 How to include the Bluetooth® wireless technology gadget 280

Part V: Hardware Developers Kit

Chapter 15: Multi-connector Specifications 283
15.1 Overview 283
15.2 Pinout of the Multi-connector 284
15.2.1 USB 286
15.2.2 Serial interface hardware 286
15.2.3 Serial interface software 287
15.2.4 HotSync® interrupt hardware 288
15.2.5 HotSync® interrupt software 288
15.2.6 Power output 288
15.2.7 Audio detection 289
15.2.8 Audio output 289
15.3 Peripheral requirements 290
15.3.1 Audio peripherals 291
15.3.2 General serial peripherals 291
15.4 Peripheral detection 292
15.4.1 Class-level detection 292
15.4.2 Audio peripheral detection timing diagrams 294
15.4.3 Serial peripheral detection timing diagram 295
15.4.4 Peripheral detection timing specifications 295
15.5 Interfacing with an audio peripheral 296

15.6 Interfacing with a serial peripheral 298
15.6.1 Electrical diagram of a serial peripheral 299
15.6.2 Serial peripheral design guidelines 300
15.7 Serial Peripheral Usage 301
15.7.1 Serial support on the Multi-connector interface 301
15.7.2 How to use the serial port 301
15.7.3 Multi-connector peripheral attach and detach notifications 301
15.7.4 Known Issues 302
15.7.5 Coding example 303
15.7.6 Serial peripheral detection 304
15.7.7 Connector library (PmConnectorLib) 305
15.7.8 Serial HotSync 307
15.7.9 Known Issues 307
15.8 Interfacing with Smart serial peripherals 308
15.8.1 Smart serial peripheral handshaking process 308

Chapter 16: Headset Jack Specifications 315
16.1 Overview 315
16.1.1 Standard 2.5mm cell phone headset (3-pin) 316
16.1.2 Stereo headphones (3-pin, 2.5mm or 3.5mm via adapter) 316
16.1.3 (Custom) Combination headphone/headset (4-pin, 2.5mm) 316
16.2 Stereo audio accessories 317
16.3 Microphones 318
16.4 Speaker Architecture 319
16.5 Usage scenarios 321
16.5.1 Treo™ 600 smartphones 321
16.5.2 Treo™ 650 smartphones and later 324

Chapter 17: External Hardware Drawings 325
17.1 Palm™ devices 325
17.2 Expansion Parts Store 325
17.3 Cable and connector drawings 325

Part VI: Appendix

Appendix A: Sample Code 329

Index 335

What's New

This section provides an overview of new features and product developments that have been implemented since the last revision of this guide.

Overall, the current revision of this document (Rev. J) reflects updates to features and specifications for the Palm® Centro™ smartphone maintenance release, CDMA radio version.

Specifically, the following items have been added or updated in the current revision of the *Palm Developer Guide, Palm OS Platform*:

- Added information on JPEGLib, CameraLib, and ImageLib to **Section 5.2.4 on page 64**.
- Added information on getting the Cell ID for Centro CDMA smartphones to **Section 8.1.4.2 on page 103**.
- Added information on a Verizon-specific API, PhnLibGetMMSUAString(), to **Section 8.1.5 on page 104**.
- Added information on a Verizon-specific priority field, PhnMsgBoxDataType, to **Section 8.1.6 on page 104**.
- Corrected code sample in **Section 8.2.4.2 on page 108**.
- Added information on resets to **Section 13.7 on page 256**.

What's new in Centro™ smartphones

The Centro smartphone is the newest addition to Palm's line of smartphone products. Centro is Palm's smallest form factor smartphone to date, and is available in two radio versions, CDMA and GSM:

- The Centro for GSM networks is a quad-band GSM/GPRS/EDGE world device, and is based on the Treo 680 base applications.
- The Centro for CDMA networks is a dual band 1xRTT/EvDO device, and is based on the Treo 755p base applications.

Hardware features for both GSM and CDMA versions include:

- Smaller removable battery (1150mAh)
- 64MB of RAM, and 64MB of NAND Flash
- Support for microSD cards of up to 4GB (no SDIO support)
- New color form factors vary by carrier.

The Centro smartphone runs on Palm OS® 5.4.9 by ACCESS. Centro also includes on-device registration and over-the-air (OTA) download of bonus software using MyCentro. For complete hardware and software specifications, see **Chapter 3**.

The latest header files are available as part of the Palm OS Platform SDK 5.6. Download the SDK from the Palm Developer Network website today.

New Cell ID APIs

For Centro GSM and CDMA smartphones, new APIs are available that will allow developers to retrieve the Cell ID (also known as the Cell of Origin, or COO) of a phone. For details, refer to **Section 8.1.4 on page 103**.

PART I

Introduction

This part of the guide details the overview of the Developer Guide, discusses the steps required to get started developing applications for the Palm OS platform, and includes an overview of the Palm® product line.

This page intentionally blank

CHAPTER 1

1. Overview

This chapter discusses how this *Developer Guide* is organized, the typographical conventions used in this guide, where to find more documentation and resources, and introduces Palm developer marketing programs.

1.1 How this guide is organized

This guide contains an overview of products that use the Palm OS by ACCESS, including Palm® Centro™ and Treo™ smartphones, LifeDrive™ mobile managers, and Tungsten™, and Zire™ handhelds. This guide also includes hardware specifications for each device and a matrix that identifies the software features available on each device.

The libraries discussed in this guide are organized by general category, such as Multimedia Libraries and Application Libraries.

In addition to the discussion of the libraries, this guide contains debugging information for troubleshooting problems on Palm devices, as well as style conventions for how certain features should be used. It also includes coding examples and specific references to more information in the *Palm API Guide, Palm OS Platform*.

Specifically, the Developer Guide is organized in the following chapters:

- **Chapter 1** - The Overview chapter includes information on how this document is organized, typographical conventions used in this guide, where to find more documentation, and information on Palm developer programs.
- **Chapter 2** - The Quick Start Guide explains how to set up your development system in order to write software applications for Palm devices on the Palm OS platform by ACCESS, including ACCESS and Palm requirements.
- **Chapter 3** - The Product Overview chapter describes the differentiating software features of Treo smartphones, as well as other Palm mobile managers and handheld devices. It also describes the tools, interfaces, and libraries of the SDK, and hardware specifications by device.
- **Chapter 4** - This chapter describes the separate PIM SDK and how to use its database structures to include PIM features in your applications.
- **Chapter 5** - The Multimedia chapter describes multimedia features and formats used by device.
- **Chapter 6** - The Data Communications chapter includes information on the data communication features and APIs, including the NetPref library, NetMaster library, HTTP library, the NetServices API, and basic email best practices.

- **Chapter 7** - This chapter provides reference information on the HTML library, including its usage model, architecture, and features.
- **Chapter 8** - The Telephony chapter provides reference material for the Telephony APIs in the Palm OS SDK.
- **Chapter 9** - The SMS chapter describes the SMS library usage model.
- **Chapter 10** - The System Extensions chapter provides details about the system extension features and APIs available in the Palm OS SDK from Palm, Inc. These features are differentiations from the SDK from ACCESS.
- **Chapter 11** - The Applications chapter details the features and APIs available in some of the Palm applications.
- **Chapter 12** - The SDIO chapter provides information on writing Palm OS applications that interact with SDIO hardware.
- **Chapter 13** - The Debugging chapter details how to debug problems with Palm APIs using tools and utilities available for Palm smartphones.
- **Chapter 14** - The Style Guide chapter outlines guidelines to follow when developing applications that work with the Blazer web browser.
- **Chapter 15** - This chapter defines the interfaces and interactions of the Palm expansion Multi-connector, also known as the Athena connector, and its surrounding circuits and controlling software.
- **Chapter 16** - This chapter defines the interface and interactions of the headset jack and its surrounding circuits and controlling software.
- **Chapter 17** - This chapter explains where to find external hardware drawings for Palm devices and peripherals.
- **Appendix A** - The Sample Code Appendix includes a listing and description of each sample application available in the Palm SDK.

1.2 Typographical conventions

This guide uses the following typographical conventions:

- **API names** - API names and associated parameters are formatted in the `Courier` font.
- **Directory paths -** Directory paths are shown in *italics*.
- **Navigational Steps** - Navigational steps through web site menus or other GUI features of applications are formatted in **Bold** font with arrows (**>**) between steps.
- **Sample Code** - Sample Codes used in this guide as examples of using specific APIs are formatted in the `Courier` font and set apart from other text.
- **URLs -** URLs are in bold font and underlined, for example: **http://www.palm.com.**

1.3 Additional documentation and resources

1.3.1 Palm® Customer Service and Support

User Manuals, support information, and other documentation on Palm OS-based devices can be found at the Palm Customer Service and Support website, **http://www.palm.com/us/support/**.

1.3.2 PDN Knowledge Base

For a searchable collection of Frequently Asked Questions, visit the PDN website at **http://pdn.palm.com**. When you join the program and log in, click the Knowledge Base link on the navbar. Use the pull-down menus to sort for Palm OS platform-specific issues.

The PDN Knowledge Base is updated with new Answers on a monthly basis.

1.3.3 ACCESS® documentation

For information on Palm OS programming, ACCESS Co LTD (formerly PalmSource) provides the *Palm OS Programmer's Companion* and the *Palm OS Programmer's API Reference*. Join the ACCESS Developer Network, and find these documents at the following URL:

http://www.access-company.com/developers/documents/palmos/palmos.html

1.3.4 Expansion Parts Store

The Expansion Parts Store, hosted by Northstar Systems, offers PDN software and hardware developers development kits (including hardware drawings) for the creation of peripherals for Palm products. Supported devices currently include Treo 650 smartphones, Tungsten T5, and many other handheld and mobile manager devices. The Northstar site includes information on connectors, cables, modem housings, handheld plastics, as well as SDIO and Bluetooth development tools.

To find the link to the Expansion Parts Store, log into PDN, then use the side navigation menu to go to **develop > get devices > parts**.

1.3.5 Cable and connector drawings

ATL Technologies manufactures cable assemblies and connector systems used with Palm devices. At their website, ATL provides external hardware drawings in PDF format for peripherals including the USB sync cable, power cable, Multi-connector, and a charging cradle.

To find the link to ATL Technologies, log into PDN, then use the side navigation menu to go to **develop > get devices > parts**.

1.4 Palm® developer marketing programs

For more information on all Palm marketing and sales programs, visit the Palm Developer Network (PDN) website and select **Market** from the navbar.

1.4.1 Designed for Palm® Products Program

The Designed for Palm Products (DFPP) Logo Program is Palm's compatibility and logo program for mobile application developers. Developers participate in the program by submitting applications for compatibility testing done by a third party testing facility, and upon passing, receive use of the Palm logo and other exclusive marketing benefits.

For complete program details and benefits, go to PDN at **https://pdn.palm.com**. Then, from the navbar on the left side of the PDN home page, select **Market > Designed for Palm Products program**.

Here, you will find the Designed for Palm Product Program Datasheet and complete details on how to apply for and prepare your applications for the DFPP program.

1.4.2 Device loaner program

As part of our effort to support the growth and evolution of Palm products, Palm is pleased to provide developers with the opportunity to borrow Palm devices. PDN Program members can use these loaner devices to develop, troubleshoot, test, port, and market applications that leverage Palm technology. It's our goal to provide programs that encourage support of Palm technology by connecting developers to the tools they need.

For more information on the Device Loaner Program, go to PDN and from the navbar, select **Market > Device loaner program**.

1.4.3 Device discount program

Palm is happy to offer our PDN developers discounts on products through the Palm devices and handheld stores.

For current offers, go to PDN and from the navbar, select **Market > Device discount program**.

1.5 Submitting documentation feedback

In order to better support the Palm developer community, the Technical Publications team is interested in your feedback about this guide or other Palm Developer Network (PDN) documentation. To submit questions, comments, or documentation requests, send email to **Developer.Publications@palm.com**. Include the document title as well as specific section numbers, as necessary.

Please do not submit inquiries unrelated to documentation issues.

CHAPTER 2

2. Quick Start Guide

This chapter gives instructions on how to begin developing applications for devices running Palm® OS by ACCESS® Systems America (formerly PalmSource). It details the required components from ACCESS and Palm®, Inc. and how to set up your development environment. It also explains the contents of the SDK from Palm, Inc.

2.1 Introduction

With its focus on the mobile user, Palm OS was designed from the beginning for mobile computing, and provides a flexible, easy-to-use, and compatible development platform.

Leading the mobile computing revolution since the introduction of the first Palm Pilot in 1996, Palm OS by ACCESS provides the largest selection of mobile software in the world. The Palm OS platform gives users the ability to control all of their mobile information, communication, and entertainment needs, no matter where their lives take them.

Components from both ACCESS and Palm, Inc. make up the required tools for developers to begin writing applications for devices running Palm OS:

- The Garnet OS SDK from ACCESS (Palm OS 5.x) provides a generic platform for developing applications that run on the Palm OS platform.
- The Palm OS SDK from Palm, Inc. provides Palm device-specific differentiations to the generic Garnet OS SDK. To develop applications that run on specific Palm handhelds, mobile managers, and smartphones, you will need the Palm OS SDK from Palm, Inc.

NOTE: The Garnet OS SDK is required. Palm no longer supports the Cobalt OS SDK from ACCESS.

The following sections describe the steps required to begin developing applications for devices running Palm OS.

2.2 ACCESS® requirements

2.2.1 Join the ACCESS® Developer Program

To download the latest Core Garnet OS SDK from ACCESS Co. LTD. (formerly PalmSource), you will first need to register for the ACCESS Developer Network at the following URL:

http://www.access-company.com/developers/

2.2.2 Download the ACCESS® Palm OS® SDK

From the ACCESS Developer Network, you will be able to download tools to set up your development environment and the latest Core Garnet OS SDK. The Garnet OS SDK includes headers files, documentation, and samples necessary to begin application development, but does not include any device-specific differentiations from Palm, Inc. For the Palm, Inc. SDK, see **Section 2.4 on page 23**.

IMPORTANT: The Garnet OS SDK does not include a build environment, nor does it include any device differentiations from Palm, Inc.

Next, from the ACCESS Developer Network:

1. Download and install the Garnet OS SDK (68K) R3.
2. Download and install the latest version of the Garnet OS 68K API SDK, which is an update to the Garnet SDK.

For more information and getting started tips on the Palm OS platform, see **http://www.access-company.com/developers/start.html**.

2.3 Development environment

The next step is to set up your integrated development environment (IDE). To develop applications for devices running Palm OS, you will need to acquire one or more of the following tools:

- **Garnet OS Developer Suite -** The Garnet OS Developer Suite from ACCESS runs under Microsoft Windows and is based on Eclipse, so most of its code runs in the Java Virtual Machine (JVM). The Garnet OS Developer Suite allows developers to build both Protein applications (all ARM-native code) for Palm OS Cobalt, and 68K applications for all versions of the Palm OS currently shipping. Download the Palm OS Developer Suite from the ACCESS Developer Network:

  http://www.access-company.com/developers/

NOTE: The Garnet OS Development Suite includes the Garnet OS SDK.

- **PRC-Tools -** PRC-Tools is a freeware complete compiler tool for building Palm OS applications in C or C++. PRC-Tools includes GNU packages, GCC, binutils, GDB, and other post-linker tools. A link to PRC-Tool resources can be found at the ACCESS Developer Network.

- **CodeWarrior Development Studio for Palm OS Platform -** Metrowerks CodeWarrior, available for Windows and Mac OS, is a complete programming tool for the Palm OS platform. CodeWarrior is available at the following URL:

  http://www.freescale.com/webapp/sps/site/homepage.jsp?nodeId=012726

  When you have installed CodeWarrior, make sure that the Garnet OS SDK headers and libraries are installed at *C:\ProgramFiles\Metrowerks\CodeWarrior\Palm OS Support*. You may install the Palm OS SDK from Palm, Inc. in any directory, as long as the headers and libraries used in your application link to the correct location.

NOTE: To use CodeWarrior for Palm OS development, you will need to use the Palm OS Support version of the Garnet OS Developer Suite.

2.4 Palm® requirements

To develop applications specific to Palm handheld, mobile manager, and smartphone devices, you will need to download and install the Palm OS SDK from Palm, Inc. You will also need to link your applications to the headers and libraries provided in the Palm OS SDK, which includes all feature differentiations from the generic Garnet OS SDK from ACCESS.

2.4.1 Join the Palm® Developer Network (PDN)

The Palm Developer Network (PDN) from Palm, Inc., located at **http://pdn.palm.com**, is a website resource for mobile hardware and software developers creating prosumer, enterprise, and services solutions targeting Palm devices who want to increase their solution sales, profitability and brand value.

PDN is a comprehensive mobile solution community developer program that offers developers a broad set of services that help you build your business with Palm – quickly and easily. PDN was designed by developers for developers, and offers a comprehensive set of technical, marketing, and sales services designed to help mobile solutions developers move rapidly from concept to market with reduced cost and complexity.

For registered members of PDN, the website provides resources including:

- Palm SDKs
- Simulators/Emulators
- Debugging tools
- Documentation
- Knowledge Base of developers' frequently asked questions. (For more information, see **Section 1.3.2 on page 19**.)
- Developer community forums
- Newsletter
- Information on the Palm Device Loaner Program
- Information on the Designed for Palm Products (DFPP) Compatibility Logo Program. (For more information, see **Section 1.4.1 on page 20**.)

For complete details on PDN, visit **http://pdn.palm.com**.

2.4.2 Download the Palm OS® Platform SDK

The latest Palm OS Platform SDK from Palm, Inc. is available for download from PDN at **http://pdn.palm.com**.

IMPORTANT: Before using Palm, Inc.'s Palm OS SDK, you must download the current ACCESS® Palm OS SDK, which is discussed in **Section 2.2.2 on page 22**.

The Palm, Inc. Palm OS SDK includes:

- **Header Files -** Palm-specific header files are required in addition to the header files provided by the ACCESS SDK. Header files are generally updated with each product launch.
- **Documentation:**
  - **Palm Developer Guide, Palm OS Platform -** The *Palm Developer Guide, Palm OS Platform* (this document) is the comprehensive guide to software and hardware development for all Palm devices on the Palm OS platform. The *Developer Guide* is generally updated for each major product launch.
  - **Palm API Guide, Palm OS Platform -** In the past, detailed documentation on API functions and structures was included in the *Developer Guide*. Complete API documentation is now generated directly from Palm source code and presented in a separate document. Refer to the *Palm API Guide, Palm OS Platform* in compressed HTML format (.chm) on the PDN as part of the SDK download.The *API Guide* is updated for each SDK build.
  - **Application Notes -** When relevant, Application Notes on specific developer issues are generated between SDK releases and *Developer Guide* revisions. Therefore, they are generally the most up to date source of information.
  - **Training Presentations -** Training presentations by Palm engineers are posted on PDN on topics such as new devices and debugging.
- **Sample Code -** The Sample Code section of the Palm OS Platform SDK is used to illustrate concepts and API usage, allowing developers to grasp basic and complicated ideas quickly. For more information about Sample Code, see **Appendix A**.
- **Debugging Tools and Utilities -** Debugging tools and utilities provided in the Palm OS SDK include:
  - PalmDebugger
  - PPP Tracer
  - TraceViewer
  - MemoryInfo
  - DebugPrefs.c.

For more information on Debugging, see **Chapter 13**.

- **Simulators/Emulators -** A simulator is provided for each individual product at the Palm Developer Network (PDN) on that product's page. (Palm OS platform-

specific simulators are provided by ACCESS, but they do not include any of Palm's device differentiations.) Generally, simulators are not updated, but new simulators are made available with each maintenance release.

The Palm OS Platform SDK is updated with every product launch or announcement. The SDK is versioned using the following X.Y.Z method:

- X is incremented if a major change/addition occurs.
- Y is incremented when a new product is released or announced.
- Z is incremented when a change is made to the SDK that does not warrant an X or a Y change.

This page intentionally blank

CHAPTER 3

3. Product Line Overview

This chapter provides an overview of Palm® smartphones, Tungsten™ handhelds, Zire™ handhelds, and the LifeDrive™ mobile manager, as well as a high-level description of the features available in each product line. A hardware features table illustrates the hardware and electrical specifications of each model, and a software features table illustrates the specific features and APIs that apply to each device.

3.1 Centro™ smartphones

The Centro smartphone by Palm is an entry-level compact smartphone that integrates a mobile phone, SMS and MMS messaging, web browsing, and a Palm organizer. Centro is the smallest Palm form factor to date.

Centro smartphones are available in two radio versions, GSM and CDMA, with the following specifications:

1. **GSM** - Centro smartphones feature a quad-band GSM/GPRS/EDGE world radio on the following frequencies:
   - 850MHz (NA band)
   - 900MHz (EU/Asia band)
   - 1800MHz (EU/Asia band)
   - 1900MHz (NA band)
2. **CDMA** - Centro smartphones feature a dual-band CDMA/1xRTT/EvDO nationwide radio on the following frequencies:
   - 800MHz (Cellular band)
   - 1900MHz (PCS band)

One of the key differentiators of the Centro smartphone is the integration of the main applications and the user interface, which makes applications easy to use.

Software applications included vary by carrier. Additional applications may be available by carrier and via Over the Air (OTA) download.

The Centro smartphone includes support for microSD cards up to 4GB (no SDIO support).

For a complete list of Centro smartphone hardware specifications, see **Section 3.6.1 on page 34**.

For a complete list of Palm software libraries compatible with Centro smartphones, see **Section 3.8.1 on page 40**.

3.1.1 What's not supported by Centro™ smartphones

Centro smartphones use Palm OS® version 5.4.9 by ACCESS. However, as each licensee can choose to implement only certain features of the operating system as it applies to its product, Palm implements certain features and not others.

Centro smartphones do not support:

- INet library

  Palm has never supported the INet library, contained in the header file `INetMgr.h`, or ported it to Palm products.

- `Lz77Mgr.h` header file
- `SmsLib.h` header file

  Centro smartphones have their own SMS library. The Centro SMS library supports the Exchange Manager.

- Telephony Manager

  Centro smartphones have their own Phone library. The telephony header files `TelephonyMgr.h`, `TelephonyMgrTypes.h`, and `TelephonyMgrUI.h` are not supported for Centro smartphones.

- Fax services are not supported by Centro smartphones.

Centro smartphones do not include the zLib that was included in the Treo 600 smartphone. A version of this library can be found at: **http://www.copera.com/zlib-armlet/**.

3.2 Treo™ smartphone product line

The Treo by Palm is a family of compact smartphones that integrates a mobile phone, wireless data applications such as messaging and web browsing, and a Palm OS® organizer.

Treo smartphones are available in two radio versions, GSM and CDMA, with the following specifications:

1. **GSM** - Treo 600, Treo 650, and Treo 680 smartphones feature a quad-band GSM/GPRS/EDGE world radio on the following frequencies:
   - 850MHz (NA band)
   - 900MHz (EU/Asia band)
   - 1800MHz (EU/Asia band)
   - 1900MHz (NA band)
2. **CDMA**
   - Treo 600 and Treo 650 smartphones feature a dual-band CDMA/1xRTT nationwide radio on the following frequencies:
     - 800MHz (Cellular band)
     - 1900MHz (PCS band)
   - Treo 700p and Treo 755p smartphones feature a dual-band CDMA2000/1xRTT/EvDO nationwide radio on the following frequencies:
     - 800MHz (Cellular band)
     - 1900MHz (PCS band)

One of the key differentiators of the Treo smartphone is the integration of the main applications and the user interface, which makes applications easy to use. Most applications are common to both versions of the Treo smartphone, GSM and CDMA, and they include:

- Phone application
- SMS messaging
  - GSM - SMS messages can be received and sent
  - CDMA - SMS services may or may not be offered and supported by the carrier
- MMS
  - GSM - All GSM phones include MMS
  - CDMA - Only Sprint and Verizon phones include MMS - Rest of World (ROW) release phones do not include MMS
- Proxyless Blazer® web browser supporting direct download of ring tones, applications, and documents
- Photo capture application
- Email applications are available, but the type of application varies by carrier
- Palm OS® organizer applications, such as Calendar, Contacts, Tasks, and Memos
- Music playback capability on Treo 650, Treo 680, Treo 700p, and Treo 755p smartphones

- Streaming Audio and Video applications on Treo 680, Treo 700p, and Treo 755p smartphones
- Bluetooth 1.2 for Treo 700p ROW and Treo 755p smartphones, which resolves car-kit compatibility issues that caused the Bluetooth (ACL) connection between the Treo 700p and Treo 755p smartphone and the car-kit to drop.

NOTE: Some applications are applicable only to the GSM or CDMA version. Treo smartphones might also be configured differently depending on the carrier.

For a complete list of Treo smartphone hardware specifications by device, see **Section 3.6.1 on page 34**.

For a complete list of Palm software libraries compatible with Palm OS-based Treo smartphones, see **Section 3.8.1 on page 40**.

3.2.1 What's not supported by Treo™ smartphones

Treo smartphones use Palm OS® version 5.x by ACCESS. However, as each licensee can choose to implement only certain features of the operating system as it applies to its product, Palm implements certain features and not others.

Treo smartphones do not support:

- INet library
- Palm has never supported the INet library, contained in the header file `INetMgr.h`, or ported it to Palm products.
- `Lz77Mgr.h` header file
- `SmsLib.h` header file
- Treo smartphones have their own SMS library. The Treo SMS library supports the Exchange Manager.
- Telephony Manager
- Treo smartphones have their own Phone library. The telephony header files `TelephonyMgr.h`, `TelephonyMgrTypes.h`, and `TelephonyMgrUI.h` are not supported in Treo smartphones.
- Fax services are not supported by Treo smartphones.
- Treo 650, Treo 680, Treo 700p, and Treo 755p smartphones do not include the zLib that was included in the Treo 600 smartphone. A version of this library can be found at: **http://www.copera.com/zlib-armlet/**.

3.3 Tungsten™ handheld product line

Devices in the Tungsten product line are designed to target the power business user. Tungsten handhelds provide easy, reliable access to business data, as well as seamless integration with the desktop business environment. Tungsten handhelds provide a large, easy-to-read display, compatibility with the most popular business applications, a large amount of storage (which is not lost when battery power is depleted), and powerful organization and search functions. Some models of Tungsten handhelds also enable the user to mount the handhelds as a drive on a compatible PC.

For a complete list of Tungsten hardware specifications by device, see **Section 3.6.2 on page 36**.

For a complete list of Palm libraries compatible with Tungsten handhelds, see **Section 3.8.2 on page 42**.

3.3.1 What's not supported by Tungsten™ handhelds

Tungsten handhelds use Palm OS version 5.x by ACCESS. However, as each licensee can choose to implement only certain features of the operating system as it applies to its product, Palm implements certain features and not others.

Tungsten handhelds do not support:

- INet library

  Palm has never supported or ported the INet library, contained in the header file `INetMgr.h`, to its products.
- `Lz77Mgr.h` header file

3.4 LifeDrive™ mobile manager

The LifeDrive mobile manager lets users keep track of schedules, business and personal contacts, to-do lists, and even Microsoft Office and multimedia files. LifeDrive mobile manager also offers a 4GB hard drive that lets users carry files and hours of music, photos, and videos. Users can transfer information in real time between the device and their computer and, on a Windows computer, select which files and folders to synchronize. Users can import photos and videos from a digital camera's memory card, or connect wirelessly to a Wi-Fi network.

LifeDrive mobile manager is the first Palm device to use a hard drive for memory (storage). LifeDrive mobile manager's hard drive changes some of the basic assumptions that Palm OS applications typically make about the speed of various operations. Not only does it use DBCache (introduced in NVFS devices), but developers should also consider the following performance issues:

- If the drive is automatically turned off (after ten seconds of inactivity, meaning no active writes or reads to/from the drive), the system stalls for one or two seconds while the drive spins up again.
- Whenever the heads of the drive are over the platters, the hard drive is susceptible to damage from drops. The heads are over the platters during every read and write operation and remain there until there have been about two seconds of no read/write activity.

For a complete list of LifeDrive hardware specifications by device, see **Section 3.6.2 on page 36**.

For a complete list of Palm libraries compatible with LifeDrive mobile managers, see **Section 3.8.2 on page 42**.

3.4.1 What's not supported by LifeDrive™ mobile managers

LifeDrive mobile managers use Palm OS version 5.x by ACCESS. However, as each licensee can choose to implement only certain features of the operating system as it applies to its product, Palm implements certain features and not others.

LifeDrive handhelds do not support:

- INet library

  Palm has never supported or ported the INet library, contained in the header file `INetMgr.h`, to its products.

- `Lz77Mgr.h` header file

3.5 Zire™ handheld product line

Devices in the Zire product line are designed to target the consumer who wants an easy-to-use handheld that is ready out of the box and useful in both a personal and business environment. Zire handhelds leverage MP3 capability, a color screen, and expandibility to appeal to casual technology users interested in value, as well as savvy young technology users interested in style and the latest functionality.

For a complete list of Zire hardware specifications by device, see **Section 3.6.3 on page 37**.

For a complete list of Palm software libraries compatible with Zire handhelds, see **Section 3.8 on page 40**.

3.5.1 What's not supported by Zire™ handhelds

Zire handhelds use Palm OS version 5.x by ACCESS. However, as each licensee can choose to implement only certain features of the operating system as it applies to its product, Palm implements certain features and not others.

Zire handhelds do not support:

- INet library

  Palm has never supported or ported the INet library, contained in the header file `INetMgr.h`, to its products.

- `Lz77Mgr.h` header file

3.6 Hardware feature matrix

3.6.1 Palm® smartphone hardware features

Feature	Treo 600	Treo 650	Treo 680	Treo 700p	Treo 755p	Centro
Processor						
Type	TI OMAP 310 ARM	Intel XScale PXA270 ARM	Intel XScale PXA270 ARM	Intel XScale PXA270 ARM	Intel XScale PXA270 ARM	Intel Bulverde ARM
Speed	144MHz	312MHz	312MHz	312MHz	312MHz	312Mhz
Memory						
RAM	32MB	32MB	64MB	32MB	32MB	64MB
NAND Flash User Store	N/A	32MB	64MB	64MB	64MB	64MB
Battery						
Type	Rechargeable Lithium Ion	Rechargeable Lithium Ion	Rechargeable Lithium Ion	Rechargeable Lithium Ion	Rechargeable Lithium Ion	Rechargeable Lithium Ion
mAh	1800	1900	1200	1800	1600	1150
Standby or use time	up to 300 hours	GSM/GPRS - up to 300 hours CDMA - up to 336 hours	up to 300 hours	up to 300 hours	up to 300 hours	GSM - up to 300 hours CDMA - up to 250 hours
Talk time	up to 6 hours	up to 6 hours	up to 4 hours	up to 4.5 hours	up to 4.5 hours	GSM - up to 4 hours CDMA - up to 3.3 hours
Removable	No	Yes	Yes	Yes	Yes	Yes
Form factor						
Size	4.41" x 2.36" x .87" without antenna	4.41" x 2.36" x .87" without antenna	4.4" x 2.3" x .8" internal antenna	4.4" x 2.3" x .9" without antenna	4.4″ x 2.3″ x .8" internal antenna	4.2″ x 2.1″ x .73″internal antenna
5-way button	Yes	Yes	Yes	Yes	Yes	Yes
Grafitti	None	None	None	None	None	None
Keyboard	Built-in QWERTY keyboard	Built-in QWERTY keyboard	Built-in QWERTY keyboard	Built-in QWERTY keyboard	Built-in QWERTY keyboard	Built-in QWERTY keyboard

Feature	Treo 600	Treo 650	Treo 680	Treo 700p	Treo 755p	Centro
Display						
Resolution	160 x 160 pixels	320 x 320 pixels, 186 DPI	320 x 320 pixels, 186 DPI	320 x 320 pixels, 186 DPI	320 x 320 pixels, 186 DPI	320 x 320 pixels 186 DPI
Density	16-bit (65,536 colors)	16-bit (65,536 colors)	16-bit (65,536 colors)	16-bit (65,536 colors)	16-bit (65,536 colors)	16-bit (65,536 colors)
Wireless						
	CDMA or GSM/GPRS	CDMA/1xRTT or GSM/GPRS/EDGE, Bluetooth® 1.1	Quad-band GSM/ GPRS/EDGE (850/ 900/1800/1900) Bluetooth® 1.2	CDMA/1xRTT/EvDO Bluetooth® - Sprint and Verizon, Bluetooth® 1.2 - Rest of World*, Bluetooth® 1.2 with patch	CDMA/1xRTT/EvDO Bluetooth® 1.2 with patch	CDMA/1xRTT/EvDO or GSM/GPRS/EDGE, Bluetooth® 1.2 with patch
Camera						
	VGA 640 x 480 0.5 mega-pixels	VGA 640 x 480 0.5 mega-pixels	VGA 640 x 480 0.5 mega-pixels	1280 x 1024 1.3 mega-pixels	1280 x 1024 1.3 mega-pixels	1280 x 1024 1.3 mega-pixels
Interface Connector						
	Treo 600 smartphone connector (USB, serial without flow control)	Multi-connector (USB, serial without flow control)	Multi-connector (USB, serial without flow control)	Multi-connector (USB, serial without flow control)	Multi-connector (USB, serial without flow control)	Multi-connector (USB, serial without flow control)
Audio						
	2.5mm headset jack, stereo compatible	2.5mm headset jack, stereo compatible	2.5mm headset jack, stereo compatible	2.5mm headset jack, stereo compatible	2.5mm headset jack, stereo compatible	2.5mm headset jack, stereo compatible
	speakerphone	speakerphone	speakerphone	speakerphone	speakerphone	speakerphone

* For the current list of carriers included in the Treo 700p ROW release, visit the Palm Developer Network (PDN) at **http://pdn.palm.com** and navigate to the Treo 700p device page.

3.6.2 Palm® handheld and LifeDrive™ mobile manager hardware features

Feature	Palm T|X handheld	LifeDrive mobile manager	Tungsten T5	Tungsten E	Tungsten E2	Tungsten C
Processor						
Type	Intel Xscale PXA270 ARM	Intel Xscale PXA270 ARM	Intel XScale PXA270 ARM	TI OMAP 311 ARM	Intel XScale PXA255 ARM	Intel XScale PXA255 ARM
Speed	312MHz	416MHz	312MHz	126MHz	200MHz	400MHz
Memory						
RAM	32MB SDRAM + 128MB NAND Flash	32MB + 3.85GB hard drive	256MB	32MB	32MB	64MB
ROM	12MB (masked)	16MB	32MB	8MB	8MB (masked)	16MB
Battery						
Type	Rechargeable Lithium Ion	Rechargeable Lithium Ion	Rechargeable Lithium Ion	Rechargeable Lithium Ion	Rechargeable Lithium Ion	Rechargeable Lithium Ion
mAh	1300	1660	1020	800	1020	1500
Standby or use time	N/A	N/A	48 days	21 days	48 days	30 days
Talk time	N/A	N/A	N/A	N/A	N/A	N/A
Removable	No	No	No	No	No	No
Form factor						
Size	4.76" x 3.08" x .61" without flipcover	4.76" x 2.87" x 7.4"	4.5" x 3" x .5" without flipcover	4.5" x 3.1" x .5"	4.5" x 3.1" x .59" without flipcover	4.0" x 3.07" x .65"
5-way button	Yes	Yes	Yes	Yes	Yes	Yes
Grafitti	Dynamic	Dynamic	Dynamic	Yes	Yes	None
Keyboard	None	None	None	None	None	Built-in QWERTY keyboard
Display						
Resolution	320 x 480	320 x 480	320 x 480 pixels	320 x 320 pixels	320 x 320 pixels	320 x 320 pixels
Density	16-bit (65,536 colors)	16-bit (65,536 colors)	16-bit (65,536 colors)	16-bit (65,536 colors)	16-bit (65,536 colors)	16-bit (65,536 colors)
Wireless						
	802.11b (WPA2 enabled) Bluetooth®	Wi-Fi (802.11b), Bluetooth® 1.1	Bluetooth®	None	Bluetooth®	Wi-Fi (802.11)

Feature	Palm T|X handheld	LifeDrive mobile manager	Tungsten T5	Tungsten E	Tungsten E2	Tungsten C
Camera						
	None	None	None	None	None	None
Interface Connector						
	Multi-connector (USB, serial without flow control)	Multi-connector (USB, serial without flow control)	Multi-connector (USB, serial without flow control)	Standard Mini-USB	Multi-connector (serial without flow control)	Universal Connector (USB, serial with flow control)

3.6.3 Palm® Z22 organizer and Zire™ handhelds hardware features

Feature	Palm Z22	Zire 31	Zire 72
Processor			
Type	Samsung S3C2410A ARM	Intel XScale PXA255 ARM	Intel XScale PXA270 ARM
Speed	200MHz	200MHz	312MHz
Memory			
RAM	16MB SDRAM + 32MB NAND Flash	16MB	32MB
ROM	6MB ROM	4MB	8MB
Battery			
Type	Rechargeable Lithium Ion	Rechargeable Lithium Ion	Recharge-able Lithium Ion
mAh	720	900	950
Standby or use time	14 days	48 days	29 days
Talk time	N/A	N/A	N/A
Removable	No	No	No
Form factor			
Size	2.7" x 4.06" x 0.6"	4.4" x 2.9" x .6"	4.6" x 2.95" x .67"
5-way button	Yes	Yes	Yes
Grafitti	Yes	Yes	Yes
Keyboard	None	None	None
Display			
Resolution	160x160 pixels	160x160 pixels	320 x 320 pixels
Density	16-bit (65,536 colors) 12-bit actual	16-bit (65,536 colors)	16-bit (65,536 colors)
Wireless			

Feature	Palm Z22	Zire 31	Zire 72
	None	None	Bluetooth®
Camera			
	None	None	Photo: 1280 x 960 Video: 320 x 240
Interface Connector			
	Standard Mini-USB	Standard Mini-USB	Standard Mini-USB

3.7 SDIO support

An SDIO (SD Input/Output) card extends the functionality of devices with SD card slots. SDIO cards support different data transfer modes, including SPI, SD 1-bit, and SD 4-bit modes.

Many SDIO cards are labelled "low speed" or "full speed."

- If an SDIO card is a low-speed card, then it is required to support only the SPI and 1-bit data transfer modes.
- If an SDIO card supports full-speed data transfer, then all three data modes (SPI, SD 1-bit, and SD 4-bit) are supported.

The following table* lists the transfer modes of SDIO cards that are compatible with Palm devices.

Device	SPI	SD 1-bit	SD 4-bit
Centro smartphones	-	-	-
LifeDrive mobile manager	-	●	●
Palm T|X handheld	-	●	●
Palm Z22 organizer	-	-	-
Treo 600 smartphone	-	●	-
Treo 650 smartphone	-	●	●
Treo 680 smartphone	-	●	●
Treo 700p smartphone	-	●	●
Treo 755p smartphone	-	-	-
Tungsten C handheld	●	●	-
Tungsten E handheld	●	●	●
Tungsten E2 handheld	-	●	-
Tungsten T5 handheld	-	●	●
Zire handheld	-	-	-
Zire 31 handheld	-	●	-
Zire 71 handheld	●	●	●
Zire 72 handheld	●	●	●

For information on developing SDIO applications for Palm devices, see **Chapter 12**.

* In this and the following tables, a bullet (●) indicates that the feature is available on that device.

3.8 Software compatibility specifications (Palm® libraries)

3.8.1 Palm® smartphones

Software	Treo 600	Treo 650	Treo 680	Treo 700p	Treo 755p	Centro
Palm OS	5.2.1	5.4	5.4.9	5.4.9	5.4.9	5.4.9
Multimedia						
Ring Tones	●	●	●	●	●	●
Sound and Voice Recording	●	●	●	●	●	●
Camera	●	●	●	●	●	●
Photos	●	●	●	●	●	●
Video	-	-	●	●	●	●
Codec Plug-in Manager	-	●	●	●	●	●
Data communications						
Network preferences	●	●	●	●	●	●
HTTP	●	●	●	●	●	●
Wi-Fi	-	-	-	-	-	-
NetMaster	●	●	●	●	●	●
HTML Library	-	●	●	●	●	●
Telephony						
Telephony	●	●	●	●	●	●
SMS	●	●	●	●	●	●

Software	Treo 600	Treo 650	Treo 680	Treo 700p	Treo 755p	Centro
System extensions						
Transparency	●	●	●	●	●	●
File Browser API	-	-	●	-	-	-
Smart Text Engine	●	●	●	●	●	●
REM sleep	●	●	●	●	●	●
MMS helper functions	●	●	●	●	●	●
NVFS	-	●	●	●	●	●
Handspring® extensions	●	●	●	●	●	●
Keyguard	●	●	●	●	●	●
Option and shift	●	●	●	●	●	●
Tips and tutorial	●	●	●	●	●	●
Full-screen writing	-	-	-	-	-	-
Applications						
Web browser	●	●	●	●	●	●
VersaMail®	-	●	●	●	●	●

3.8.2 Palm® handhelds, organizers, and mobile managers

Software	Palm T|X handheld	Palm Z22 organizer	LifeDrive mobile manager	Tungsten T5	Tungsten E	Tungsten E2	Tungsten C	Zire 31	Zire 72
Palm OS	5.4.9	5.4.9	5.4.8	5.4	5.2.1	5.4.7	5.2.1	5.2.8	5.2.8
Multimedia									
Ring Tones	-	-	-	-	-	-	-	-	-
Sound and Voice Recording	●	-	●	●	-	-	●	-	●
Camera	-	-	-	-	-	-	-	-	●
Photos	●	●	●	●	●	●	-	●	●
Video	●	-	●	●	-	-	-	-	●
Codec Plug-in Manager	●	-	●	●	-	●	-	-	●
Data communications									
Network preferences	-	-	-	-	-	-	-	-	-
HTTP	●	-	●	●	-	●	-	-	-
Wi-Fi	●	-	●	-	-	-	●	-	-
NetMaster	-	-	-	-	-	-	-	-	-
HTML Library	●	-	●	-	-	-	-	-	-
Telephony									
Telephony	-	-	-	-	-	-	-	-	-

Software	Palm T|X handheld	Palm Z22 organizer	LifeDrive mobile manager	Tungsten T5	Tungsten E	Tungsten E2	Tungsten C	Zire 31	Zire 72
SMS	-	-	-	-	-	-	-	-	-
System extensions									
Transparency	-	-	-	-	-	-	-	-	-
File Browser API	-	-	●	●	-	-	-	-	-
Smart Text Engine	-	-	●	-	-	-	-	-	-
REM sleep	●	-	-	-	-	-	-	-	-
MMS helper functions	-	-	-	-	-	-	-	-	-
NVFS	●	●	●	●	-	●	-	-	-
Handspring® extensions	-	-	-	-	-	-	-	-	-
Keyguard	-	-	-	-	-	-	-	-	-
Option and shift	-	-	-	-	-	-	-	-	-
Tips and tutorial	●	●	●	●	-	●	-	-	-
Full-screen writing	●	●	●	●	-	●	-	●	●
Applications									
Web browser	●	-	●	●	-	●	-	-	-
VersaMail®	●	-	●	●	-	●	-	-	-

3.9 sysExternalConnectorAttachEvent and sysExternalConnectorDetachEvent notifications

The `sysExternalConnectorAttachEvent` and `sysExternalConnectorDetatchEvent` notifications are used with the Palm Multi-connector. For more information, see **Chapter 15**.

	Power notification 0x0008	Serial notification 0x0040	USB notification 0x0010
Tungsten T5 handheld	●	●	●
Tungsten E2 handheld	●	●	●
LifeDrive mobile manager	●	●	●
Treo 650 smartphone	-	-	-
Treo 680 smartphone	●	●	●
Treo 700p smartphone	●	●	●
Treo 755p smartphone	●	●	●
Centro smartphone	●	●	●

3.10 kPmConnectorClass Notifications

The `kPmConnectorClass` notifications are used by LifeDrive mobile managers to detect accessories.

	Audio With Headset 0x05	Audio No headset 0x00	Carkit 0x01	Generic Serial 0x03
LifeDrive mobile manager	-	●	●	●

PART II

Features and Libraries

This part of the guide details the software libraries available in the Palm OS® SDK.

This page intentionally blank

CHAPTER 4

4. PIM SDK

This chapter incudes information on the PIM SDK, and how to use its database structures for PIM features.

4.1 The PIM SDK

The PIM SDK includes the latest information on PIM databases and structures, including header files, documentation, and sample code. It must be downloaded separately from the latest Palm OS SDK at **http://pdn.palm.com**.

To find the PIM SDK, navigate to **develop > platform SDKs > Palm OS**. The link to the PIM SDK is included on the Palm OS platform page, which is called, "Updated PIM Database Structures." Alternately, you can search the PDN Knowledge Base for **Answer ID 418**.

For complete documentation see the document *Application Note 4: Accessing PIM Databases* in the PIM SDK.

4.2 Known issue

Palm PIM applications include 4 features:

- Contacts
- Calendar
- Tasks
- Memo

Palm provides two methods for developing applications using Contacts. One is to call the Contacts library APIs listed in `Contacts.h`. The second method is to use the APIs provided in the PIM SDK to access the PIM database directly. Please be aware that using the Contacts library APIs listed in `Contacts.h` may not provide satisfactory performance.

Palm provides only one method for developing applications using the other PIM features: Calendar, Tasks, and Memo. This method is to use the APIs provided in the PIM SDK to access the PIM database directly.

When different applications use both methods simultaneously, there may be conflicts. To prevent conflicts, when you call APIs provided in the PIM SDK to access the PIM database, pass the `dmModeExclusive` argument.

If you have used the `dmModeExclusive` argument and still have database conflicts, contact Palm DTS.

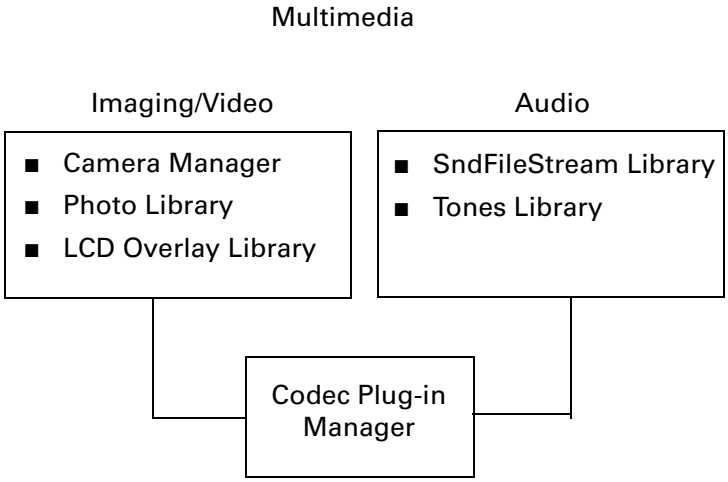
This page intentionally blank

CHAPTER 5

5. Multimedia

This chapter details the multimedia features and libraries available in the Palm OS® SDK. Palm OS devices feature imaging, video, and audio multimedia capabilities.

This chapter begins with a detailed description of the Codec Plug-in Manager, which is used to register and provide codecs for multimedia features where necessary. Other components used to provide each feature are illustrated in the diagram below.



5.1 Codec Plug-in Manager

Available on:

- Centro™ and Treo™ smartphones
- LifeDrive™ mobile manager
- Tungsten™ T5, Tungsten™ E2, and Palm® T|X handhelds
- Zire™ 72 handhelds

The Codec Plug-in Manager unifies codecs available for Palm devices and provides a standardized way to access and load codecs.

The Codec Plug-in Manager is declared in the header file `palmOnePalmCodecPluginMgr.h`, while the method that codecs use to handle individual formats is declared in the header file `palmOneCodecFormat.h`. Each media format is denoted by a four-byte string (for example, `JPEG`).

Because there are many separate formats handled by the Codec Plug-in Manager, and new formats may be added later as new codecs are written or released, the particulars of handling specific formats are beyond the scope of this chapter. A complete list of the codec media formats currently available exists in the `palmOneCodecFormat.h` section of the *Palm API Guide, Palm OS Platform*.

For examples of how to handle specific formats, refer to the Sample Code section of the Palm OS SDK.

5.1.1 Codec Plug-in Manager overview

At each bootup, the Codec Plug-in Manager searches for codec plug-ins first in RAM, then in the system image. It recognizes codecs by the PRC type `'CdPl'` and loads all codecs into memory, regardless of whether they will be used or not.

Each PRC of type `'CdPl'` is then queried for the format pair it supports. A format pair consists of two parts:

1. The input format
2. The output format that will come out of the codec

For example, an MPEG4 codec (decoder) supports MPEG4 ->YUV, so its format pair is [MPEG4, YUV]. The expected input is MPEG4, and the output format is YUV.

A single PRC can contain multiple codecs and support multiple format pairs.

IMPORTANT: Major codecs such as **JPEG**, **GIF**, or **MP3** should each be implemented as single PRCs.

For each codec plug-in, there can be more than one codec. These codecs may also have the exact same input and output formats. For this reason, every format pair has an additional parameter associated with it, called a Codec ID. This codec ID parameter is used to differentiate between the different codecs in a codec plug-in. In most cases, one codec plug-in only contains one codec, so you can set the Codec ID parameter to 0 `(palmNULLCodecID)`.

For more information on Codec IDs, see **Section 5.1.3 on page 53**.

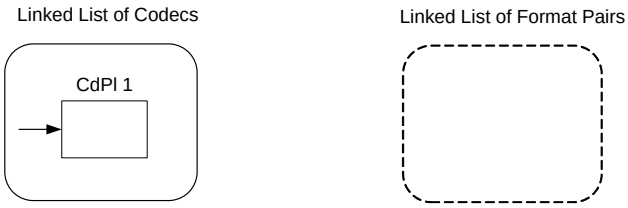
The Codec Plug-in Manager creates and manages two linked lists:

1. **A linked list of codecs -** Includes codec plug-ins, or `'CdPl'`s, where each codec is added to the front of this list upon discovery.
2. **A linked list of format pairs -** Includes format pairs, where each pair is added to the beginning of this list upon discovery, and each format pair has a reference to a codec plug-in in the linked list of codecs

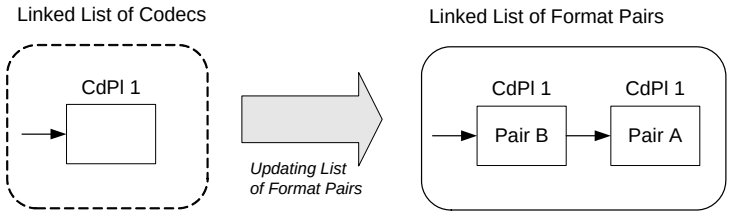
The following diagram is an example of the process.

NOTE: In this diagram, items with solid lines indicate which of the two linked lists is being acted upon at the time of the change. Items with dashed lines indicate that this list is *not* being acted upon.

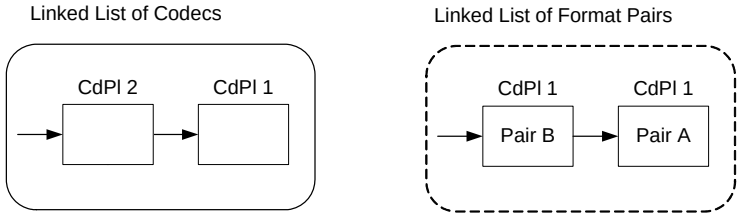
1. At bootup, the Codec Plug-in Manager looks for codecs, and finds first codec plug-in, `'CdPl' 1`:



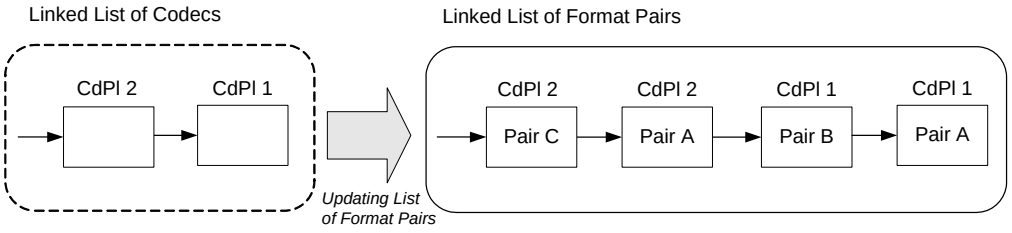
2. The Codec Plug-in Manager then queries `'CdPl' 1` for the format pairs it supports. In this example, `'CdPl' 1` supports two pairs, "Pair A" and "Pair B":



3. Next, the Codec Plug-in Manager finds another codec plug-in, `'CdPl' 2`. (Notice that `'CdPl' 2` is placed in front of `'CdPl' 1` in the linked list.)



4. The Codec Plug-in Manager then queries `'CdPl' 2` for the format pairs that it supports. `'CdPl' 2` supports two pairs, "Pair A" and "Pair C".



When the Codec Plug-in Manager receives a request for a codec plug-in that supports a particular format pair, the Codec Plug-in Manager will traverse the linked list of format pairs to see if the requested format pair is supported. The Codec Plug-in Manager will then return the codec that corresponds to the *first* format pair that matches what it's looking for. This means that a codec loaded later *always* has a higher priority than a codec loaded earlier.

As shown in the previous diagram example, when an application asks the Codec Plug-in Manager for a codec that supports the format pair "Pair A", the Codec Plug-in Manager will traverse the linked list of format pairs and find format pair "Pair A" as the second element, and return `'CdPl' 2`. It is important to note in this example that Codec Plug-in Manager will always return `'CdPl' 2`, because it was loaded last and is therefore the first one in the format pairs linked list, even though there is another codec that supports the format pair "Pair A" in the same list.

To override this codec priority, use a codec's Creator and Codec ID to create a codec session with a specific codec by calling `CodecMgrCreateSessionByID ()`.

5.1.2 Codec wrapping

For the Codec Plug-in Manager to recognize a codec, the codec must be "wrapped". This means that each codec must be type `'CdPl'`, and must have the correct structure for the format pairs it supports. A codec that is wrapped will be registered with as a codec that can be used by applications when calling the Codec Plug-in Manager.

If you are developing a codec plug-in for the Codec Plug-in Manager, you must follow the function numbers in the order specifically illustrated in the following table. Currently, only 12 functions are supported (0-11, as shown in the following table).

To use the function name, call it the same way you would call a function that is in a 68K shared library.

Function Number	Function
0	CodecOpen
1	CodecClose
2	CodecSleep
3	CodecWake
4	CodecLibAPIVersion
5	CodecEnumerateSupportedFormats
6	CodecCreateSession
7	CodecResetSession
8	CodecExitSession
9	CodecGetMaxDestBufferSize
10	CodecEncodeDecode
11	CodecCustomControl

5.1.3 Codec Plug-in Manager process

To use the Codec Plug-in Manager, an application must determine if the input/output format of the codec is supported. Optionally, it can also specify which particular codec to use.

The Codec Plug-in Manager selects a particular codec based on four criteria:

1. **Input format -** The format of the data input to the codec.
2. **Output format -** The format of the data output from the codec.
3. **PRC Creator ID -** The creator ID of the PRC containing the desired codec on the device. This is optional; you can have the Codec Plug-in Manager select an appropriate codec based on the input and output formats.

IMPORTANT: PRC Creator IDs are unique and must be registered with ACCESS (formerly PalmSource).

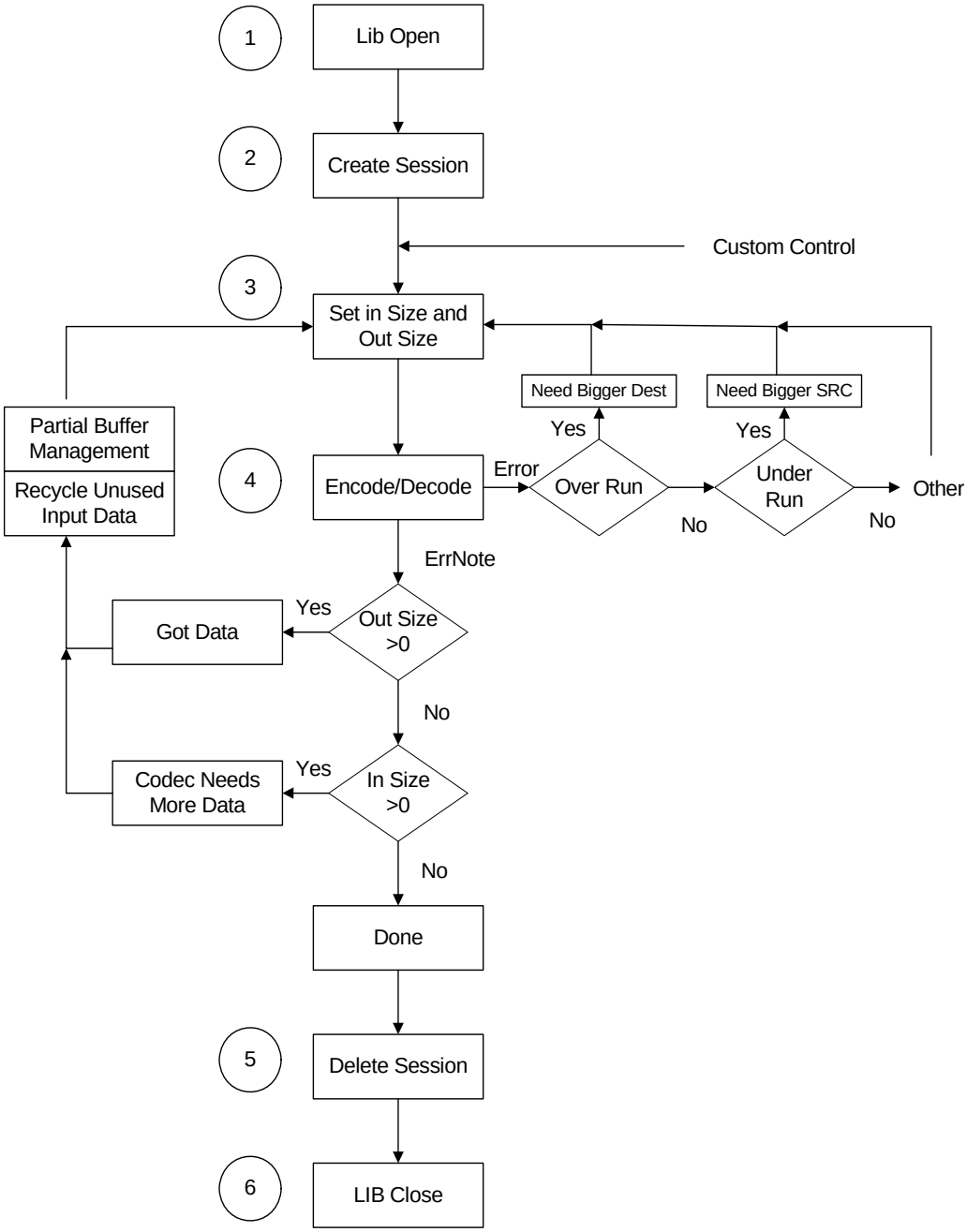
---

4. **Codec ID -** The unique ID of the codec. This is provided because a PRC file can contain multiple codecs that have the same input and output formats. Again, this is optional; you can have the Codec Plug-in Manager select an appropriate codec based on the input and output formats.

Once an application has selected a codec, it must go through the following steps, which are illustrated in the diagram on the next page:

1. Open the library.
2. Create a session with the Codec Plug-in Manager.
3. Specify the input and output formats and, optionally, which particular codecs to use.
4. Start decoding or encoding.
5. Delete the session once the decoding or encoding is complete.
6. Close the library.

The following diagram shows the process flow of a typical Codec Plug-in Manager session.



5.1.4 Media codec formats supported by device

The following table details the codecs that are supported for imaging, video, and audio features on Palm OS devices.

Feature	Technology				Product											
	Codec	Description	Encoder	Decoder	Zire		Tungsten				Treo					Centro
					22	72	T5	E2	T|X	Life Drive	6XX CDMA	6XX GSM	680 GSM	700p	755p	
Imaging	**BMP**	Windows bitmap	-	●	-	●	●	●	●	●	●	●	●	●	●	●
	GIF	Graphics interchange format	-	●	-	●	●	●	●	●	●	●	●	●	●	●
	JPEG	standard image compression algorithm	●	●	-	●	●	●	●	●	●	●	●	●	●	●
	TIFF	Still image bitmaps stored in tagged fields	-	●	-	●	●	●	●	●	●	●	●	●	●	●
Video	**H263**	Provisional ITU-T standard	●	●	-	●	●	●	●	●	●	●	●	●	●	●
	MJPEG	JPEG encoding for moving images	-	●	-	●	●	●	●	●	●	●	●	●	●	●
	MPEG4	Extension of MPEG1 and MPEG2 compression	-	●	-	●	●	●	●	●	●	●	●	●	●	●
			●		-	●	-	-	-	●	●	-	-	●	●	●

Feature	Technology				Product											
	Codec	Description	Encoder	Decoder	Zire		Tungsten				Treo					Centro
					22	72	T5	E2	T|X	Life Drive	6XX CDMA	6XX GSM	680 GSM	700p	755p	
Audio	ADPCM	ADPCM decoder	-	●	-	●	●	●	●	●	●	●	●	●	●	●
	AMR	AMR-NB	●	●	-	-	-	-	-	-	-	●	●	●	●	●
		3GPP AMR/AMR2/ GSM AMR	●	-	-	-	-	-	-	-	-	●	●	●	●	●
	MP3	Audio Level 3 (MP3)	-	●	-	●	●	●	●	●	●	●	●	●	●	●
	QCELP	QCELP 13K encoder library following the standard TIA/EIA IS733	●	●	-	-	-	-	-	-	●		●	●	●	●

5.1.5 For more information

The Codec Plug-in Manager header files include:

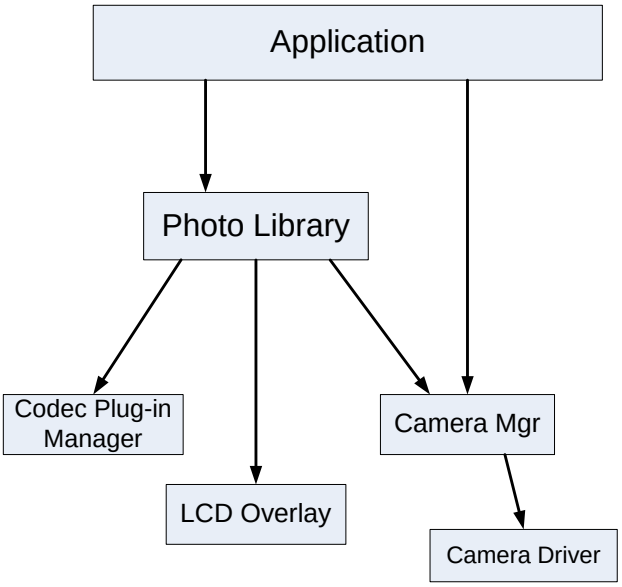
- `palmOneCodecPluginMgr.h`
- `palmOneCodecPluginMgrCommon.h`
- `palmOneCodecFormat.h`

For more detailed information on the Codec Plug-in Manager, refer to the *Palm API Guide, Palm OS Platform* at the Palm Developer Network (PDN).

Also, refer to the Sample Code section of the Palm OS SDK for coding examples.

5.2 Imaging

On Palm Treo™ smartphones and handhelds that include cameras, an application must work with several software components to provide imaging functionality, including the Photo Library, Camera Manager, Codec Plug-in Manager, LCD Overlay, and Camera Drivers, as shown in the following diagram.



5.2.1 Photo Library

Available on:

- Centro™ and Treo™ smartphones
- LifeDrive™ mobile manager
- Tungsten™ T5, Tungsten™ E, Tungsten™ E2, and Palm® T|X handhelds
- Zire™ 31 and Zire™ 32 handhelds
- Palm® Z22 organizer

The Photo Library includes three basic categories of functionality:

1. Capturing images and video with a camera attached to device
2. Storing images and video
3. Manipulating images

The Photo Library uses the LCD Overlay to preview what the camera is scanning. The Photo Library itself provides the camera user interface, but uses the Camera Manager to capture photos. For more information, see **Section 5.2.2 on page 60**.

When storing and manipulating images, the Photo Library provides the ability to select images, open, close and save images, get image information, scale, crop and rotate images, and delete images.

There are three versions of the Photo Library: version 1, version 2, and version 3. In general, structures, function names, and things that ends in "V2" are available only in version 2 and above. Items that ends in "V3" are available only in version 3. Exceptions are noted where appropriate.

Use `PalmPhotoLibGetVersion()` to check the version of the library that exists on the device.

The Photo Library is declared in the header file `palmOnePhoto.h`.

5.2.1.1 Coding examples

The following code is an example of how to use the Photo Library to capture images:

```
MemSet(&cParam, sizeof(PalmPhotoCaptureParamV2), 0);

//saving to memory (versus SD card, for example)
cParam.fileLocation.fileLocationType = palmPhotoMemoryLocation;

cParam.fileLocation.file.MemoryFile.bufferSize = 0;
cParam.fileLocation.file.MemoryFile.bufferP = NULL;
cParam.imageInfo.width = width;
cParam.imageInfo.height = height;
cParam.imageInfo.bitsPerPixel = 16;

//specifying RGB565 file format (versus JPEG, for example)
cParam.imageInfo.fileFormat = palmPhotoRGB565FileFormat;

//capturing picture
imageH = PalmPhotoCaptureImageV2(photoLibRefNum, &cParam );
```

The following code is an example of how to use the Photo Library to select images:

```
SelectionParam.albumID = PALM_PHOTO_ALBUM_ALL;
SelectionParam.offset = 0;
SelectionParam.selectionCount = 3;
SelectionParam.filterCallback = NULL;
SelectionParam.userDataP = NULL;
SelectionParam.selectedImages.imageCount= 0;
PalmPhotoSelectDlg(photoLibRef, &SelectionParam, palmPhotoDlgSelection, true);
```

For more coding examples, refer to the Sample Code section of the Palm OS SDK.

5.2.1.2 For more information

The Photo Library header files include:

- `PalmPhoto.h`
- `palmOnePhoto.h`
- `palmOnePhotoCommon.h`

Photo Utility header files include:

- `PalmPhotoUtilProt.h`
- `PalmPhotoUtilTypesProt.h`

For more detailed information on the Photo Library and Photo Utility header files, refer to the *Palm API Guide, Palm OS Platform* at the Palm Developer Network (PDN).

Also, refer to the Sample Code section of the Palm OS SDK for coding examples.

5.2.2 LCD Overlay

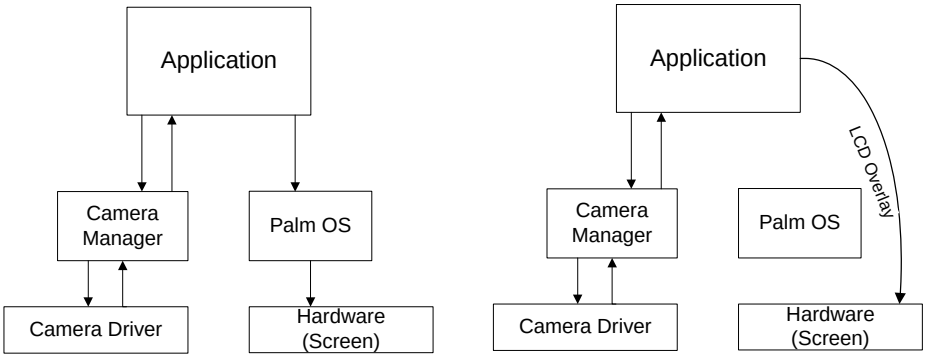
Available on:

- Centro™ and Treo™ smartphones
- Zire™ 72 handhelds

On devices with a camera, the LCD Overlay allows users to preview what the camera is scanning. The LCD Overlay is declared in the header files `palmOneLcdOverlay.h` and `palmOneLcdOverlayCommon.h`.

Without the LCD Overlay, applications would have to negotiate the Palm OS software layer to access the hardware screen buffer, which might impact performance. In order to preview what the camera is scanning, the LCD Overlay allows applications to

access the screen buffer directly by bypassing the Palm OS layer of software, as shown in the following diagrams:



Without LCD Overlay **With LCD Overlay**

The LCD Overlay uses the YUV color model. For this reason, it is easy for a decoder to use. Please note that although decoders usually separate three individual buffers for Y, U, and V color definitions, the LCD Overlay only expects one pointer.

5.2.2.1 External dialogs

External dialogs are pop-up windows used by applications to notify the user that an action is taking place. Examples include battery level notifications, volume level graphics, or SMS message dialogs.

Only the LCD Overlay is aware of what is being rendered on the screen when it has control of the screen. (The LCD Overlay is beyond control of the Windows Manager.) For this reason, applications must handle external dialog pop-ups whenever necessary.

On Treo 700p smartphones and later, two notifications have been created to allow applications to handle external dialogs:

- `#define sysNotifyExternalDialogOpenedEvent` is sent when an external dialog is about to be displayed.
- `#define sysNotifyExternalDialogClosedEvent` is sent when an external dialog is about to be closed.

The following use case will describe how to use these notifications.

Assume that an application called PlayerX is currently playing video as the active application. Another application called LauncherY wants to display a mini-LauncherY external dialog when the Side key is pressed.

Since it is not possible for PlayerX to know that it should pause the video while the mini-LauncherY dialog is being displayed, LauncherY should broadcast `sysNotifyExternalDialogOpenedEvent` before it tries to draw an external dialog, then broadcast `sysNotifyExternalDialogClosedEvent` when it is finished displaying the dialog.

PlayerX should register for both notifications and keep track of how many external dialogs are currently open. If PlayerX is playing video when it receives `sysNotifyExternalDialogOpenedEvent`, then it should pause the video until all external dialogs are closed.

5.2.2.2 For more information

The LCD Overlay header files include:

- `palmOneLcdOverlay.h`
- `palmOneLcdOverlayCommon.h`

For more detailed information on the LCD Overlay header files, refer to the *Palm API Guide, Palm OS Platform* at the Palm Developer Network (PDN).

Also, refer to the Sample Code section of the Palm OS SDK for coding examples.

5.2.3 Camera Manager

Available on:

- Centro™ and Treo™ smartphones
- Zire™ 72 handhelds

This section provides reference information for the Camera Manager. You can use the functions in this library to turn the camera on, control the settings, and to allow an application to capture and preview images and video.

The Camera Manager is declared in the header files `palmOneCameraCommon.h` and `palmOneCamera.h`.

IMPORTANT: For information on a known issue with Camera Manager and streaming on Treo 700p, Treo 755p, and Centro smartphones, see **Section 5.5.2 on page 73**.

If the device includes a camera slider, such as the Zire 72 handheld, camera slider notification information is declared in the header file `PalmCameraSlider.h`. For more information on the camera slider, see **Section 5.2.3.3 on page 64**.

There are three versions of the Camera Manager: Version 1, Version 2, and Version 3. Version 2 has several features in addition to those in Version 1. Version 3 includes minor changes to Version 2, such as additional image formats. The differences are noted throughout this section.

NOTE: The Camera Library is currently still supported for the Treo 600 smartphones. In the future, however, the Camera Manager should be used instead.

5.2.3.1 Using the Camera Manager

Depending on the hardware available on a particular device, Camera Manager settings and functionality may or may not be available. For this reason, you should always use the `CamLibControl` query commands to check for the features and settings that are available. For more information, see the information on `CamLibControl` in the *API Guide*.

Programmatically, you will know if a camera is available on a device if the Camera Manager is present.

The Camera Manager is more difficult to use than the Photo Library, but it does provide options to control settings for photographic contrast, exposure, sharpness, and light.

When using the Camera Manager, an application must implement many steps manually. For example, to display a preview window on the screen, the application must communicate with the camera driver and manually place the image data on the screen.

Also, keep the following points in mind when using Camera Manager:

- The Camera Manager always returns bitmap data.
- The Camera Manager communicates directly with the camera driver.
- The Camera Manager does not use the Codec Plug-in Manager.
- The Camera Manager can be used to capture images manually, but there is no user interface.

To use the Camera Manager to capture images, an application should use the following steps:

1. Open the Camera library.
2. Turn the camera on.
3. Proceed with camera functionality, such as turning on image preview, adjusting settings, and so forth.

5.2.3.2 Resources required for camera functionality

The following requirements are necessary to take advantage of camera functionality:

- To preview images, capture images, and display the camera configuration dialog box, the device must support 16-bit color-depth mode. The camera configuration dialog box may or may not be present on a particular device.
- Depending on the available camera hardware, images can be previewed in resolution sizes listed in the following table:

Format	Size	Memory	Use
SXGA	1280 x 1024	2.6MB	images
SXGA	1280 x 960	2.4MB	images
VGA	640 x 480	600KB	images
QVGA	320 x 240	150KB	images/video
QQVGA	160 x 120	37.5KB	images/video
QCIF	176 x 144	50KB	images/video
CIF	352 x 288	198KB	images/video

NOTE: Video is not available in Camera Manager V1. The QCIF and CIF formats are not available in Camera Manager V1 or V2.

For a matrix of available image formats by device, see **Section 5.1.4 on page 56**.

5.2.3.3 Using the Camera Slider on Zire™ 72 handhelds

The Zire 72 handheld includes a camera slider that opens to reveal a camera shutter button and camera lens. Camera slider notification information is declared in the header file `PalmCameraSlider.h` for the Zire 72 handheld.

Applications for the Zire 72 should check for the presence of a camera slider, register for camera slider notifications, and then turn the camera on if the slider is open or if no slider is present. Typically, an application should follow this general workflow:

1. Open the Camera library.
2. Check to see if a camera slider is present.
3. If a camera slider is present:
   - Register for camera slider notifications.
   - Check to see if the slider is open.
4. Turn the camera on.
5. Proceed with camera functionality, such as turning on image preview, adjusting settings, and so forth.

5.2.3.4 For more information

The Camera Manager header files include:

- `palmOneCamera.h`
- `palmOneCameraSlider.h`
- `palmOneCameraCommon.h`

For more detailed information on the Camera Manager header files, refer to the *Palm API Guide, Palm OS Platform* at the Palm Developer Network (PDN).

Also, refer to the Sample Code section of the Palm OS SDK for coding examples.

5.2.4 JPEGLib, CameraLib, and ImageLib

The JPEGLib has been deprecated and removed from the document *Palm API Guide, Palm OS Platform*, which is part of the SDK download.

The CameraLib and ImageLib still exist, and are included in the API Guide.

5.2.4.1 CameraLib

CameraLib consists of:

- The old Camera Library header files CameraLib.h and CameraLibCommon.h.
- The Camera Manager header files palmOneCamera.h, palmOneCameraSlider.h, and palmOneCameraCommon.h.

To provide camera support to applications on Zire 72, Treo 650, and later devices, use the Camera Manager Library header files instead of the old Camera Library. Refer to **Section 5.2.3 on page 62** for more information.

5.2.4.2 ImageLib

ImageLib provides support for image drawing and manipulation on Treo 600 devices (using the header files ImageLib.h and ImageLibCommon.h).

For Treo 650 and later devices, use the palmOne Photo Library before using the Image Library. The Photo Library allows better integration with the built-in Media application and storing images on the device.

To use this library, first check the version of the library and then make the appropriate API calls depending on the version. The header files for the Photo Library are included in the Imaging Library (palmOnePhoto.h and palmOnePhotoCommon.h).

Refer to **Section 5.2.1 on page 59** for more information on the Photo Library.

5.3 Audio

5.3.1 Voice recording and sound libraries

Available on:

- Centro™ and Treo™ smartphones
- LifeDrive™ mobile manager
- Tungsten™ T5, Tungsten™ C, and Palm® T|X handhelds
- Zire™ 72 handhelds

5.3.1.1 Palm® OS Sound Manager library

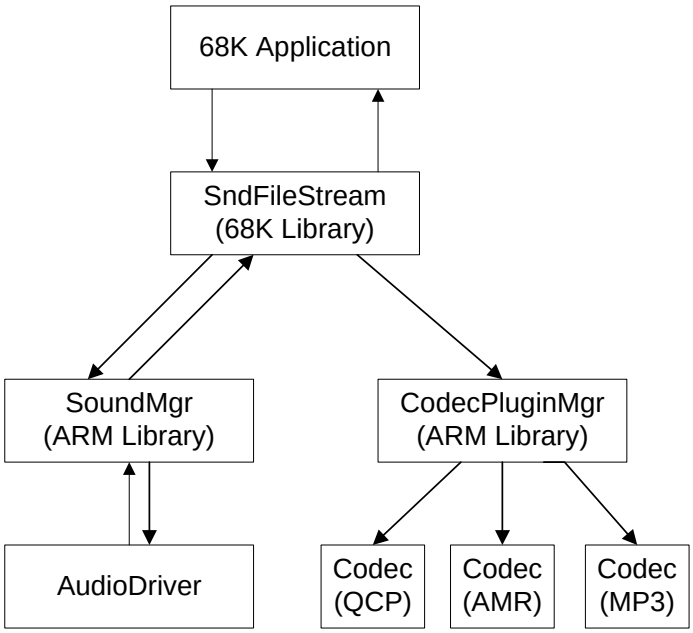
In general, use the Palm OS Sound Manager Library `HsSoundLib` to control voice recording and sounds, create audio streams, and support audio stream playback. This Library supports both synchronous and asynchronous mode.

5.3.1.2 Sound File Stream library (SndFileStream)

`SndFileStream`, a 68k shared library, is used mostly by audio applications to play and record audio files in different formats by managing the Sound Manager and the Codec Plug-in Manager. It provides support for formats including AMR, WAV, QCELP, and MP3. `SndFileStream` also provides support for recording and playing audio from files.

`SndFileStream` is declared in the header file `palmOneSndFileStream.h`.

The following diagram details how an application can use `SndFileStream` to work with other components to play and record audio.



5.3.1.2.1 Audio playback

In an audio playback application, `SndFileStream` parses the audio header information in the audio file that is being played. This header file includes:

- File format
- Sample rate
- Duration
- Meta data

`SndFileStream` then uses the parsed info to create an audio stream through Sound Manager. Depending on the audio format, it also creates a decoding session through the Codec Plug-in Manager. Next, `SndFileStream` manages reading bitstreams from the audio file, passing the bitstream data to the codec, and passing the decoded audio samples to the Sound Manager to play.

For the Treo 700p smartphone, there are APIs that help retain audio sync. For more information, refer to the *Palm API Guide, Palm OS Platform*, or see the `HsSoundLib.h` header file.

5.3.1.2.2 Audio recording

In an audio recording application, there is no header file to parse. Instead, the application is responsible for passing information to `SndFileStream`. `SndFileStream` must then go through the following steps:

1. Generate a header file.
2. Write the header to the recording file (before writing any recording data to that file).
3. Create an audio stream through the Sound Manager.
4. Create an encoding session through the Codec Plug-in Manager.
5. Get raw Pulse Code Modulation (PCM) audio data from Sound Manager.
6. Pass raw data to the codec to encode.
7. Write the encoded bitstream data to the recording file.

5.3.1.2.3 Audio pause and resume

For Treo 680 smartphones and later, audio pause and resume functionality has been added to the `HsSoundLib.h` header file. To access these features, use the calls `PmSndStreamPause` and `PmSndStreamResume`.

5.3.1.3 Changes for Centro™ smartphones

For Centro smartphones, the header file `HsSoundLibCommon.h`, located at *Incs/Common/Libraries/HsSoundLib/*, now includes two new sound port configuration types:

```
/**<Input port used by OutRadioMic switch to get uplink audio from BT
headset. */
#define hsSndPortID_InBtHeadset       HS_AC97(0x00000015)

/**<Input port used by OutRadioMic switch to get uplink audio from BT
handsfree mode. */
#define hsSndPortID_InBtHandsfree     HS_AC97(0x00000016)
```

Also, for Centro smartphones, the version of the sound library is defined as:

```
/**<Version of the sound library. */
#define hsSndLibVersion               0x010D0000
```

5.3.1.4 For more information

The voice recording and sound header files include:

- `HsSoundLib.h`
- `palmOneSndFileStream.h`

For more detailed information on the voice recording and sound header files, refer to the *Palm API Guide, Palm OS Platform* at the Palm Developer Network (PDN).

Also, refer to the Sample Code section of the Palm OS SDK for coding examples.

5.3.2 Tones library

Available on:

- Centro™ and Treo™ smartphones

The Tones library is used to manage the two phone ring tone databases, the "MIDI Ring Tone DB" and the "Palm OS Tone DB". The `TonesLib.h` header file, which is provided with the Palm SDK, contains all the public information referenced in this section, including constants, structure definitions, and function prototypes.

Use the Tones library to create personalized ring tones. Supported formats include:

- AMR
- MIDI
- MP3
- QCELP
- WAV

NOTE: MIDI ringtones are limited to 64KB in size. Other types do not have this limit.

The Tones library uses the Codec Plug-in Manager to provide codecs. Codecs then convert between audio formats.

5.3.2.1 Ring tone databases

Two ring tone databases are accessed by the Tones library, the "MIDI Ring Tone DB" and the "Palm OS Tone DB."

- The MIDI Ring Tone database contains all MIDI data for ring tones, and is accessed directly by the Tones library. This database is used for backwards compatibility.
- The Palm OS Tone database is used for WAV, AMR, QCELP, and MP3 file types. It contains only the names of the tones stored as file streams. This database is specific to alert tones.

Each record in the MIDI Ring Tone database should contain one MIDI sound, using a Format 0 Standard MIDI File (SMF). For more information on the database format, see the Sound Manager see the Sound section of the *Palm OS Companion and Reference* available at **http://www.access-company.com/developers/documents/palmos/palmos.html**.

The following table contains the details of the system MIDI Ring Tone database.

Feature	Description
Database name	`#define TonesDBName 'MIDI Ring Tones'`
Database type	`smfr`
Creator code	`#define hsFileCMultiChannelRingToneDB MCnl`

In order for the system to recognize a new ring tone, the tone must be installed in the database using an alarm management tool. If you are creating ring tone management applications, you may want to keep a separate database of archived ring tones.

We recommend that you use the attributes listed in the following table for such a database.

Feature	Description
Database type	`smfr`
Creator code	`HSsf`

You can use standard Palm OS database calls to install the sound records into this database from their own application. To do this, typically, you would create a MIDI sound database, install it on the smartphone, and then use an alarm management program to copy the sounds into the MIDI Ring Tone database.

5.3.2.1.1 Restoring the Ring Tone databases

The MIDI Ring Tone and Palm OS Tone databases are stored in RAM in order to allow applications to add and delete ring tones. The OS also includes a copy of this database frozen in the system image. The database is copied to RAM after a hard reset or if the database has been deleted from RAM.

To restore original ring tones, simply delete the database.

If ring tones are lost due to a hard reset, you may use HotSync to replace the data.

5.3.2.1.2 Ring tone tools

Palm smartphones use standard Palm OS MIDI files for ring tones, and all popular third-party tools for creating Palm OS system sounds can be used to create ring tones.

5.3.2.1.3 Example code for creating and adding ring tones

To add a new ring tone programmatically to the Tones library using a single interface, use the `TonesLibToneCreate()` and `TonesLibToneWrite()`, and `TonesLibToneClose()` functions, as shown in the example code below:

```
Err TonesLibToneCreate (UInt16 refNum, UInt32* streamP, ToneType toneType,
       CharPtr toneName, Boolean protectedTone);

TonesLibToneWrite(ref,stream,soundBuf,bufSize);

Err TonesLibToneClose (UInt16 refNum, UInt32 stream);
```

In this example, the `TonesLibToneCreate()` function includes the following parameters:

- `refNum` - The reference to the Tones library, which is already open.
- `streamP` - This parameter is returned to the caller and should be used during the calls to `TonesLibToneWrite()` and `TonesLibToneClose()`.
- `toneType` - This is the format of the tone, which may be:
  - `toneTypeMIDI`
  - `toneTypeWAV`
  - `toneTypeQCELP`
  - `toneTypeAMR`
  - `toneTypeMP3`
- `toneName` - This is a string limited to 31 characters.
- `protectedTone` - This parameter instructs the Tone library to enforce forward lock on the created ringtone.

The `TonesLibToneClose()` function includes these parameters:

- `refNum` - The reference to the Tones library, which is already open.
- `streamP` - This parameter identifies which stream is being closed. It is the value that was returned from `TonesLibToneCreate()`, which will be returned to the caller, and should be used during subsequent calls to `TonesLibToneWrite()`.

In this example, the `TonesLibToneCreate()` function accepts a `toneType` parameter. If the `toneType` parameter is set to `toneTypeMIDI`, the tone is created and `TonesLibToneClose()` internally calls `TonesLibAddMidiTone`.

Otherwise, `TonesLibAddMidiTone` can be called separately, as shown in the code snippet below:

```
TonesLibAddMidiTone(ref,midiTone,''Yes'',false);
```

NOTE: `The TonesLibToneWrite()` function can be called repeatedly (in a loop, for example) and the data will be appended to the open stream. This way, the caller does not need to buffer an entire audio file before calling `TonesLibToneWrite()`.

In the next code example, the application scans the SD card for MIDI, WAV, or MP3 files and allows them to be added to the Tones library.

```
static Err PrvAddToneToTonesLib(UInt16 vfsVolume, UInt16 toneType, Char *fileName,
Char *path, Char *storeNameP)
{
       Err err = errNone;
       Char fullPath[512];
       FileRef fileRef = NULL;
       UInt32 newTone = 0;
       UInt8 *dataP = (UInt8*)MemPtrNew(4096);
       UInt32 numBytesRead;
       UInt16 tonesLibRefNum = sysInvalidRefNum;

       if(!dataP) goto Done;

       // Load TonesLib
       err = SysLibFind(tonesLibName, &tonesLibRefNum);
       if (err)
          err = SysLibLoad(tonesLibType, tonesLibCreator, &tonesLibRefNum);

       if(err) goto Done;

       // Create full path
       StrCopy(fullPath, path);
       StrCat(fullPath, fileName);

       // Open the file
       err = VFSFileOpen(vfsVolume, fullPath, vfsModeRead, &fileRef);
       if(err) goto Done;

       // Create tone
       err = TonesLibToneCreate(tonesLibRefNum, &newTone, toneType, storeNameP, false);
       if(err) goto Done;

       // Write the tone
       while(!err)
       {
          err = VFSFileRead(fileRef, 4096, dataP, &numBytesRead);
          if(numBytesRead == 0) break;

          err = TonesLibToneWrite(tonesLibRefNum, newTone, dataP, numBytesRead);
       }
```

```
Done:
    if(dataP) MemPtrFree(dataP);
    if(fileRef) VFSFileClose(fileRef);
    if(newTone) TonesLibToneClose(tonesLibRefNum, newTone);

    return err;
}
```

5.3.2.2 For more information

The Tones library header files include:

- `TonesLib.h`

For more detailed information on the Tones library header files, refer to the *Palm API Guide, Palm OS Platform* at the Palm Developer Network (PDN).

Also, refer to the Sample Code section of the Palm OS SDK for coding examples.

5.4 Video playback

Palm devices do not currently provide a framework for video playback. This section is not meant to be a tutorial on creating a video playback application. It is only a guideline for what an application would need to do to provide such functionality.

To provide video playback, an application must work with Photo Library, Camera Manager, the LCD Overlay, and the Codec Plug-in Manager by performing the following tasks:

1. Parse the video file for video and audio content.
   - Determine the kinds of tracks that are to be played: video, audio and/or both.
   - Determine the track formats.
   - Determine the target resolution.
2. Query the Codec Plug-in Manager for the necessary codec(s).
3. Split (demultiplex) the original content file into separate audio and video streams.
4. Create a Codec Plug-in Manager session for each codec you will use. (For example, one for audio and one for video.)
5. Call `CodecMgrEncodeDecode` for every data packet for the audio and video streams.
6. Scale the resolution of the video.
7. Manage synching video and audio together.
   - For audio, the sampling rate tells you how much data must be played back. Use the provided time stamps (the reference clock).
   - For video, a timestamp is provided, but only the audio timestamp is usually used as a reference.
8. Send the decoded and scaled content to the screen and speaker for playback.

5.5 Streaming

Available on:

- Centro™ and Treo™ smartphones

Streaming audio and video is available for Centro and Treo smartphones.

For more information on streaming specific to the Blazer® 4.5 web browser, see the following sections:

- **Section 11.1.3.2 on page 204**
- **Section 11.1.3.4 on page 208**
- **Section 14.1.4.4 on page 264**.

5.5.1 Best practices for encoding video for streaming

To encode video for the best streaming experience on EvDO products such as Treo 700p and Treo 755p smartphones, use the following best practices:

- Consider formatting a small render-friendly web page to host your streaming content. For an example, see **http://mobile.palm.com**.
- Links should be delivered via RTSP:// or progressively 'streamed' through an http:// link.
- For encoding, the total bit rate should not exceed 200kbps, which includes video and audio.
- Use AAC and AAC Plus Audio formats only.
- Use the .MP4 extension.
- Use a screen resolution of 320 x 240.
- For streaming, use 15fps at 128kbps (MPEG-4 or H.264 / stereo AAC or AAC Plus).
- For sampling, use 128kbps at 48kHz.

5.5.2 Known issue

In the Treo 700p, Treo 755p, and Centro smartphones, the following API does not work with streaming, and is a known issue:

```
CamLibControl([LibRefNum], kCamLibCtrlStreamStart, [&StreamType]);
```

In this API, the key parameter is the second parameter, which controls the camera. The first and third parameters, surrounded by brackets, may be customized.

For streaming, the second parameter is `kCamLibCtrlStreamStart`. When you use this API call with this parameter to stream in the Treo 700p smartphone, the call will fail, although `CamLibControl` will still return `0`. Returning `0` indicates that the function is successful, even though it failed. The camera will not be turned on.

Currently, there is no workaround. For more information on streaming with your application, refer to the Streaming code example in **Section 11.1.3.4 on page 208**.

This page intentionally blank

CHAPTER 6

6. Data Communications

This chapter details the data communication features and APIs available in the Palm OS® SDK.

6.1 NetPref Library API

Available on:

- Centro™ and Treo™ smartphones

This section provides detailed information about the NetPref Library API.

The NetPref Library was created to provide better support for the GSM/GPRS and CDMA/1XRTT network parameters, dynamic UI flags, Home/Roaming network configurations, CCSM database utilization, synchronization with IOTA-provisioned settings, and configuration of fallback services required by the features of the Palm smartphone. The network database access was redesigned by moving database and record access operations from the Network panel into the NetPref Library.

6.1.1 Loading the library

The NetPref Library is designed as a Palm OS shared library. The NetPref Library should be loaded for use and unloaded after use by a client application. Link the NetPref Library when you need it and then unlink it when you have finished. The system does not load the NetPref Library at reset or start-up time and leave it permanently installed, as is done with some other libraries. This method helps avoid some HotSync® conflicts, such as an attempt to install over a protected database.

For examples of linking and unlinking, refer to `NetPrefUtilNetPrefLibLink` and `NetPrefUtilNetPrefLibUnlink` as defined in the `NetPrefUtils` package.

The following code sample demonstrates how to link the NetPref Library based on the `NetPrefUtil` package.

```
extern Boolean
NetPrefUtilNetPrefLibLink (NetPrefUtilNetPrefLibType* netPrefLibP)
{
   Boolean          isSuccessful = false;
   Err              err = 0;
   UInt16           refNum = 0;
   NetPrefContextTypeTag* cxtP = NULL;

ErrNonFatalDisplayIf (!netPrefLibP, "null arg");
  ErrNonFatalDisplayIf (netPrefLibP->linkSignature
              == netPrefUtilNetPrefLinkSignature,
              "NetPref lib already linked");
   err = SysLibLoad (netPrefLibTypeID, netPrefLibCreatorID,
                     &refNum);
   if (err)
      {
      ErrNonFatalDisplay ("failed to load NetPrefLib");
      goto Exit;
      }
   err = NetPrefLibOpen (refNum, &cxtP);
   if (err)
      {
      ErrNonFatalDisplay ("failed to open NetPrefLib");
      goto Exit;
      }
   // "Construct" the NetPref lib "instance"
   isSuccessful = true;
Exit:
   if (err)
      {
      if (refNum != 0)
         SysLibRemove (refNum);
      MemSet (netPrefLibP, sizeof(*netPrefLibP), 0);
      }
   return (isSuccessful);
} // NetPrefUtilNetPrefLibLink
```

6.1.2 NetPref Library information

The following table shows the attributes of the NetPref Library and related information. For more detail, see the *Palm API Guide, Palm OS Platform*.

Description	Attribute
Creator ID	`HsNP`
Type ID	`libr`
Library database name	`NetPrefLibrary`
Library name	`HsNetPrefLibrary.lib`

Header files	`NetPrefUtils.h`
	`NetPrefLibrary.h`
	`NetPrefLibTypes.h`
	`NetPrefLibErrors.h`
	`NetPrefLibTarget.h`
	`NetPrefLibFieldInfoTable.h`

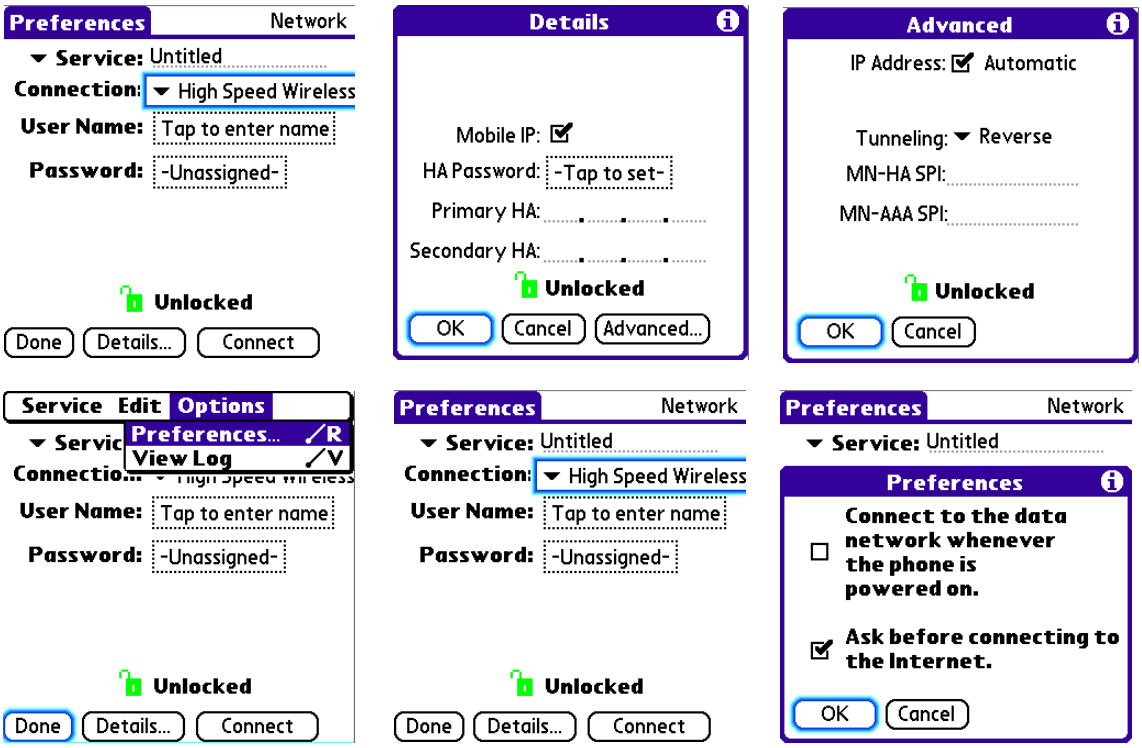
6.1.3 NetPref panel

The Network Preference panel has been modified to provide support to the various Palm smartphone features that are not possible by using the original Palm OS software (3.5 and 5.x) Network Preference panel implementation. Changes include parameters specific to GSM/GPRS and CDMA/1-X (Simple-IP & Mobile-IP), IOTA support, CCSM table support, and various UI features such as hiding certain fields and locking certain services.

In other changes, the network database record format was extended in a backward-compatible way, and the network database access logic was separated into the NetPref Library. The Network Preferences panel as well as other system components, such as NetMaster library, the IOTA application, and network profile creator, uses the NetPref Library to read, write, create, duplicate, and delete network service profiles.

In addition, the configuration of NetLib was redesigned to dynamically perform during each network login instead of doing so only when a user selects a service. This dynamic configuration implementation was moved from the Network panel to the NetMaster library, which is described later in this section. This change permits support for dynamic network configuration based on location, such as Home versus Roaming, executing GPRS and One-X specific functions during login, and implementing the service fallback feature.

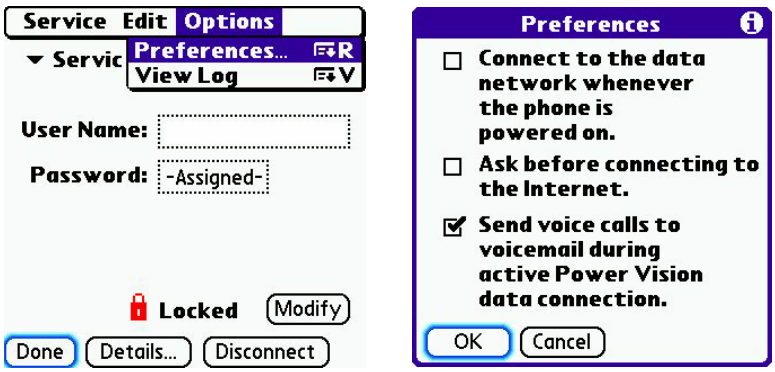
The following are examples of Preference Panel displays.



The Sprint Treo 700p and Treo 755p smartphones includes an option that lets a user decide whether or not they want to interrupt an active data connection with an incoming call, or if the call should be sent to voicemail. See the following screens for settings.

To set this option in the Network Preferences panel, users must select **Network > drop-down menu > Options > Preferences**. Whether a user sets this feature or not, an application does not need to account for it in any way.

NOTE: Of the three options on the Preferences form, only the option, "Send voice calls to voicemail during active Power Vision data connection" is new to the Sprint Treo 700p smartphone only. This option is not available for Verizon.



The Network Preference panel supports the legacy Network Preference panel's limited launch code API, such as enumerating profile names, getting or setting the default profile, and so forth. Applications such as the HotSync application use this API to select the appropriate network preference profile. This implementation provides full backward-compatibility with the legacy API.

- `sysAppLaunchCmdPanelCalledFromApp`

  Displays the network panel as if it were a dialog box popped up from the calling application, returning to the calling application when the dialog box is dismissed by tapping the Done button, for example.

- `svcCFACmdQuickEdit`

  Manifestation of `sysAppLaunchCmdPanelCalledFromApp` that brings up the "quick- edit" form of the panel, such as the phone number form for a dial-up service profile.

  - `sysSvcLaunchCmdSetServiceID` - Sets the default service.
  - `sysSvcLaunchCmdGetServiceID` - Gets the default service.
  - `sysSvcLaunchCmdGetServiceList` - Gets a list of service names and corresponding IDs.
  - `sysSvcLaunchCmdGetServiceInfo` - Gets the service name when the unique ID of the service is given.
  - `sysSvcLaunchCmdGetQuickEditLabel` - Gets the "quick-edit" value string to display to the user, such as the phone number value from a dial-up service profile.

NOTE: The network preference database has been restructured to support the new parameters needed by the persistent data connection. Any preexisting application that reads, creates, or modifies network services or profiles directly in the original Palm OS network preference database will probably not work with the products that support the latest architecture. If standard Palm OS 3.5 APIs were used, such applications might still work. Legacy profiles are supported because they are converted to the new structure of the database the first time they are accessed. To create or read network profile information in new applications, use the NetPref Library. The library provides all the necessary routines to interface with the network preference database. There is no longer any need to read/write directly to the network preference database.

6.2 NetMaster library API

Available on:

- Centro™ and Treo™ smartphones

This section describes in detail the NetMaster library features and APIs.

Palm created the NetMaster library to:

- Provide better support for handling and managing network logins, network service connections, network context, and session controls needed by network applications in the Palm smartphones.
- Allow multiple concurrent applications or tasks to detect, log in, preempt, and switch network profile connections.

Palm® designed the NetMaster library API to work around the limitations of the current Palm OS® network architecture. For example, the MMS application, running in the background, may receive a trigger and need to log in with the MMS APN (network profile), while the browser application has NetLib open with the Internet or browser-specific APN. NetLib does not provide a way to arbitrate between the needs of both applications. Thus, NetLib could not support multiple concurrent data sessions over the same network interface (network profile).

The NetMaster library API attempts to bridge the gap by keeping track of the current NetLib usage by various tasks and making intelligent decisions allowing when a particular network profile connection request may preempt and shut down another previously active profile. This API also attempts to simplify and make more robust the process of logging in with a network profile that is not the "default" network profile. Before this API, applications had to get the current default network profile ID, set a new default, call NetLibOpen, do their networking, and then restore the default ID. This works if there is only single data networking application at a time. However, with multiple concurrent clients possibly running different tasks, system data integrity (current or default profile ID) is susceptible due to ill-managed multiple writers, system reset, or system crash.

NOTE: The NetMaster library API is *not* intended for all networking applications. It is intended to be a temporary solution to a temporary limitation in Palm OS software's Network library. Future versions of Palm OS software or NetLib may make it impossible or unnecessary for this API to be supported, and it would then be deprecated.

Most networking applications that connect using the "default" network profile (as selected in the Network Panel) should continue to use the standard NetLib API (NetLibOpen and NetLibClose) in order to remain compatible with future releases of Palm OS software.

6.2.1 Usage model

This section details important information and guidelines for using the NetMaster library API:

- Only the Motorola 68K version of the NetMaster Data Session Control API is provided. However, ARM code may call this API using the Palm OS software 5.x PACE API, including `PceSysLibFind68Klibrary` and `PceCall68KlibRoutine`.
- *Do not* intermix the NetMaster Data Session Control API with NetLib's *session control* API, which includes `NetLibOpen`, `NetLibOpenConfig`, `NetLibOpenIfCloseWait`, `NetLibClose`, `NetLibConnectionRefresh`, `NetLibIFUp`, `NetLibIFDown`, `NetLibIFAttach`, and `NetLibIFDetach`. You *must* use either the one set of session control API or the other, exclusively.
- The NetMaster Data Session Control API works closely with the NetPref Library. Applications first look up the desired network profile record using the NetPref Library API, and then provide this record's ID to `NetMasterSCNetAttach` via the `netPrefRecID` parameter. This API only works with network profiles that are managed by the NetPref Library. Other profiles are *not* supported.
- The design of the NetMaster Data Session Control API is based on the concepts of data sessions and data contexts. A given data session represents an established (logged-in) data session via a specific network profile ID. An *active* data session is one that is believed to be logged in; otherwise, it is *inactive*. A given data context represents a *client* (an executable such as an application or library) that successfully attached to a data session by calling `NetMasterSCNetAttach`.
- Multiple data contexts may be "attached" to a single data session. The Data Session Control API provides functions that operate on both data contexts and data sessions. Most clients utilizing this API use the functions `NetMasterSCNetAttach`, `NetMasterSCNetDetach`, `NetMasterSCSessionIDGetFromContext`, and `NetMasterSCSessionIsActive`. The remaining functions are mainly for troubleshooting purposes.
- A client begins networking over a particular network profile by calling `NetMasterSCNetAttach( )` with the desired network profile ID and other specialized parameters. The client uses `NetMasterSCNetAttach` instead of NetLib's `NetLibOpen` or `NetLibOpenConfig`. If there is already an active data session where the requested network profile ID is either the primary or fallback profile, `NetMasterSCNetAttach` attaches the caller to that data session. See the NetPref API documentation for definitions of primary and fallback profiles.

  If, on the other hand, the active data session was established from a different network profile, `NetMasterSCNetAttach` employs an internal algorithm to see if it can preempt (shut down) the active data session, and to do so if it can. If it cannot make room for a new data session by tearing down an existing data session, the function fails.

  If there is no active data session or if one was successfully shut down to make room for the new data session, `NetMasterSCNetAttach` attempts to establish a new data session with the requested network profile, and attach the client to that data session. If it's successful, `NetMasterSCNetAttach` returns an error code of 0 and a nonzero Context ID of a newly created data context that is attached to the data session. The returned Context ID may be passed to other functions of this API family that require a data context ID. This data context ID is valid only until

you destroy it by calling `NetMasterSCNetDetach`. Once the Context ID is destroyed, it *must not* be passed to any functions.

If `NetMasterSCNetAttach` fails, you may need to retry at a later time. The error code `netErrAlreadyOpenWithOtherConfig` indicates that `NetMasterSCNetAttach` could not preempt another active data session. Other nonzero error codes are typically either NetLib or NetMaster library error codes. When the client has finished using the network session that was acquired with `NetMasterSCNetAttach`, such as when exiting the networking application or terminating its task, the client *must* destroy the data context by calling `NetMasterSCNetDetach`. Users of the NetMaster Data Session Control API call `NetMasterSCNetDetach` instead of `NetLibClose`.

- Once attached to a data session, the client may periodically call `NetMasterSCSessionIsActive` to check if that session is still logged in. `NetMasterSCSessionIsActive` takes a data session ID as a parameter. Call `NetMasterSCSessionIDGetFromContext` to get the data session ID from a valid data context ID. If it reports that the data session is not active, this means that the data session was shut down for some reason, such as loss of network coverage for an extended time, wireless mode being turned off, preempted by another client requesting a different network profile, and so on. When this occurs, you should destroy the data context by calling `NetMasterSCNetDetach`, since this API has no concept of data session reactivation. If you need to resume your data session, you need to get a new data context via `NetMasterSCNetAttach`, as discussed previously.
- If the anchor timeout that the application initially requested when calling `NetMasterSCNetAttach` is now bigger than necessary, the application *must* reduce its anchor timeout to the minimum acceptable value by calling `NetMasterSCContextAnchorTimeoutSet` so that other services may be activated more quickly when necessary.

6.2.2 Loading the library

The NetMaster Library is designed as a Palm OS shared library. The system software preloads the NetMaster library at system reset/startup before broadcasting the `sysAppLaunchCmdSystemReset` launch code. Clients of the NetMaster library should only call `SysLibFind (netMasterLibName,...)` to get a library `refNum` of NetMaster. If `SysLibFind` returns a nonzero error, clients *must* assume that NetMaster was not loaded for a good reason (such as when the user performs a safe reset), or that NetMaster is simply not present on the system, and fail gracefully.

NOTE: Clients must not load NetMaster themselves such as via `SysLibLoad` or `SysLibInstall`. Keep in mind that if the system didn't load it, there was a good reason.

The following code sample demonstrates how to use the NetMaster library.

```
UseNetMasterLibrary(UInt16* libRefP)
{
    Err error  = 0;
    Err ifErrs = 0;

    // Routine is pointless without this parameter
    if (!libRefP)
        return memErrInvalidParam;

    // Get the NetMaster Library
    error = SysLibFind("HsNetMasterLibrary.lib", libRefP);
    if (error)
    {
        //It's not already here—don't load it:
        //there is a good reason it's not loaded
    }
    return error;
}
```

6.2.3 Library information

The following table contains the attributes of the library and related information.

NOTE: The library code is compiled for structures to be aligned on 2-byte boundaries. This is especially important for CodeWarrior users, as the default setting might not be compatible.

Description	Attribute	Comment/Description
Creator ID	`HsNM`	
Type ID	`libr`	
Library's database name	`NetMasterLibrary`	
Library name	`HsNetMasterLibrary.lib`	

Header files	`NetMasterLibrary.h`
	`NetMasterLibErrors.h`
	`NetMasterLibTarget.h`
	`NetMasterLibTraps.h`
		By including `NetMasterLibrary.h` you are effectively including the other files.

6.3 Email client best practices

This section details general best practices for email client behavior, especially for working with MMS.

- If the NetMaster Library is present on the device, the network session should be established using `NetMasterSCNetAttach` with Low Priority (not Default Priority). Otherwise, if the device does not have the NetMaster Library, use `NetLibOpen`.
- The email client should never assume that it is the only networking application running, or that it is the most important application running. It should NOT force a network interface to shut down or detach.
- If the NetMaster Library is present on the device, use NetMaster APIs to create the data context. Otherwise, use NetLib. If the attempt fails, the email client should try again later rather than force the system to bring up PPP over the session it needs.
- The email client should be registered for NetLib media `up` and `down` notifications. When appropriate, it should check if PPP is up or down using `NetLibMaster` APIs and find out what service is currently associated with the PPP connection using `NetLibIFSettingGet`.

6.4 HTTP library

Available on:

- Centro™ and Treo™ smartphones
- LifeDrive™ mobile manager
- Tungsten™ T5, Tungsten™ E2, and Palm® T|X handhelds

The HTTP library is a shared library. This HTTP library was added to Palm OS software 5.X to give applications a high-level interface to implement HTTP access.

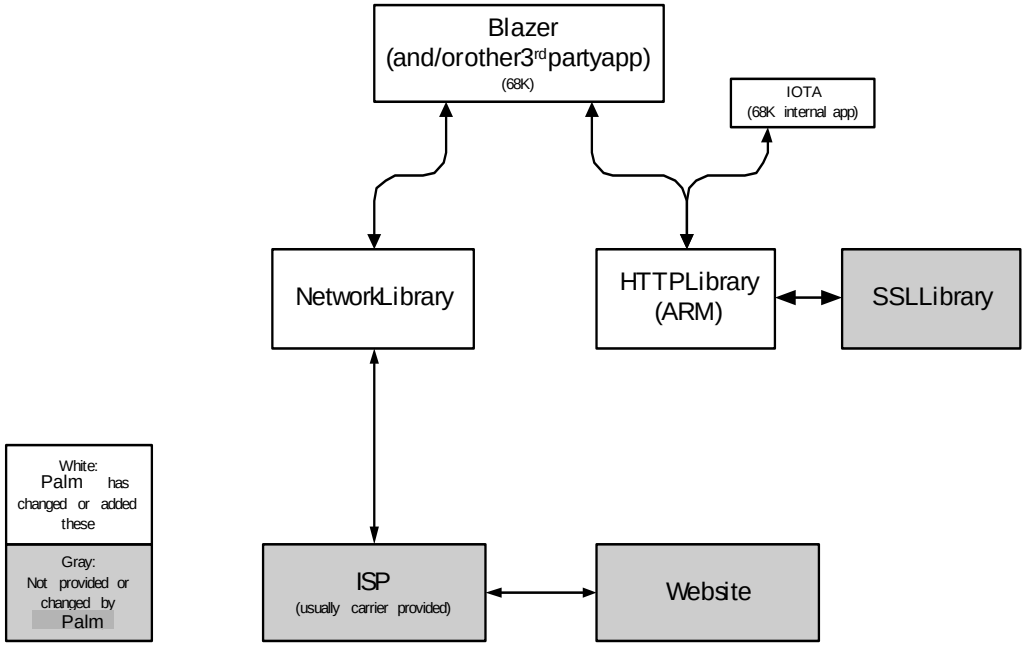
The HTTP library is used by the Palm Blazer® web browser and other applications. Third-party applications can share and use the HTTP library to implement their own HTTP access requirements, however, they should not access the HTTP library at the same time. To indicate this, an error message has been added with the Treo 680 smartphone. For more information, see **Section 6.4.3 on page 87**.

The HTTP library supports HTTP protocol versions 1.0 and 1.1 and the WAP 2.0 HTTP client profile. The HTTP library also supports SSL using the Palm OS SSL library for secure (HTTPS) connections, proxy configuration, and cookies.

6.4.1 Architecture

The HTTP library is provided at the Palm OS library level. It is implemented at a peer level with the Palm OS NetLib library and SSL library. Applications must call the NetLib library to initialize and enable the TCP sockets for the HTTP library. It will call the SSL library directly to implement authentication and cryptography necessary for secure access. So, after the NetLib library is initialized in the standard way, the applications will directly call into the HTTP library to implement all HTTP access.

The following figure illustrates the architecture of the HTTP library.



6.4.2 Functional highlights

The HTTP library is designed to be shared by multiple applications, though it is not re-entrant and should not be actively called by more than one application at a time. If the HTTP library is already being used by an application, and another application calls the HTTP library with `HS_HTTPLibOpen()`, the applications may crash.

The HTTP library is implemented as a single instance running in the execution task of caller applications. The library does not launch any additional tasks and is safe to use as a background task, because it does not display a user interface. The library does provide the necessary callbacks to allow applications to handle a user interface in the application code, if necessary.

The HTTP library requires applications to handle the NetLib library interface by passing the HTTP library a reference to NetLib wrapper function callbacks. The HTTP library intelligently manages the pool of sockets. The library provides the calling application an API to specify the maximum number of sockets used by the HTTP library.

The HTTP library supports the following:

- HTTP versions 1.0 and 1.1 as defined in RFCs 1945 and 2616, and the WAP 2.0 HTTP client profile
- Basic, digest, and proxy authentication as specified in RFC 2617
- SSL (HTTPS) connections
- GZIP and Zlib (using deflate) compression formats
- Chunked encoding allowing applications to begin sending data before the application knows the total amount of data to be sent
- Individual application user agent profile - An application can control the user agent profile by adding a simple string or WAP 248 UA Prof to the request header.
- Keep-alive connections - Keep-alive connections are not shared among applications. Different sockets are used for multiple keep-alive connections. For example, two sockets are used to open two separate connections to **www.yahoo.com**.
- Non-blocking sockets

6.4.3 httpErrorLibraryAlreadyOpen error message for Treo™ 680 smartphones

To assist in preventing application crashes, `HS_HTTPLibConst.h` has been updated with an error message. As of the release of the Treo 680 smartphone, if an application has already opened the HTTP library with the call `HS_HTTPLibOpen()`, any other application that makes the same call will be returned the error `httpErrorLibraryAlreadyOpen`. This error message will also be included in future maintenance releases.

For example, if a user is using the Blazer web browser to access the Internet, and a background application, such as MMS tries to open the HTTP library, the background task will receive the error `httpErrorLibraryAlreadyOpen` and the HTTP library will not be opened. This example is illustrated in the following code sample:

```
Err err = HS_HTTPLibOpen();

if(err!= httpErrorOK)

{

ErrNonFatalDisplay(''error opening the httplibrary; it is already in use;
please try again later'');

}
```

6.4.4 HTTP library interface to SSL

Palm OS 5.x and later includes two security-related shared libraries: the SSL library and the Cryptography Provider Manager. These two libraries provide sufficient crypto functionality to negotiate secure SSL connections with most secure websites.

The SSL library in Palm OS 5.x and later includes:

- SSL protocol implementation, version 3 only—not SSL v2 or TLS v1
- RSA public/private key algorithm for key exchange
- RC4 symmetric cipher for bulk data encryption
- Authentication of the server-side of the connection using digital certificates and signature verification
- Message verification using the MD5 and SHA-1 hash algorithms
- SSL session resumption

The SSL library in Palm OS software supports four SSL cipher suites:

- sslCs_RSA_RC4_128_MD5
- sslCs_RSA_RC4_128_SHA1
- sslCs_RSA_RC4_56_SHA1
- sslCs_RSA_RC4_40_MD5

The first two cipher suites include a connection that uses the RC4 symmetric cipher with a 128-bit key for data encryption, the RSA algorithm for key exchange, and either

the MD5 or the SHA1 hash algorithm for verifying message integrity. These two cipher suites are widely supported by popular web servers on the Internet.

The other security-related library included in Palm OS software is the Cryptography Provider Manager (CPM). The CPM exports an API that allows you to perform specific cryptographic operations. The CPM in Palm OS software provides access to the following:

- RC4 symmetric cipher, variable key length
- SHA-1 hash algorithm
- Message verification

6.4.5 HTTP library use of certificates/public key infrastructure

The HTTP library reads a set of root certificates from a database on start-up. These root certificates identify each of the major certificate authorities and are used by the HTTP library during SSL connection establishment to authenticate the remote web server. The database containing the root certificates is burned into the ROM with the HTTP library.

Certain large corporations and institutions act as certificate authorities and issue their own certificates. Such an organization then uses certificates signed using its self-issued root certificate to identify its secure web servers. In order for a user's web browser to negotiate a secure SSL connection with a web server identified by the certificate, the user must add the corporation's self-issued root certificate to his or her web browser's set of trusted root certificates. The HTTP library does not include a native mechanism that allows the user to add trusted root certificates to the certificate database on the user's device. However, you can design your calling application to check the remote server's certificates and display a UI to add them as trusted.

6.4.6 HTTP library implementation

The HTTP library APIs can be categorized into these four groups:

1. Palm OS library management - The Palm OS library management APIs include functions to open, close, sleep, wake, or count the HTTP library.
2. Library initialization and finalization - The initialization and finalization APIs include functions to load, open, initialize, close, and remove the HTTP library.
3. Stream operations - The stream operation APIs include functions to create, configure, send, and receive requests and responses. They also include functions to load, open, initialize, close, and remove the HTTP library.
4. SSL - The SSL category APIs include functions to authenticate, certify, encode, and decode secure connections.

See the *Palm API Guide, Palm OS Platform* for more details about the available HTTP APIs.

6.4.7 General HTTP program information

The following pseudo code demonstrates the usage model of a typical client application using the HTTP library:

```
Library Open
   find or load HTTP library
Library Initialize
   Initialize global environment variables:  application,
   netlib, peer
   Open NetLib library
   Initialize HTTP library.
   Set up connection time-outs
   (Confirm certification)
   (Set proxy)
Stream Create
Stream Initialize
Stream Send Request and Read Response
   Send request
   Loop on Read Response until Read terminates
Stream Close
Library Finalize
Library Close
```

6.4.7.1 Initialization

The following sample code demonstrates the HTTP library initialization. The sample code shows that the initialization includes opening both the HTTP library and the NetLib library, setting all the callback for calling NetLib's TCP socket functions, and setting the application execution environments through the three global variables structures—struct `HS_HTTPLibNetInfo`, struct `HS_HTTPLibPeer`, and struct `HS_HTTPLibAppInfo`.

The application's SSL certificate, proxy connection, and connection timeouts are also initialized.

```
void HTTPLibInitialize(void)
{
err = HS_HTTPLibOpen(gRefNum);

   /* gPeer */
   MemSet(&gPeer, sizeof(HS_HTTPLibPeer));0
   gPeer.HS_HTTPLibPeerTCPOpen = &PrvTCPOpen;
   gPeer.HS_HTTPLibPeerTCPClose = &PrvTCPClose;
   gPeer.HS_HTTPLibPeerTCPIsConnected = &PrvTCPIsConnected;
   gPeer.HS_HTTPLibPeerTCPConnect = &PrvTCPConnect;
   gPeer.HS_HTTPLibPeerTCPRead = &PrvTCPRead;
   gPeer.HS_HTTPLibPeerTCPWrite = &PrvTCPWrite;
   gPeer.HS_HTTPLibPeerTCPCanReadWrite = &PrvTCPCanReadWrite;

   gAppInfo.maxSockets = 3;
   gAppInfo.isForeground = true;
   gAppInfo.cookieMaxJarSize = (UInt16)300 * (UInt16)1024;

   PrvPeerTCPInitialize();
   gLibHandle = HS_HTTPLibInitialize(gRefNum, &gAppInfo,    &gNetLibInfo,
&gPeer);
```

```
   /* set callbacks */
   //HS_HTTPLibSetSSLServerCertConfirmProc(gRefNum,    gLibHandle,
&test_confirm_cb,    (HS_HTTPLibOpaque)gLibHandle);
   //HS_HTTPLibSetTunnelingCallback(gRefNum, gLibHandle,
   &PrvTunnelingCallback, NULL);

   /* set timeout time */
   HS_HTTPLibSetConnectTimeOut(gRefNum, gLibHandle, -1);
   HS_HTTPLibSetReqTimeOut(gRefNum, gLibHandle, -1);
   HS_HTTPLibSetRspTimeOut(gRefNum, gLibHandle, 10 * 1000);

   /* set proxy info if used*/
   //HS_HTTPLibSetProxy(gRefNum, gLibHandle, ProxyHost,
   StrLen(ProxyHost), ProxyPort, ProxyPort, NoProxyHost, 0);
   //HS_HTTPLibSetUseProxy(gRefNum, gLibHandle, true);
}
.................
.................

/* Wrappers for NetLib callbacks */
void PrvPeerTCPInitialize( ) {.... NetLibOpen( .... }

Int32 PrvTCPOpen( ) {..... NetLibSocketOpen( .... }

void PrvTCPClose( ) { .... NetLibSocketClose( .... }

Int32 PrvTCPIsConnected( ) {.... NetLibSelect( .... }

Int32 PrvTCPConnect( ) {.... NetLibSocketConnect( .... }

Int32 PrvTCPRead( ) {..... NetLibReceive( .... }

Int32 PrvTCPWrite( ) {.... NetLibSend( .....}

Int32 PrvTCPCanReadWrite( ) {....  NetLibSelect( .... }
```

6.4.7.2 Finalization

The following sample code demonstrates the HTTP library finalization. The finalization appears as a short sequence of freeing memory and closing the NetLib and HTTP libraries.

```
void HTTPLibInitialize(void)
{
HS_HTTPLibFinalize(gRefNum, gLibHandle);
PrvPeerFinalize();
HS_HTTPLibClose(gRefNum, &count);
}

void PrvPeerTCPFinalize( ) { .... NetLibIFDown( .... }
```

6.4.7.3 Processing Loop

The HTTP library does not contain an internal processing loop, so applications should implement a processing loop to send and receive incoming stream data. Applications can implement the processing loop as a state machine that models the sequence of communication between the client and the server through the HTTP protocol. The following example shows how an HTTP sequence is modeled into a state machine.

Common HTTP accesses follow this sequence pattern:

1. Generate an HTTP request and send the request.
2. Read the response header.
3. Read the content of the data stream.
4. Continue reading content until the end or until an error occurs.
5. Close the stream connection.

Applications can define the preceding state transitions into the state machine to process streaming data.

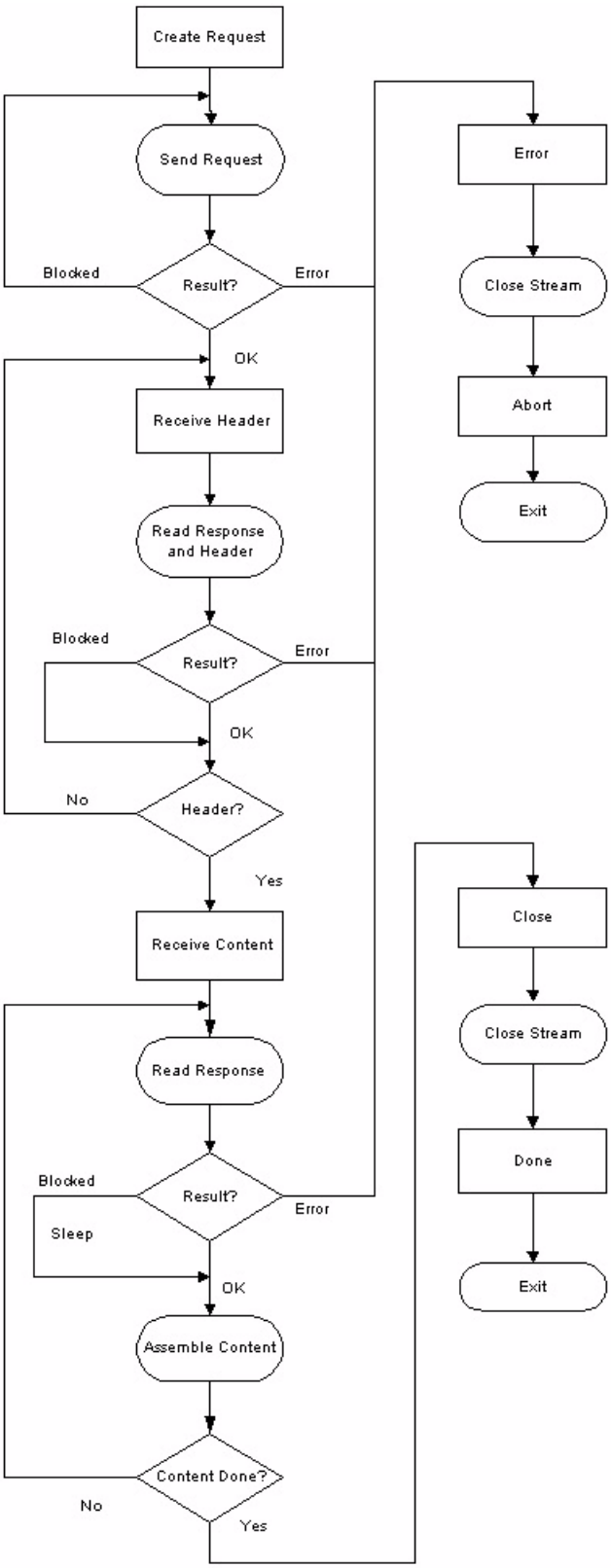
Applications can also implement the following state transition pattern for successful HTTP accesses in the state machine:

1. Initiate the state machine with a request creation with the state `kDownloadState_Request`.
2. Through the state machine, send a request out and switch the state to `kDownloadState_ReceiveHeader`.
3. Read the initial response and header and then transition to the state `kDownloadState_ReceiveContent`.
4. Continue to loop and read the response until the data is complete, and then transition to the state `kDownloadState_Close`.
5. Close the stream and switch to the state `kDownloadState_Done`.
6. Exit with the condition `httpErrorOK`.

An error can occur during the state `kDownloadState_Request_xyz` or `kDownloadState_Receive_xyz` when switching to the state `kDownloadState_Error`, `kDownloadState_Cancel`, or `kDownloadState_Abort`. These states trigger a loop exit, and the appropriate error conditions are recorded.

The sample code in the Palm sample application call `HTTPLibTest` includes code for an application's processing loop to send an HTTP request and receive the response through a state machine. This processing loop can be modified to adapt to other applications. Refer to the `HTTPLibTest` project.

The flowchart on the following page illustrates how an HTTP sequence is modeled into a state machine.


Create Request
Send Request
Blocked
Result?
Error
OK
Receive Header
Read Response and Header
Blocked
Result?
Error
OK
No
Header?
Yes
Receive Content
Read Response
Blocked
Result?
Error
Sleep
OK
Assemble Content
Content Done?
No
Yes
Error
Close Stream
Abort
Exit
Close
Close Stream
Done
Exit

6.5 Net Services API

Available on:

- LifeDrive™ mobile manager
- Tungsten™ C and Palm® T|X handhelds

This section provides reference information for the Net Services API. You can use the functions in this API to check a handheld's radio hardware, to add or change network user profiles, and to implement other 802.11 network service tasks.

The NetServices library is available only on handhelds equipped with Wi-Fi (wireless fidelity), either through built-in hardware resident on the device or through a Wi-Fi add-on accessory.

The Net Services API is declared in the header file `PalmNetServices.h`. The Net Services API also uses data structures declared in the header file `PalmWiFiCommon.h`.

6.5.1 Overview of the Net Services feature

Handhelds that include Wi-Fi functionality include a panel that allows users to add network profiles in order to connect to different 802.11 Internet access points. Users can choose different profile names, SSIDs (service set identifiers), encryption methods, and other profile characteristics based on the requirements of the access points and on their own personal preferences.

Using the Net Services API, you can create your own panel to allow users to create profiles, to specify encryption methods, and to connect to access points.

NOTE: The Net Services API does not provide a method to add profiles to the existing Wi-Fi panel. If you create a Net Services application, you must design your own panel to display and create profiles, connect to access points, and so forth. Furthermore, if you plan to add, delete, or modify your own separate set of profiles, you should design your own panel because the default Wi-Fi panel may overwrite the profiles you create and modify. If you simply want to replace the default Wi-Fi panel, set the creator ID and panel type to that of the default Wi-Fi panel.

This page intentionally blank

CHAPTER 7

7. HTML Library

This chapter provides reference information on the HTML library, including its usage model, architecture, and features.

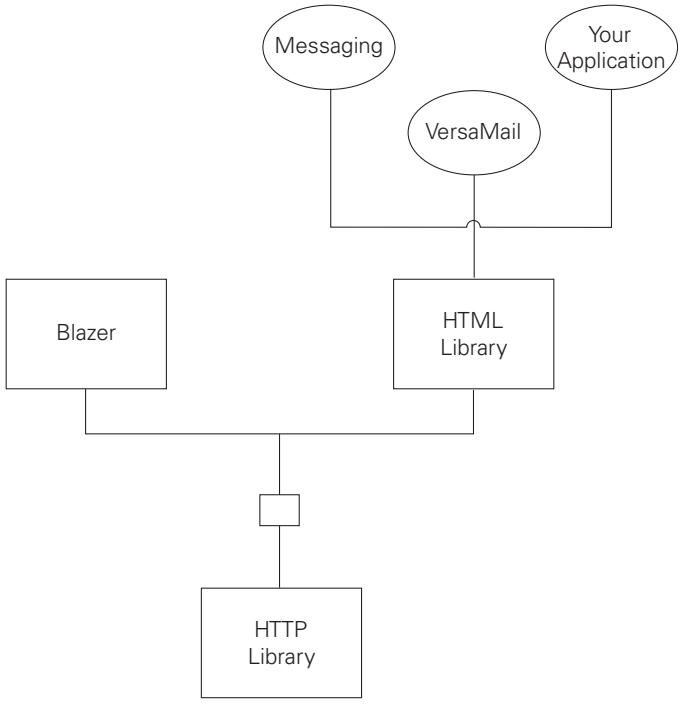
Available on:

- Centro™ and Treo™ smartphones
- LifeDrive™ mobile manager
- Palm® T|X™ handheld

The HTML library is a shared library, added to enable applications to render data. It is only available on the devices listed at the beginning of this chapter. The HTML library API is declared in the header file `HtmlLib68K.h`. For more information, refer to the *Palm API Guide, Palm OS Platform* at **http://pdn.palm.com**.

7.1 Architecture

The HTML library is provided at the Palm OS® library level. It is used by VersaMail® and Messaging applications. The following figure shows the architecture of the HTML library.



7.2 Usage model

When using the HTML library, keep the following details in mind:

- Only one application should be using the HTML library at a time. When the browser is running, other applications cannot use the HTML library. Conversely, when another application is using the HTML library, the browser should not be running.
- The HTML library does not support frames, iframes, forms, and JavaScript.

The following steps demonstrate the usage model of a typical application using the HTML library. For more information, refer to the sample code `HtmlLibTest68K` in the Sample Code section of the SDK.

1. Find and load the HTML library:

```
Err SysLibLoad (UInt32 libType, UInt32 libCreator, UInt16 *refNumP)
```

2. Open the HTML library:

```
Err HTMLLibOpen (UInt16 refnum)
```

3. Initialize the HTML library:

```
MemHandle HtmlLibInitialize (UInt16 refnum, RectangleType bounds,
Err *errP)
```

4. Set up callback functions as needed:

- `typedef void HtmlLibLinkSelCallback (MemHandle htmlLibH, Char *url, void *cbData)`
- `typedef Boolean HtmlLibScanPhoneNumberCallback (MemHandle htmlLibH, Char *buffer, Int32 length, Int32 *patternStart, Int32 *patternEnd)`
- `typedef void HtmlLibInclusionCallback (MemHandle htmlLibH, Char *url, void *cbData)`
- `typedef void HtmlLibSubmitFormCallback (MemHandle htmlLibH, Char *url, HtmlSubmitFormMethod method, Char *query, void *cbData)`

5. Create content data objects as needed:

```
MemHandle HtmlLibCreateContentObject (UInt16 refnum, MemHandle htmlLibH)
```

6. Add image/text data:

- `Err HtmlLibAddTextData (UInt16 refnum, MemHandle contentH, Char *url, Char *mimeType, Char *charset, void *data, Int32 dataLen)`
- `Err HtmlLibAddImageData (UInt16 refnum, MemHandle contentH, Char *url, Char *mimeType, void *data, Int32 dataLen)`

7. Render as needed:

```
void HtmlLibRenderData (UInt16 refnum, MemHandle contentH)
```

8. Destroy content data objects:

```
void HtmlLibDestroyContentObject (UInt16 refnum, MemHandle contentH)
```

NOTE: Make sure to call `HtmlLibAbortRenderData` before destroying a content object.

9. Finalize the HTML library:

```
void HtmlLibFinalize (UInt16 refnum, MemHandle htmlLibH)
```

10. Close the HTML library:

```
Err HtmlLibClose (UInt16 refnum)
```

11. Unload the HTML library:

```
SysLibRemove()
```

7.3 Image rendering

For HMTL content that includes images, the application can provide the image content to the HTML library in one of the following methods:

1. Provide image content(s) with `InclusionCallback`. To do so, call `HtmlLibAddImageData` when `HtmlLibrary` calls `InclusionCallback`.
2. Provide image content(s) by calling `HtmlLibAddImageData` before `HtmlLibRenderData`. This method is particularly useful if the HTML content contains a lot of images and you do not want the application to show "no image" icons. This way, the image content(s) are provided to the HTML library before the page starts rendering.

7.4 Debugging

If you have trouble rendering data, use the following steps to troubleshoot:

1. Try sending the HTML code you are trying to render to the Blazer® web browser programmatically using the Exchange Manager or a filestream. If it crashes the Blazer application, there is a bug in the HTML Renderer.
2. Modify the HTML library sample code to render the same HTML.
3. Recompile. If it crashes the sample code but not the Blazer application, there is probably a bug in the HTML library.
4. If it works in both the Blazer application and the sample code, but crashes your application, it is something that you are doing differently from the sample code, and you should investigate this difference.
5. Try rendering your HTML from the SD card using the Blazer web browser. First, make sure that the browser is in optimized mode. To render your HTML from the SD card, use the following path:

   ```
   file:///<OPTIONAL_SDCARDNAME>/<OPTIONAL_PATH>/filename
   ```

   If the content renders correctly, there may be a bug in your application.

NOTE: Link walking is only supported in the table unrolled version of the HTML library. If you use the header `<META NAME="HandheldFriendly" content="True">` then you are not in optimized mode, and link walking will not work.

For more information on general debugging issues, refer to **Chapter 13**.

This page intentionally blank

CHAPTER 8

8. Telephony

This section provides reference material for the Telephony APIs in the Palm OS® SDK.

8.1 Overview of the Telephony API libraries

Available on:

- Centro™ and Treo™ smartphones

The `HsPhone.h` header file provided in the Palm OS SDK contains all the public files referenced in this section, including all constants, structure definitions, and function prototypes. For more details, refer to the *Palm API Guide, Palm OS Platform* at **http://pdn.palm.com**.

The telephony libraries include the following categories:

Phone library category	Description
SMS	Functions that apply to sending, receiving, and managing SMS messages. Declared in `HsPhoneSMS.h`.
GSM	Functions that apply to GSM products with some functions that apply to both GSM and CDMA products. Declared in `HsPhoneGSM.h`.
CDMA	Functions that apply only to the CDMA product. Declared in `HsPhoneGSM.h`.

The telephony libraries depend on a set of mostly common structures, types, and enumerations. You can also look at the latest version of the header files in the Palm OS SDK for the most up-to-date definitions.

NOTE: The Palm Telephony library does not support third-party applications that control phone calls directly. The supported method is to use the helper application described in **Section 8.2.3 on page 107**.

Because the telephony libraries contain so many different enumerations, structures, and types, the telephony header files are divided into categories so that they are easier to understand. These categories are defined in the following table.

Category	Description
Library	General Palm OS library functions and enumerations related to the telephony libraries. Declared in `HsPhoneLibrary.h`.
Network	Cellular network types and functions. Declared in `HsPhoneNetwork.h`. Developers should have a basic knowledge of how a cellular network behaves. The following concepts should be familiar to you if you want to use the Telephony Network APIs: ■ Current carrier versus home carrier ■ How to select the current carrier when more than one is available ■ Phone registration process with the wireless network ■ Interactions with a SIM card ■ Voicemail box interaction from the network
Audio	Specific audio definition: ringing profile and slider switch.
Events	Telephony events sent to an application that registers for them. Its main enum is `PhnEventCode`, which contains the notification received by a registered application. Refer to the sample code in the Palm OS SDK.
Security	Password Type enumerations.
IOTA	Internet Over The Air enumerations.
Misc	Other functions. Declared in `HsPhoneMisc.h`.

NOTE: The telephony libraries support the specific requirements of the Palm OS-based smartphones. Palm smartphones do not support Palm OS telephony *functions*. If you want to use the telephony functions on the Palm smartphone, you must use the Palm Telephony library functions.

8.1.1 CDMA and GSM libraries

Beginning with the Treo 680 with Phone Application 3.0, the `GSMLibrary` has been replaced by the `PhoneInterfaceLibrary`, which is used for both GSM and CDMA. The `PhoneInterfaceLibrary` supports the same APIs and functionality as the GSM library.

The`PhoneInterfaceLibrary` applies to the Palm Treo 680, 755p and later Palm smartphones. It supports the same APIs for both GSM and CDMA

Library	Constant (#define)	Value	Description
Basic definition	`phnLibDbType`	`'libr'`	Database type ID
CDMA and GSM			
	`phnLibCDMADbCreator`	`'PIL!'`	CDMA library creator ID
	`phnLibCDMADbName`	`''Phone Library''`	CDMA library Database name
	`phnLibCDMAName`	`''Phone Library''`	CDMA library name

For Treo 600, Treo 650, Treo 700p smartphones, the CDMA and GSM Telephony APIs are almost the same. However, because the actual radio architecture differs, Palm has two different libraries as defined in the following table. It is important to use the correct library for the correct radio architecture when using the Telephony APIs.

Library	Constant (#define)	Value	Description
Basic definition	`phnLibDbType`	`'libr'`	Database type ID
CDMA			
	`phnLibCDMADbCreator`	`'HsCL'`	CDMA library creator ID
	`phnLibCDMADbName`	`''Phone Library''`	CDMA library Database name
	`phnLibCDMAName`	`''PhoneLib.prc''`	CDMA library name
GSM			
	`phnLibGSMDbCreator`	`'GSM!'`	GSM library creator ID
	`phnLibGSMDbName`	`''Phone Library''`	GSM library Database name
	`phnLibGSMName`	`''GSMLibrary.lib''`	GSM library name

8.1.2 Using indicators

Indicators are available for all Palm OS-based smartphones.

8.1.2.1 GSM Connected indicator

On a Palm smartphone, if you want your application to determine whether the smartphone is registered with a network, use the Telephony library function `PhnLibRegistered`.

You may want your application to update the GSM Connected indicator when the following phone events occur:

- `phnEvtRegistration`
- `phnEvtError`
- `phnEvtIndication`
  - `indicationNetworkAvailable`
  - `indicationStartingRadio`
  - `indicationResettingRadio`
  - `indicationPoweringOffRadio`

NOTE: To convey whether the phone is connected to a network, the simplest approach is to use the Palm system Signal gadget. (See **Section 14.2.4 on page 280** for more information.) It appears when the phone is connected to a network, and does not appear when the phone is not connected.

8.1.2.2 Carrier's name indicator

To retrieve the current carrier's (or operator's) name, use the Telephony library function `PhnLibCurrentOperator`.

You may want your application to update the Operator's Name indicator when the following phone events occur:

- `phnEvtRegistration`
- `phnEvtError`
- `phnEvtIndication`
  - `indicationNetworkAvailable`
  - `indicationStartingRadio`
  - `indicationResettingRadio`
  - `indicationPoweringOffRadio`

8.1.2.3 Voicemail indicator

To retrieve the current status of Voicemail, use the Telephony library function `PhnLibBoxInformation`.

You should update the Voicemail indicator in your application when a `phnEvtVoiceMail` event occurs. `phnEvtVoiceMail` events occur when new voicemail messages are received and when users clear their voicemail.

8.1.3 Using PhnLibRegister

As a general best practice, `PhnLibRegister` should not be called when an application is handling `PhnLib` notifications. This should not be done because by the time `PhnLib` notifications are received, the `PhnLibRegister` function has already been called.

There is no need to register specifically for `phnEvtRegistration` because if you register any service, for example `phnServiceAll`, you will automatically receive `phnEvtRegistration`.

When an application re-registers for `phnEvtRegistration` notifications in its notification handler, the current notification loop is halted, and the Phone Application does not receive them. This may cause problems with the Phone Application.

Currently, no error is returned when an application re-registers for notifications.

8.1.4 Getting the Cell ID on Centro™ smartphones

For Centro™ GSM and CDMA smartphones, APIs are available that will allow developers to retrieve the Cell ID (also known as the Cell of Origin, or COO) and local area code (LAC) of a phone.

Technically, COO positioning allows a phone to be located based on the closest base station or tower, and regards this location to be the location of the caller. Applications can use these general positions to provide location based services (LBS).

8.1.4.1 Centro™ GSM smartphones

The API is in the header `Incs\68k\libraries\telephony\HsPhoneGSM.h`.

The prototype is as follows:

```
Err PhnLibGetGSMCellInfo (UInt refNum, UInt32 *pLac, UInt32 *pCellId)
```

- `pCellId` - returns the Cell ID (or COO)
- `pLac` - returns the LAC of the caller

This API is GSM specific and is only supported on Centro GSM smartphones. It is not supported on earlier devices.

8.1.4.2 Centro™ CDMA smartphones

The Centro CDMA smartphone uses an API in the header file `Incs\68k\libraries\telephony\HsPhoneMisc.h`.

The prototype of the API is as follows:

```
Err PhnLibGetBaseStationIdInfo( UInt16 *base_id, UInt16 *sid, UInt16 *nid );
```

- `base_id` - is the Base Station ID
- `sid` - is the System ID
- `nid` - is the Network ID

Use the test file CDMAStatus.prc to obtain the values returned from the API. This is available at the PDN Knowledge Base. It displays values for the Cell id (base station id), system id, and network id.

8.1.5 PhnLibGetMMSUAString()

The PhnLibGetMMSUAString() API is used with Centro Verizon devices only.

It is a new API in the header file Incs\68k\libraries\telephony\HsPhoneMisc.h for the Verizon MMS application. It returns the MMS User Agent string.

The syntax is as follows:

```
Err PhnLibGetMMSUAString( UInt16 refNum, char *pUaStr, UInt32 buf_len );
```

- **pUaStr -** The buffer in which the UA String will be written.
- **buf_len -** The length of the buffer so that Phn Lib can ensure that the buffer size is sufficient to return the UA string. (The current minimum buffer length required is 32 bytes.)

8.1.6 PhnMsgBoxDataType

The field PhnMsgBoxDataType is specific to Centro devices.

A new priority field has been added to the structure PhnMsgBoxDataType in Incs\common\libraries\telephony\HsPhoneNetworkTypes.h. This indicates voicemail priority. The syntax is as follows:

```
typedef UInt8 PhnMsgBoxPriorityType;
typedef enum
{
  kSmsPriorityNormal,
  kSmsPriorityInteractive,
  kSmsPriorityUrgent,
  kSmsPriorityEmergency,
  kSmsPriorityLast
}
PhnMsgBoxPriorityEnum;

typedef struct
  {
    Boolean         indicatorOn;    /**<        */
    PhnMsgBoxType   type;       /**<        */
    Int16           messageCount;   /**<        */
    Int16           lineNumber;     /**<        */
    PhnMsgBoxPriorityType           priority; /**< voicemail priority -
example: normal, urgent etc. */
  }
PhnMsgBoxDataType;
```

8.2 Phone Application

8.2.1 Phone Application 2.5

Available on:

- Centro™ smartphones (CDMA only)
- Treo™ 755p smartphones
- Treo™ 650 smartphones

Phone Application version 2.5 is written in 68K.

Phone App 2.5

For more information on Phone Application views, see the User Manuals at the Palm Customer Service and Support website, **http://www.palm.com/us/support/**.

8.2.2 Phone Application 3.0 and later

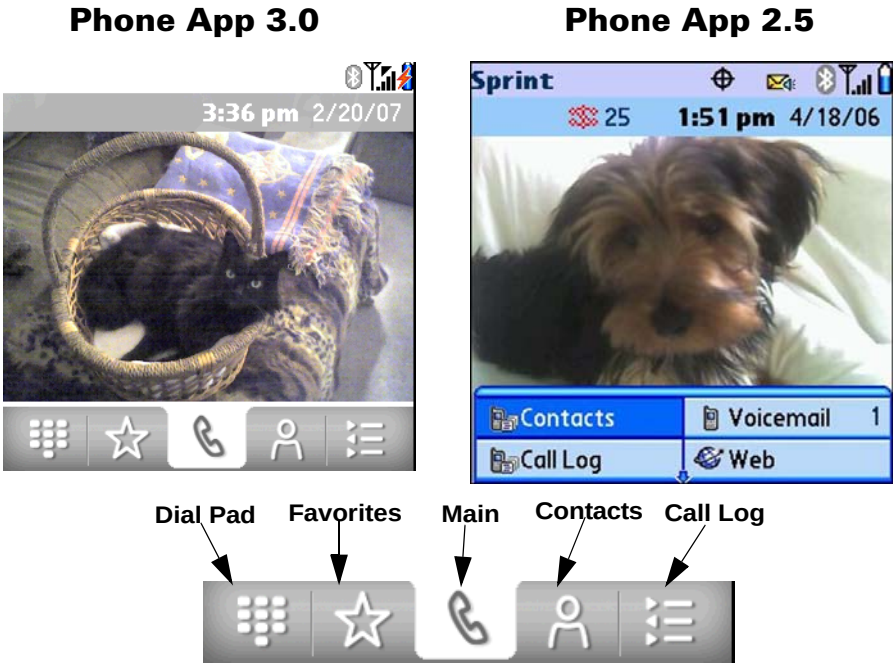
Available on:

- Centro™ smartphones (GSM only)
- Treo™ 680 smartphones

Treo 680 smartphones include Phone Application version 3.0. Centro smartphones for AT&T and Rest of World carriers include Phone Application version 3.5.

These versions include a redesigned user interface that integrates the Main, Dial Pad, Favorites, Contacts, and Call Log views into a more intuitive user experience. Most changes to the application are transparent to developers, but a few items listed in this section are different.

The redesigned Phone Application 3.0 interface is shown here in comparison to Phone Application 2.5:



The Phone Application version 3.0 has been rewritten in ARM, where previously it was written in 68K. One difference for developers to note is that ARM uses little endian sequencing method, while 68K uses big endian sequencing.

NOTE: All features in this chapter will work with Phone Application 3.0 except where noted.

For more information on Phone Application views, see the User Manuals at the Palm Customer Service and Support website, **http://www.palm.com/us/support/**.

8.2.2.1 Overlapping gadgets

In Treo 680, support has been added to `PmSysGadgetLibrary` for overlapping gadgets. If a gadget is defined with the same boundaries as an existing gadget, it will be given a lower priority than the previously defined gadget, and will only be displayed if the other gadget is not visible.

8.2.2.2 Call in Progress gadget

In Treo 680, the overlapping gadget feature is used to display the "Call in Progress" message above the carrier status gadget. This message indicates that a call is in progress. This gadget is included in the five main Phone Application views: Main, Dial Pad, Favorites, Contacts, and Call Log.

8.2.3 Launching the Phone application in a specific view

This section details how to launch the Phone application on Palm OS-based smartphones in a specific view. For more information, refer to the `Launch Commands` sample code in the Palm OS SDK.

8.2.3.1 Required headers

The headers required to launch the Phone application in a specific view are as follows:

- `Common/System/HsAppLaunchCmd.h`

  This header file includes Phone application launch commands and corresponding launch command parameter structures.

- `Common/System/HsCreators.h`

  This header file includes the Phone application creator type.

8.2.3.2 Launching the Phone application in Call Log view

To launch the Phone application in the Call Log view, use the `phoneAppLaunchCmdViewHistory` launch command as follows:

```
DmGetNextDatabaseByTypeCreator ( true,
                                 &searchState,
                                 sysFileTApplication,
                                 hsFileCPhone,
                                 true,
                                 &cardNo,
                                 &dbID);

err = SysUIAppSwitch( cardNo,
                      dbID,
                      phoneAppLaunchCmdViewHistory,
                      NULL /*paramsP*/);
```

8.2.4 Launching the Phone application in Dial Pad view

When launching the Phone application in Dial Pad view, you can do one of the following:

- Launch in Dial Pad view with no number filled in.
- Launch in Dial Pad view and prefill the number field in the Dial Pad view with a specified number without automatically dialing it.
- Launch in Dial Pad view and automatically dial a specified number.

8.2.4.1 Launching without a phone number

Use the following code as reference to launch the Phone application in Dial Pad view without pre-filling the number field:

```
DmGetNextDatabaseByTypeCreator ( true,
                                 &searchState,
                                 sysFileTApplication,
                                 hsFileCPhone,
                                 true,
                                 &cardNo,
                                 &dbID);

err = SysUIAppSwitch( cardNo,
                      dbID,
                      phoneAppLaunchCmdViewKeypad,
                      NULL /*paramsP*/);
```

8.2.4.2 Launching with the number field prefilled

NOTE: This feature will not work for Centro GSM smartphones or Treo 680 smartphones with Phone Application 3.0.

Use the following code as reference to launch the Phone application in Dial Pad view and prefill the number field:

```
PhoneAppLaunchCmdDialPtrparamsP = NULL;
UInt16 size = sizeof(PhoneAppLaunchCmdDialType);
Char* numberP = < PHONE_NUMBER_TO_PREFILL_FIELD_WITH>;

// Setup a parameter block so the Phone application pre-fills
// a phone number in the Dial Pad number field

if (numberP)
  {
size += StrLen(numberP) + 1;
  }

paramsP = MemPtrNew (size);
MemSet (paramsP, size, 0);
```

```
paramsP->version = 1;
paramsP->failLaunchCreator = <YOUR_APP_CREATOR>;
if (numberP)
    {
    paramsP->number = MemPtrNew (StrLen(numberP) + 1);
    StrCopy(paramsP->number, numberP);
    MemPtrSetOwner (paramsP->number, 0);
    }


DmGetNextDatabaseByTypeCreator ( true,
                                 &searchState,
                                 sysFileTApplication,
                                 hsFileCPhone,
                                 true,
                                 &cardNo,
                                 &dbID);

err = SysUIAppSwitch( cardNo,
                      dbID,
                      phoneAppLaunchCmdViewKeypad,
                      paramsP);
```

8.2.4.3 Launching and automatically dialing a phone number

Use the following code as reference to launch the Phone application in Dial Pad view and automatically dial a phone number:

```
PhoneAppLaunchCmdDialPtr paramsP = NULL;
UInt16 size = sizeof(PhoneAppLaunchCmdDialType);
Char* numberP = <THE_PHONE_NUMBER_YOU_WANT_AUTO_DIALED>;

// Set up a parameter block so the Phone application
// automatically dials a phone number

if (numberP)
  {
size += StrLen(numberP) + 1;
  }

paramsP = MemPtrNew (size);
MemSet (paramsP, size, 0);

paramsP->version = 1;
paramsP->failLaunchCreator = <YOUR_APP_CREATOR>;
if (numberP)
  {
paramsP->number = MemPtrNew (StrLen(numberP) + 1);
    StrCopy(paramsP->number, numberP);
    MemPtrSetOwner (paramsP->number);
  }

MemPtrSetOwner (paramsP->number, 0);
```

```
DmGetNextDatabaseByTypeCreator ( true,
                                 &searchState,
                                 sysFileTApplication,
                                 hsFileCPhone,
                                 true,
                                 &cardNo,
                                 &dbID);

err = SysUIAppSwitch( cardNo,
                      dbID,
                      phoneAppLaunchCmdDial,
                      paramsP);
```

8.2.5 Launching the Phone application in the Favorites view

To launch the Phone application in the Favorites view, use the `phoneAppLaunchCmdViewSpeed` launch command as follows:

```
DmGetNextDatabaseByTypeCreator ( true,
                                 &searchState,
                                 sysFileTApplication,
                                 hsFileCPhone,
                                 true,
                                 &cardNo,
                                 &dbID);

err = SysUIAppSwitch( cardNo,
                      dbID,
                      phoneAppLaunchCmdViewSpeed,
                      NULL /*paramsP*/);
```

8.3 Launching the Contacts application with the New Contact window open

On Palm OS-based smartphones, you can launch the Contacts application with the New Contact window open.

The headers required for launching the Contacts application with the New Contact window open are as follows:

- `Common/System/HsAppLaunchCmd.h`

  This header file includes Contact application launch commands and corresponding launch command parameter structures.

- `Common/System/palmOneCreators.h`

  This header file contains the Contacts application creator type.

To launch the Contacts application with the New Contact window open, use the `addrAppNotificationCreateNewRecord` launch command as follows:

```
notifyParamP = MemPtrNew(sizeof(SysNotifyParamType));

MemSet(notifyParamP, notifyParamSize, 0);

   notifyParamP->notifyType = addrAppNotificationCreateNewRecord;
   notifyParamP->broadcaster = <YOUR_APP_CREATOR>;
   notifyParamP->notifyDetailsP = NULL;
   notifyParamP->handled = false;

DmGetNextDatabaseByTypeCreator( true,
                                &searchState,
                                sysFileTApplication,
                                kpalmOneCreatorIDContacts,
                                true,
                                &cardNo,
                                &dbID);

err = SysUIAppSwitch(  cardNo,
                       dbID,
                       sysAppLaunchCmdNotify,
                       notifyParamP);
```

8.4 EvDO on Palm® smartphones

This section applies to:

- Centro™ smartphones
- Treo™ 700p and Treo™ 755p smartphones

Treo 700p, Treo 755p, and Centro smartphones are capable of connecting to 1xEvDO networks. To develop applications for Treo 700p smartphones, keep the following points in mind.

During a data connection session on a 1xEvDO network, applications should give priority to an incoming voice call and allow it to interrupt the data session. Unlike Treo 650 CDMA smartphones, which only support 1xRTT network connections, on the Treo 700p, 755p and Centro smartphones, it is possible for incoming calls to be received even when the data network is already connected and not dormant, as opposed to going directly to voicemail. Wireless applications should take this into account and perform the necessary work to clean up or record the states of the data transfer.

Upon voice call disconnect, applications that require an always-on data session could re-establish the data session and continue transferring data. In order to do this properly, applications should register for the following phone events:

- `phnEvtStartIncomingCall` - This phone event is broadcasted to registered applications when there is an incoming phone call
- `phnEvtDisconnectInd` - This phone event is broadcasted to registered applications when the voice call is disconnected
- `phnEvtDisconnectConf` - This phone event is broadcasted to registered applications when the voice or data call in conference mode (3-way calling) is disconnected

8.4.1 Detecting EvDO vs. 1xRTT

Because the Treo 700p, 755p, and Centro smartphones are capable of both EvDO and 1xRTT connections, it may be useful for applications to determine when the device is using the EvDO connection and when it is using the 1xRTT connection.

When connecting to an EvDO network, the user is shown the icons displayed in the following table.

Carrier	Icon
Sprint	
Verizon	
Treo 700p Rest of World* carriers	

To be notified of which connection is being used, the application should register for `phnEvtDataSessionStatus`. Once it is registered for the `phnEvtDataSessionStatus` event, use its data field to obtain `dataSesstionStatus`'s session type. Then check against the `PhnDataServiceType` enumerations for EvDO.

NOTE: Please note the extra 't' in the spelling of `dataSesstionStatus`.

* For the current list of carriers included in the Treo 700p ROW release, visit the Palm Developer Network (PDN) at **http://pdn.palm.com** and navigate to the Treo 700p device page.

Here is an example of detecting the communication type:

```
switch(pEvt -> eventType)

{

    ...

   case phnEvtDataSessionStatus:

             if (pEvt -> data.dataSesstionStatus.sessionType == phnDataService1xEVDO)

                  // Data session has just switched to EVDO


             if (pEvt -> data.dataSesstionStatus.sessionType == phnDataService1xRTT)

                 // Data session has just switched to 1xRTT

    ...

}
```

For more details of the structures referenced in the previous example, refer to the *Palm API Guide, Palm OS Platform* and to the `HsPhoneEvent.h` and `HsPhoneNetworkTypes.h` header files.

8.4.2 Troubleshooting incoming voice calls

If you application is not handling incoming voice calls correctly, you may see the following symptoms:

- The voice call goes to voicemail, but the smartphone briefly displays an incoming call dialog, then displays a missed call dialog.

  This happens because the incoming phone event is already coming through, but because the wireless application does not give up the handle on time, it is already too late for the Phone application to accept and connect the call.

- The wireless application freezes while waiting for data.

  This could happen if the wireless application is using a blocking socket with an infinite timeout, or does not abort when data stops transferring. When an incoming voice calls interrupts the data session, no more data can be sent or received. Applications must handle this appropriately by cleaning up, maintaining the state, registering for phone call disconnect events, and re-initializing variables.

This page intentionally blank

CHAPTER 9

9. SMS

This chapter describes the SMS library usage model.

9.1 What is the difference between SMS and NBS?

SMS stands for Short Message Service. This service allows short text messages to be sent and received by your mobile phone. An alert is shown every time an SMS message is received.

NBS stands for Narrow Band Socket. NBS is a special kind of SMS message. If an SMS message contains `//SCK <code>` it is treated as an NBS message.

NBS messages on Palm devices are treated in a silent manner. The message is invisible to the user. No alert is shown to the user when an NBS message is received, and NBS messages can be silently deleted.

9.2 SMS library

Available on:

- Centro™ and Treo™ smartphones

The SMS Messaging application is the main interface to the SMS Database and SMS features on a device. In addition to the SMS application, you can create your own applications to receive, send, and manage SMS messages.

SMS is a useful way to wake up a device remotely and trigger specific actions that are defined by your application. For example, an SMS could trigger an email application to retrieve new email from a corporate server.

Technically, the SMS library is part of the Telephony library, but it is logically a separate unit. The SMS library relies on functionality from the Telephony library and uses similar methods.

This chapter describes some of the key features of the SMS library including:

- Sending messages
- Receiving messages
- Encoding
- Segmenting and reassembling messages
- Storing messages in a database

9.3 What is SMS?

The point-to-point SMS provides a means of sending short messages to and from a phone. SMS is implemented using a service center that acts as a store and forwarding center for short messages. The SMS library architecture described here generally applies to CDMA.

Two point-to-point services are defined in the SMS specification:

1. **Mobile originated messages -** These messages are transported from a phone to a service center. They may be destined for other mobile users or an email gateway.
2. **Mobile terminated messages -** These messages are transported from the service center to a phone. They may have originated from another phone or from a variety of other sources, such as an email application or a website.

A single message sent on the SMS network on CDMA is limited to 160 characters. Longer messages must be segmented. For message segmentation on GSM, see **Section 9.5.4 on page 120**.

9.4 Why use the SMS library?

The SMS library enables the sending and receiving of short messages from a smartphone to another SMS-enabled recipient.

On Palm smartphones, SMS messages can be sent much more quickly than messages with a PPP connection. This is because SMS does not require an initial connection to an ISP. Connecting to an ISP can take up to 10 to 15 seconds, including modem negotiation time and user authentication.

NOTE: When sending an SMS message, the current SMS library architecture has a three- to four-second delay before finishing the sending API call. This is due to radio architecture latency.

Cost is another consideration for using SMS. A typical U.S. service provider charges 15 cents a minute for data service, but only 10 cents for an SMS.

Additionally, SMS messages can be sent directly to a device. They do not require a client application to dial in to the network to check for new email; instead, the message is sent directly to the client application. The client can even receive an SMS message while a voice or data call is in progress.

9.5 Understanding the SMS library

The SMS library takes care of the low-level details of communicating with a device. All incoming messages are indicated to a registered application using launch codes, and all successful or unsuccessful message transmissions are indicated with launch codes.

The SMS library handles the following functions for SMS messages:

- Sending
- Receiving
- Encoding*

* This encoding function refers only to character encoding. Currently, only the GSM alphabet is supported. The GSM alphabet is different from the Palm OS® alphabet. Additionally, encryption and compression of messages are not supported in the current version of the SMS library.

NOTE: The current version of the SMS library handles text messages for both the GSM and CDMA versions of devices. All messages handled by the library are assumed to be text and are handled appropriately.

The `PhoneInterfaceLibrary` (previously, the GSM library) can also handle binary messages, but the CDMA library cannot. If a client application wants to send binary data on CDMA, it must encode and decode the data properly.

Also, the CDMA version supports only 7-bit ASCII characters, 160 characters maximum*, no segmentation*.

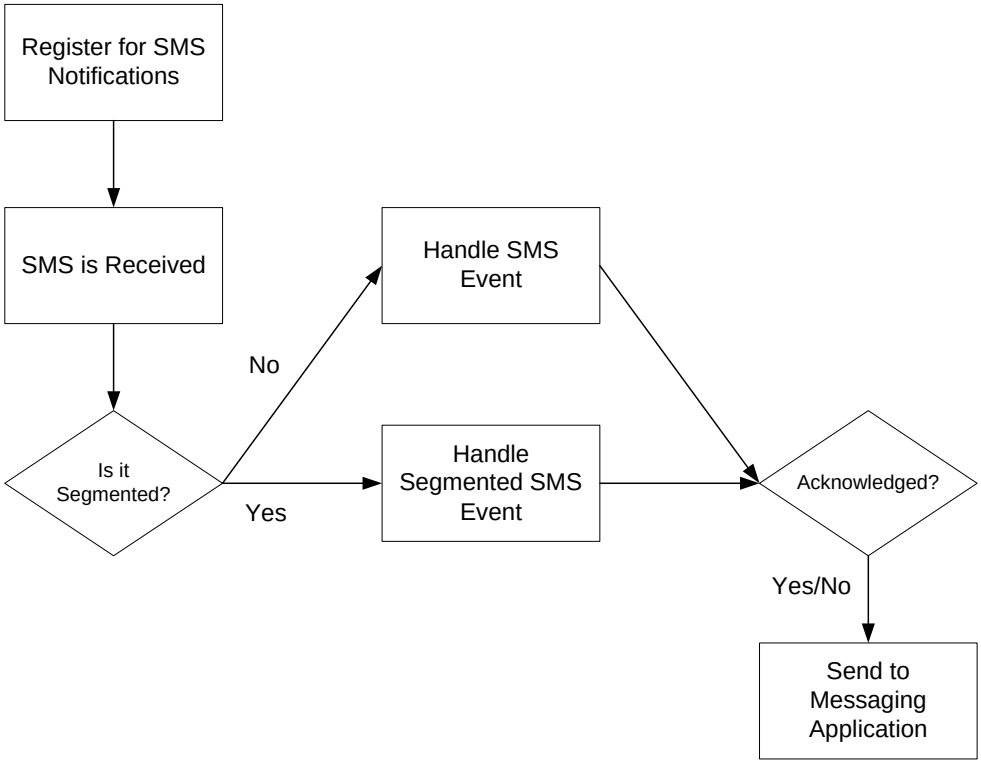
* On CDMA Treo 600, Treo 650, Treo 700p, Treo 755p, and Centro smartphones

9.5.1 Incoming SMS messages and message events

An application must register itself with the Telephony library in order to receive incoming message events. Register for the SMS service using `PhnLibRegister` where `services = phnServiceSMS`:

```
PhnLibRegister (libRef, appFileCreator, phnServiceSMS);
```

The following diagram displays the workflow of incoming SMS messages.



Each incoming message is indicated to an application by sending a `phnEvtMessageInd` event unless it is a segment of a message. In that case, the library sends a `phnEvtSegmentInd` event. The library sends a `phnEvtMessageInd` only after all of a multisegmented messages's parts have been received.

If a new message is added to the database, the application is sent the `phnEvtSegmentInd` event to trigger an update of the message list. The new message is saved in the SMS database and stays there until it's deleted unless the message is a type NBS message. In that case, the message is deleted after the `phnEvtMessageInd` notification is acknowledged by one of the registered applications.

The SMS event comes as a launch code from the Phone library. To detect an incoming SMS event, use the following code:

```
switch (eventP->eventType){
  case phnEvtSegmentInd:
  case phnEvtMessageInd:
        …
    eventP->acknowledge = true;
 break;
}
```

The NBS event comes as a notification from the Notification Manager. To detect an incoming NBS event, use the following code:

```
SysNotifyParamType* notifyP=(SysNotifyParamType*)cmdPBP ;
if (notifyP->notifyType == phnNBSEvent){
    PhnNBSNotificationEventType* NBSParamP =
    (PhnNBSNotificationEventType*)notifyP->notifyDetailsP;

    if (NBSParamP->binary) // NBS binary
}
```

9.5.2 Outgoing SMS messages

To send an SMS message, verify that the Phone is on, that it is registered with the phone library, and that it is on the network by using the following APIs:

```
PhnLibModulePowered();
PhnLibRegistered();
```

Next, get the library reference number using `PhnLibGetDBRef()`.

An SMS message is sent by creating a new outgoing message using `PhnLibNewMessage()` and filling in the desired recipient and message text. The owner and text of the outgoing message is constructed in the following manner:

```
msgID = PhnLibNewMessage(smsLibRef, kMTOutgoing);
PhnLibSetOwner(smsLibRef, msgID, appFileCreator);
PhnLibSetText(smsLibRef, msgID, message, msgLen);
```

More than one recipient may be specified if the message is being sent to a group. The recipient address(es) is constructed in the following manner:

```
addList = PhnLibNewAddressList(smsLibRef);
addressH = PhnLibNewAddress(smsLibRef, address, phnLibUnknownID);
PhnLibAddAddress(smsLibRef, addList, addressH);
MemHandleFree(addressH);
```

The address is then attached and the message is sent:

```
PhnLibSetAddresses(smsLibRef, msgID, addList);
error = PhnLibSendMessage(smsLibRef, msgID, true);
```

NOTE: The `PhnLibSendMessage` API is not synchronous.

The SMS event itself comes as a launch code from the Phone library. To check the SMS status of a sent message, use the following code:

```
case phnEvtMessageStat:
  switch (eventP->data.params.newStatus){
     case kSent:
          PhnLibGetFlags(phnLibRef,
            eventP->data.params.id, &flags);
          if ((flags & kFailed) == 0)… break;
   }
```

The text of an SMS message may be longer than 160 characters. The library segments and reassembles such messages automatically so that the segmentation is transparent to the user of the SMS library.

9.5.3 Handling the GSM alphabet and Palm OS® alphabet

A special alphabet is used to encode SMS messages. The text of all incoming and outgoing messages stored in the message database is encoded using standard Palm OS encoding.

When a message is received, its encoding is changed from the GSM alphabet to the Palm OS alphabet. Any missing characters are replaced by substitution strings. A message is encoded in the GSM alphabet when it is being sent. Optionally, substitution strings may be converted to their character equivalents. The following table shows the character with its corresponding substitution string.

GSM	Palm OS	GSM	Palm OS
Δ	\Delta	Π	\Pi
Φ	\Phi	Ψ	\Psi
Γ	\Gamma	Σ	\Sigma

GSM	Palm OS	GSM	Palm OS
Λ	\Lambda	Θ	\Theta
Ω	\Omega	Ξ	\Xi

NOTE: The client application should not assume that a message contains 160 characters or fewer. Even if the actual message contains fewer than 160 characters, the message's text may be longer than 160 characters because characters that are not available on Palm OS are replaced by substitution strings.

9.5.4 Message segmentation

Most mobile phones allow the user to compose messages of no more than 160 characters; however, in GSM, the Messaging application sets the maximum length of a message to 650 characters. Messages that contain more than 160 characters are segmented.

The SMS library supports three types of segmentation schemes. Two of these methods are text-based while the third is binary. The binary method is preferred, because it allows for the reassembly of messages even if they arrive out of order.

9.5.4.1 Binary segmentation

The binary segmentation scheme works by adding a user data header (UDH) to each segment of a segmented message. The UDH contains a reference number to identify the message, the segment's index, and the total number of segments. Because the UDH takes 6 bytes, the number of characters in a message is reduced to 154.

Using the UDH, it is possible to reassemble messages from their segments even if the segments arrive out of order. The reassembly of incoming segmented messages is transparent to the client application. The client application may retrieve a message's text even if all the segments have not been received. Use `PhnLibGetText` to retrieve the message.

When the first segment of a segmented message is received, the client receives a `phnEvtSegmentInd` event after the segment has been stored in the database. When all of the segments have been received, a `phnEvtMessageInd` event is sent.

Automatic reassembly of messages works if all parts of the message are received within six hours after the first segment is received. After six hours, segmentation information is deleted by the library.

NOTE: The automatic reassembly of messages is a GSM feature. The CDMA version of products does not support automatic reassembly of segmented messages.

9.5.4.2 Textual segmentation

The SMS library supports two textual segmentation schemes. One segmentation scheme is used only for sending segmented messages to an email gateway. The other is used for segmenting regular text messages.

The segmentation scheme used to send messages to an email gateway does not allow the reassembly of the message if the messages arrive out of order. This scheme is a recognized GSM standard. It works by inserting "+" signs into the message's text.

The length of an "inner" segment is reduced to 158 characters, and the length of the first and last segments is 159 characters. A message with three parts is segmented as follows:

```
First segment+
+Inner segment+
+Last segment
```

NOTE: For messages sent to an email gateway, the recipient's address adds to the message's length.

This segmentation scheme is used exclusively to send messages to an email gateway. The SMS library does not use this scheme to send text messages to normal subscribers.

The scheme for segmenting regular messages adds header information to every segment. The header is of the form *i/k*, where *i* is the segment's index and *k* is the total number of segments. The length of the header is not constant and is dependent upon the values of *i* and *k*. A message with three parts is segmented as follows:

```
1/3 First segment
2/3 Second segment
3/3 Last segment
```

The library does not attempt to reassemble messages when they are sent using this segmentation scheme. The application must reassemble the messages as needed. The reassembly is not automatically done by the library because the header does not indicate clearly to which message a segment belongs.

9.5.5 Message database

All incoming and outgoing SMS messages are stored in a message database on the device except for NBS-type messages. NBS message content is stored in the `paramBlock` when the application receives the notification.

An outgoing message stored in the database may be sent using the SMS library. Incoming messages received are also stored in this database. The SMS library handles only messages stored in this database. If you want an application to store the messages in a separate database, you must have the application copy the messages from the SMS database.

This section describes the fields for a single SMS message. The standard Palm OS routines for databases are used to manage these records.

IMPORTANT: A message's internal structure is private and not simple. Client applications should modify data in an SMS message only by using the functions provided by the SMS library.

A record in the message database has four separate parts. The first part has a fixed size, and the size of the other three parts is variable. As a result, a complete record has a variable size.

The four separate parts of a message database record are as follows:

1. Header information
2. Segmentation information
3. Address information
4. Message text

9.5.5.1 Header information

The first part of each record is the message header information. It has a fixed size. The fields in the header information section of the record are used to store flags and determine the size of the `size[]` field. Some fields are not used in all messages. For example, `validity` is used only for outgoing messages, and `segments` is used only for incoming messages.

The SMS header structure is as follows:

```
typedef struct  {
  UInt32 owner;
  SMSMessageType type;
  SMSMessageStatus status;
  UInt32 date;
  UInt32 flags;
  UInt8 validity;
  UInt8 segments;
  UInt16 size[1];
}SmsHeader;
```

See the *Palm API Guide, Palm OS Platform* for details on each field.

9.5.5.2 Segmentation information

The second part of a message record contains segmentation information. This part is variable in size and is accessed through the `size` array. Each segment's size is represented with 2 bytes.

9.5.5.3 Address information

The third part of a message record contains address information. This part is variable in size. The size of this address information is defined in the `size` field. The address information contains the data in a `PhnAddressList` structure.

9.5.5.4 Message text

The fourth part of a message record is the actual text of the message. This part is variable in size. The size is calculated by taking the size of the complete message and subtracting the sizes of the first three parts. All characters in this part of the record are considered to be Palm OS encoded. For outgoing messages, the characters are converted to the GSM alphabet when the message is sent.

NOTE: The conversion of the text is done just before the message is sent. The result of the conversion is not stored with the message. If sending a message fails, the text is converted again when the message is sent again.

9.6 Launching SMS from the New SMS screen

On Palm smartphones, to launch the SMS application in the New SMS screen, use the Palm OS system Helper APIs.

The `HelperServiceSMSDetailsType` structure provides the SMS message to be sent. It is used as the `detailsP` field in the `HelperNotifyExecuteType` parameter when the service class ID is `kHelperServiceClassIDSMS`.

To prefill the New SMS screen with a phone number or email address, set the `dataP` field of the `HelperNotifyExecuteType` parameter to the string `(Char *)` representing the phone number or email address, as explained in the comments below.

Launching SMS from the New SMS screen is detailed in the following code example.

```
HelperServiceSMSDetailsType smsDetails;

messageH = FldGetTextHandle(messageFldP);
messageP = MemHandleLock(messageH);
smsDetails.version = 1;  //The version number for this structure. The current version is 1.
smsDetails.message = messageP;     //The string containing the body of the message to be
                                   //sent or NULL.


HelperNotifyExecuteType execute;
HelperNotifyEventType event;
SysNotifyParamType note;
Err err = errNone;

// If you want the New SMS screen to be prefilled with a phone number or email address,
// then set execute.dataP to the string (Char *) representing the phone number or email
// address. If you do not want the New SMS screen to be prefilled with a number, set
// execute.dataP to be NULL

execute.dataP = "1234567890";

execute.detailsP = &smsDetails;
execute.displayedName = "My SMS app";
execute.helperAppID = 0; // Setting the helperAppID to 0 means "use default helper"
execute.serviceClassID = kHelperServiceClassIDSMS;
execute.err = 0;

MemSet ( &event, sizeof(HelperNotifyEventType), 0);
event.version = kHelperNotifyCurrentVersion;
event.actionCode = kHelperNotifyActionCodeExecute;
event.data.executeP = &execute;

MemSet (&note, sizeof(SysNotifyParamType), 0);
note.broadcaster =  <YOUR_APP_CREATOR>;
note.notifyType = sysNotifyHelperEvent;
note.notifyDetailsP = &event;
note.handled = false;

err = SysNotifyBroadcast(&note);
```

This page intentionally blank

CHAPTER 10

10. System Extensions

This chapter provides details about the system extension features and APIs available in the Palm OS® SDK from Palm, Inc. These features are differentiations from the Palm OS SDK from ACCESS.

10.1 Transparency API

Available on:

- Centro™ and Treo™ smartphone

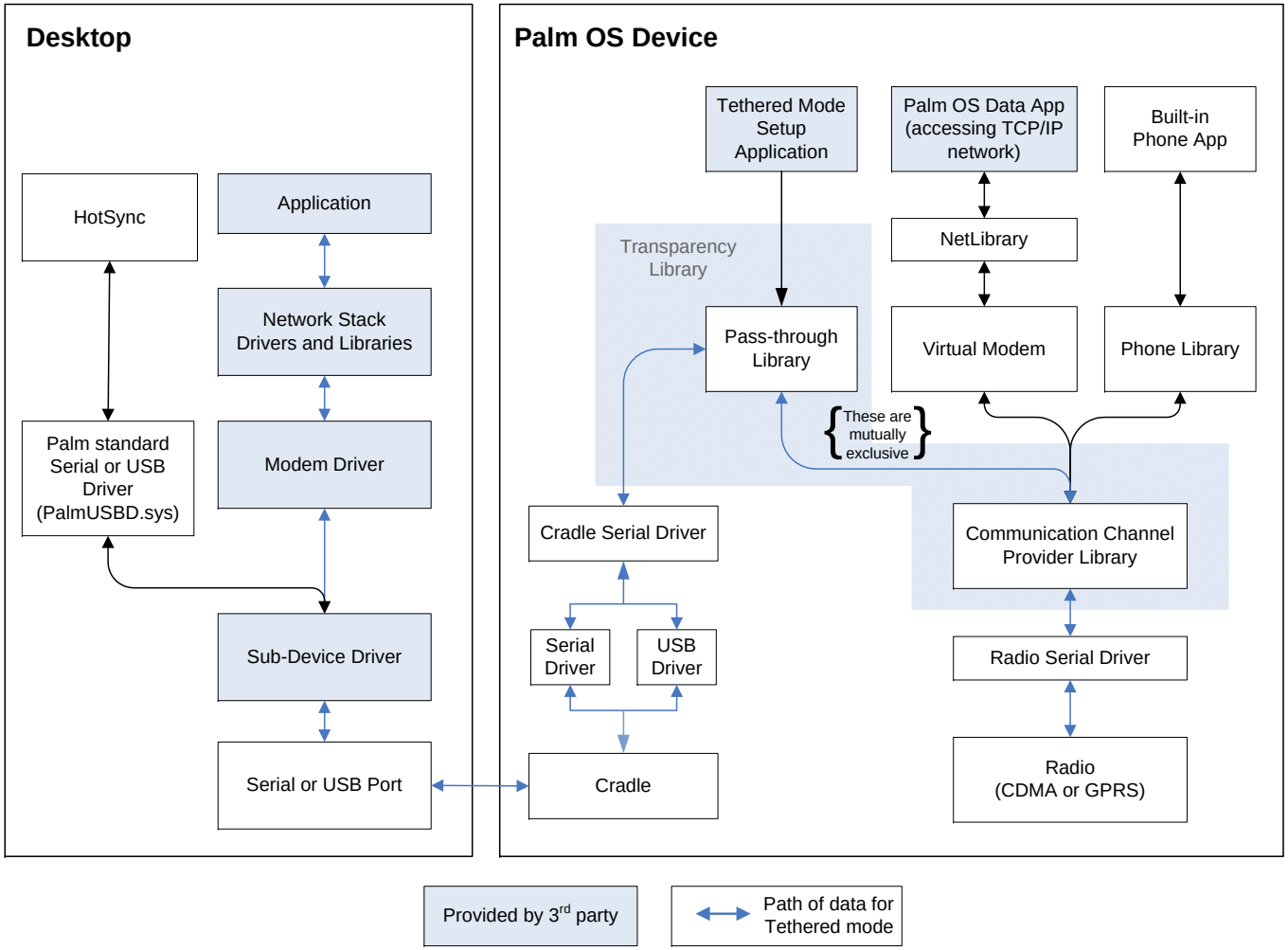
The Transparency library is used to connect the radio modem directly to one of the data ports on a Palm smartphone.

The library is the preferred method to enable tethered mode in an application because it configures the radio in the correct mode to transmit and receive data to the network. The Transparency API is preferable to directly trying to control the radio through its lower level serial driver and through the AT command.

It supports diagnostics or tethered mode. It does not support both simultaneously. See the *Palm API Guide, Palm OS Platform* for more details.

The diagram on the following page shows a possible architecture for connecting the Palm smartphone modem to a desktop using tethered mode and the Transparency library.

Connecting the Palm smartphone modem to a desktop using tethered mode and the Transparency library.



10.2 File Browser API

Available on:

- LifeDrive™ mobile manager
- Tungsten™ T5 handhelds

The File Browser API uses the Exchange Manager registry with some enums defined by ACCESS (formerly PalmSource):

```
#define exgRegEditCreatorID        0xff7b  // creator ID registry
#define exgRegEditExtensionID      0xff7d  // filename extension registry
#define exgRegEditTypeID           0xff7e  // MIME type registry
```

These constants are defined in `FileBrowserLibCommon.h`. Applications should register using `ExgRegisterDatatype` with ID `exgRegEditExtensionID`. The description of a file type being registered should look like this:

```
<Icon 1000,1100>Image
```

The first number is the ID of a bitmap family resource for the large icon provided by the application. The second number is the ID of the small icon provided by the application. You can skip the second number if the small icon ID number is one greater than that of the large icon:

```
<Icon 1000>Image
```

The text after the close-angle-quote is the description string that would normally be used. It isn't currently exposed in the File Browser UI, but it could easily be exposed later on, so be sure to include a description.

The size of the large and small icons should match the dimensions of the large and small application icons in the Applications View: 22 x 22 pixels and 15 x 9 pixels. These dimensions are expressed in normal density; the double-density dimensions are 44 x 44 and 30 x 18. The absolute maximum dimensions supported by the File Browser and Favorites applications are 32 x 22 and 16 x 11—64 x 44 and 32 x 22 in double-density. As always, be sure to include a normal-density bitmap as the first element in each bitmap family.

When the File Browser wants to open a file, it does so through the Exchange Manager. Applications typically receive `sysAppLaunchCmdExgReceiveData` to open files. The application won't receive a `sysAppLaunchCmdExgAskUser` sublaunch but will receive a `sysAppLaunchCmdExgReceiveData` sublaunch, just as it would for incoming beams.

Applications should check the name field in the socket to see if it is a "file:" URL. If so, the application should set the `goToCreator` field in the socket to its own creator ID. It can then proceed as if a beam was received, but it shouldn't create a new record. Instead, it should store the parsed data somewhere temporarily and set the `goToParams` struct in the socket to refer to this temporary area.

Alternatively, once you know you're handling a "file:" URL, you can have your application stop using the Exchange Manager. Have your application put the URL into a feature pointer and set the `goToCreator`. Then, when the application receives the `sysAppLaunchCmdGoTo` launch command, you can have the application look for this

feature pointer. If the feature pointer is found, you can have the application parse the URL to get a `volRefNum` and path. Then you can have the application proceed as described earlier for `kSysAppLaunchCmdOpenFile`. This is the approach used in the sample application. The code to parse the "file:" URLs is included in the File Browser API library. The sample application available in the Palm OS SDK includes wrappers around this entry point to make the code easier to use.

The wrapper function is as follows:

```
static Char *ParseFileURL (const Char *url, UInt16 *volRefNumP)
   {
      Char *path = NULL;
      UInt16 refNum;
      Err err;

      err = SysLibFind(kFileBrowserLibName, &refNum);
      ErrFatalDisplayIf(err, "Can't find file browser lib");
      err = FileBrowserLibOpen(refNum);
      ErrFatalDisplayIf(err, "Can't open file browser lib");
      FileBrowserLibParseFileURL(refNum, url, volRefNumP, &path);
      FileBrowserLibClose(refNum);
      return path;
```

The entry point used in this wrapper function is:

```
Err FileBrowserLibParseFileURL( UInt16 refNum, const Char *urlP,
      UInt16 *volRefNumP, Char **filePathP );
```

It takes a "file:" URL as input and outputs a `volRefNum` and path. It also allocates the path. The caller is responsible for freeing it. If the URL can't be parsed for any reason, it passes back `vfsInvalidVolRef` for the `volRefNum` and `NULL` for the path.

A `fileType` parameter is included in the Open and Save As dialog box APIs. Pass `NULL` for the `fileType` parameter to go to the root directory when switching volumes. The initial folder is the root directory unless you specify a root folder. To specify an initial file name, you must use a full path.

Pass an extension with the initial period or a MIME type to use the registered default directory for the specified file type. The default directory is used when switching volumes and when no initial directory is specified in the initial path.

For example, you could invoke the Save As dialog box with ".jpg" or "image/jpgeg" as the `fileType`, "MonaLisa.jpg" as the initial path, and the volRefNum for the SD card. The initial directory would then be /DCIM, and the file name would be pre-populated with `MonaLisa.jpg`. When the user presses the internal drive button, the /Photos and Videos directory of the internal drive is displayed. When the user presses the internal drive button again, the root directory is displayed.

By registering with the Exchange Manager and handling `sysAppLaunchCmdExgReceiveData`, an application can ensure that its files appear with the correct icon in Files, and that selecting these files opens them in the application. This is the most important aspect of the File Browser API, but there is another side to it as well. Applications can use the File Browser library to display an Open or Save As dialog box. This is most appropriate for applications that attempt to mimic their desktop counterparts. These dialog boxes should not be used for other applications.

The File Browser library's entry points are declared in `FileBrowserLib68K.h`. To display the Open and Save As dialog boxes, only four of these entry points are required: `FileBrowserLibOpen`, `FileBrowserLibClose`,

FileBrowserLibShowOpenDialog, and FileBrowserLibShowSaveAsDialog. Some constants defined in FileBrowserLibCommon.h are also needed.

For example, to display an Open dialog box:

```
UInt16 refNum;
UInt16 volRefNum;
Char *path = MemPtrNew (kFileBrowserLibPathBufferSize);
const UInt16 numExtensions = 1;
const Char *extensions[numExtensions] = {"txt"};
Err err;

ErrFatalDisplayIf (path == NULL, "Can't alloc path");

// Find the library and call the Open function. There is no need
// to load the library.
err = SysLibFind (kFileBrowserLibName, &refNum);
ErrFatalDisplayIf (err, "Can't find file browser lib");
err = FileBrowserLibOpen (refNum);
ErrFatalDisplayIf (err, "Can't open file browser lib");

// If you want to start with a particular volume, set volRefNum
// to that volume. Otherwise, use vfsInvalidVolRef.
volRefNum = vfsInvalidVolRef;

// If you want to start in a particular directory, set path to
// that directory. Otherwise, use an empty string. You can include
// a filename as well as a directory if you want a file to be
// selected initially. You can specify a filename with no path, but
// only if you specify a fileType.
path[0] = chrNull;

// Display the Open dialog box. Returns whether a file was selected.
if (FileBrowserLibShowOpenDialog (refNum, &volRefNum, path,
   numExtensions , extensions,    // filter to show only these files
   "text/plain",               // use default folder for this fileType
   "Select Item",                // title for the dialog
   kFileBrowserLibFlagNoFolders)) // pick a file, not a folder
{
   // Do something with volRefNum and path.
}

// Clean up. There is no need to remove the library.
MemPtrFree (path);
FileBrowserLibClose (refNum);
```

The File Browser library is pre-loaded when the device is reset, so you just need to find it and call the FileBrowserLibOpen and FileBrowserLibClose entry points. You don't need to call SysLibLoad or SysLibRemove.

The volRefNum and path are used both as input and as output. For input, you can do the following:

- Not specify a volume or a path (vfsInvalidVolRef and "")
- Specify a volume only
- Specify a volume and a directory but no file name
- Specify a volume, a directory, and a file name

- Specify a volume and a file name but no directory
- Specify a file name but no volume or directory

The last two options require that a `fileType` be specified as well. If the user selects a file or folder and selects OK, the volume and path of the selected file or folder is passed back in the `volRefNum` and `path` parameters and the function returns `true`. Otherwise, the parameters are left as they are and `false` is returned.

If a list of extensions is passed in, only files with one of the specified extensions are listed, along with all the folders. Use `0` for `numExtensions` and `NULL` for `extensions` to disable filtering.

You can specify a MIME type or an extension for the `fileType`. Be sure to include a period before the extension: `.txt`. If a `fileType` is specified, the Open dialog box automatically navigates to the default directory for the specified `fileType` when the user switches volumes. The default directory for a `fileType` can be set using `VFSRegisterDefaultDirectory`. It can be different for different media types—for example, SD/MMC as opposed to the internal drive. If the initial path doesn't include a directory and a `fileType` is specified, the default directory for the specified `fileType` is used. A `fileType` should be specified whenever possible; it makes it faster for the user to navigate to the appropriate directory. If you use `NULL` for the `fileType`, the user is taken to the root directory when switching volumes.

The title of the Open dialog box depends upon flag settings specified by the client application. If `NULL` is passed in for the title, the Open dialog box is given the default title specified by the flag settings in the client application.

You can use various combinations of the following flags defined in `FileBrowserLibCommon.h`:

- `kFileBrowserLibFlagOneVolume`—no volume picker
- `kFileBrowserLibFlagNoFiles`—no files, only folders
- `kFileBrowserLibFlagNoFolders`—no folders, only files

These flags should be logically combined. For example: `kFileBrowserLibFlagOneVolume | kFileBrowserLibFlagNoFiles`. Use zero to specify no flags. If you use `kFileBrowserLibFlagOneVolume`, be sure to specify a volume because the user won't be allowed to switch to any other volume. If you use `kFileBrowserLibFlagNoFiles`, the user is only allowed to pick a folder. Files won't be shown at all. If you use `kFileBrowserLibFlagNoFolders`, the user is only allowed to pick a file but can still navigate into folders to find a file. If you don't use either of these flags, the user is allowed to pick a file or a folder.

`FileBrowserLibShowSaveAsDialog` is very similar to `FileBrowserLibShowOpenDialog`. In addition to allowing the user to navigate to any directory, it includes a field where the user can enter a file name. If the specified path includes a file name, it appears in this field. An additional parameter for the default extension, if specified, is appended to the file name when the user does not enter an extension. Use `NULL` for the default extension to use the file name exactly as it is entered.

The flags used for the Save As dialog box are as follows:

- `kFileBrowserLibFlagOneVolume`—no volume picker
- `kFileBrowserLibFlagPromptOverwrite`—warn before replacing
- `kFileBrowserLibFlagRequireExtension`—only given extensions
- `kFileBrowserLibFlagNoNewFolder`—no New Folder button

The first flag is used in the same way as it is for the Open dialog box. Use `kFileBrowserLibFlagPromptOverwrite` if you want to warn the user when a file name of an existing file in the selected directory is selected. This is preferable to checking for duplicate file names afterward because it allows the user to edit the file name or choose a different directory. Use `kFileBrowserLibFlagRequireExtension` if you want to force the user to use a specific extension or one of several extensions. Pass in the list of legal extensions in the `numExtensions` and `extensions` arguments. Use the `kFileBrowserLibFlagNoNewFolder` flag if you want to prevent the user from creating new folders. This hides the New Folder button.

The `FileBrowserLibShowSaveAsDialog` function doesn't actually save anything. It just prompts the user for where to save the file and what to call it. It's up to you to do the following:

- Create the file, truncating the existing file, if any.
- Write to the file.
- Close the file.

Similarly, `FileBrowserLibShowOpenDialog` doesn't open the file or folder selected by the user. It's up to you to do the following:

- Open the file or folder.
- For folders, enumerate the contents.
- Close the file or folder.

10.3 Smart Text Engine (STE) API

Available on:

- Centro™ and Treo™ smartphones
- LifeDrive™ mobile manager

The Smart Text Engine (STE) shared library enables applications to implement rich text display and processing. The STE is designed to allow any application to automatically identify, render, and link web URLs, email addresses, and phone numbers to appropriate applications. For example, selecting a phone number in SMS dials the number, or selecting a URL in an email launches that URL in the web browser. The STE library offers a high level of convenience for end users as it seamlessly links information and communication applications.

The STE library performs three basic functions: parsing, rendering, and displaying. Applications supply the STE with a text stream object and specify the area on the screen to render. This area can include a scroll bar that is used to scroll the contents of a multi-page display. The STE passes the text through its three layers to create, display, and activate links.

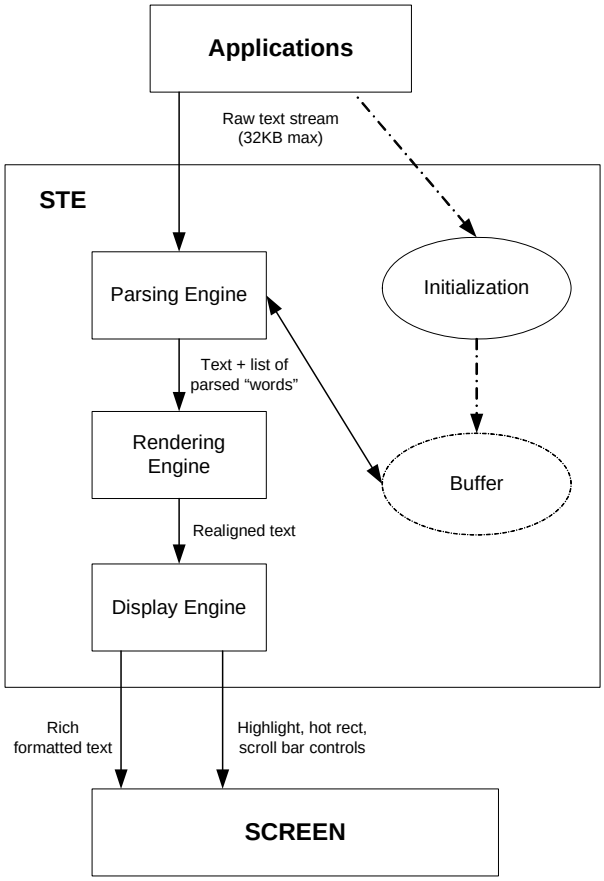
The STE library solves the traditional Palm OS software limitations of the text field: black- and-white display with one font style. Using the STE library, applications can mix bold and standard fonts, mix colors, add graphics, and even add emoticons.

In addition to the information provided here, you can also refer to the STETest.zip sample code file in the Palm OS SDK for information on how to use this library.

Refer to the *Palm API Guide, Palm OS Platform* for detailed information about each of the STE APIs.

10.3.1 STE architecture

The STE includes several components—the parsing engine, the rendering engine, the display engine, event handling, and text selection. This modular design allows new components to be added easily. For example, if an HTML parser is needed, it can be added to the STE library without major architectural modifications. The following figure shows the architecture of the STE.



10.3.1.1 STE parsing engine

The parsing engine is responsible for finding the URLs, email addresses, and phone numbers within text. It also detects emoticons and special Smart Text delimiters or tags that affect the text format display. This allows applications to quickly detect whether the text has STE delimiters.

Scanning for URLs, email addresses, and phone numbers requires complex processing in the STE library. A specific algorithm is used to check the validity of characters in URL and email addresses text strings. Checking phone numbers is more complex because it involves appending multiple numbers separated by spaces to compile the final phone number. A basic matching algorithm is used to find emoticons with one special condition: all smileys must have either a colon (:) or semicolon (;) for the eyes.

Smart Text delimiters are identified by the characters "`//STE`" followed by additional characters that define the exact properties of the delimiter.

The parsing engine works by separating the string into separate *words*—groups of characters separated by a space character. Each of the words is pre-scanned to check whether it is a potential URL, email address, phone number, or emoticon type. If the word qualifies as any of these types, it is then further scanned to see if it matches the requirements for that type. Phone number checks require scanning for consecutive groups of words to form the final complete phone number.

The result of the parsing engine is a list of parsed items. This list is then used by the rendering engine to format and display the rich text.

10.3.1.2 STE rendering engine

The rendering engine takes the text input, along with the parsed info list, and determines exactly where the text belongs in the list. If there is no formatting involved, this is very much like displaying text in a text field. When special objects and formatting are added to text, the rendering process becomes more complex.

The basic algorithm goes through the text string and determines whether the next word can fit on the current line in the display. All the STE text string display properties are considered when determining fit. The font can be bold, normal, or colored. There can be emoticons and other bitmaps displayed, as well. The different widths of the text and graphics in different display modes are also considered when calculating fit. The engine keeps track of the actual text that is displayed on each line to process text selection and manipulation.

After all the text has been parsed and rendered, it can be displayed.

10.3.1.3 STE display engine

The display engine takes the data structures created by the rendering engine and displays the correct data at the correct location in the list. The display engine also controls text highlighting and scroll bar positioning.

10.4 REM Sleep API

Available on:

- Centro™ and Treo™ smartphones
- Palm® Tungsten T|X handheld

The REM Sleep API allows applications on a smartphone to run while the display is off and the keyboard and digitizer are disabled. The REM Sleep API is on Palm smartphones and T|X handhelds, but not on prior Tungsten™ or Zire™ handhelds.

REM sleep is especially useful for applications that constantly run in the foreground, but also want to conserve the battery, such as MP3 players, etc.

REM sleep mode is initiated at the point when the smartphone would otherwise go into deep sleep mode. You should already be familiar with the sleep-deferral and notification mechanisms in the Palm OS before using the REM Sleep API. For more information on notifications, refer to the "Notifications" section, in particular the "Sleep and Wake Notifications," in the *Palm OS Programmer's Companion, vol. I*.

The smartphone may be put to sleep for two reasons: The user presses the power button or another button that functions as a power button, or the smartphone remains idle for the duration set as the auto-off timeout value.

For an example of how to use the REM Sleep API, refer to `_REMTimer` in the Sample Code section of the SDK.

10.4.1 Normal sleep deferral

The process of a smartphone going to sleep begins with a virtual key event. In the case of a user pressing the power button, a `vchrHardPower` event occurs. In the case of the auto-off timeout value being reached, a `vchrAutoOff` event occurs.

When the initiating event is processed by `SysHandleEvent`, a sleep-request notification is broadcast to all registered recipient applications. If any recipient sets the `deferSleep` member of the notification parameter block, the system does not go to sleep and the smartphone continues to run.

When sleep is deferred, the application or library responsible should enqueue `vchrResumeSleep` after it has finished doing whatever caused it to defer the sleep. The event `vchrResumeSleep` works similarly to `vchrHardPower` and `vchrAutoOff` in that it causes the sleep-request notification to be broadcast. Another application may then set the `deferSleep` member, causing the defer process to continue.

If the sleep request is not deferred, the system enqueues a virtual key event with the `vchrPowerOff` character. When the `vchrPowerOff` event enters `SysHandleEvent`, a sleep notification is broadcast to inform its recipients that the system is going to sleep, and the system is immediately put to sleep thereafter.

The `sysNotifySleepRequestEvent` and `sysNotifySleepNotifyEvent` APIs are part of the Garnet OS header file `NotifyMgr.h`. For more information and documentation, refer to the ACCESS Developer Network at the following URL:

http://www.access-company.com/developers/index.html

These APIs work as described:

- `sysNotifySleepRequestEvent` - A sleep request event is broadcast to indicate that a device is about to go to sleep, and provides an opportunity for applications to perform an action or even delay going to sleep. This event may be broadcast multiple times if one or more clients defer sleep.
- `sysNotifySleepNotifyEvent` - A sleep notify event is broadcast to indicate that a device is going to sleep immediately after the broadcast is finished.

10.4.2 REM sleep mode

REM sleep mode occurs in the time between when the sleep-request notification occurs and when the sleep notification occurs. If the sleep-request notification is not deferred, the display is turned off and then a REM request notification is broadcast (`hsNotifyRemSleepRequestEvent`). The REM sleep-request notification has the exact same parameter block as the sleep-request notification. A recipient of the notification that needs REM sleep simply sets the `deferSleep` member in the parameter block. If the `deferSleep` parameter is set in the REM sleep-request notification, the system broadcasts a REM notification to inform recipients that REM sleep has been entered.

The expectation is that whatever is happening during REM sleep is a transient condition that will in most cases lead to deep sleep shortly thereafter. To accommodate this expectation, the REM-request notification is periodically resent until it is not deferred. The current implementation resets the auto-off timer to expire a few seconds after entering REM sleep so that the whole REM-request notification process repeats periodically. You should not rely on this particular implementation, but instead defer the REM request as often as it comes.

For more information, refer to the sleep mode flowchart in **Section 10.4.4**.

IMPORTANT: An application has to register for REM sleep notification (`hsNotifyRemSleepRequestEvent`) and normal sleep (`sysNotifySleeprequestEvent`) notification when initializing the application in order to support any kind of sleep deferral.

The `hsNotifyRemSleepRequestEvent` and `hsNotifyRemSleepEvent` APIs are defined in the header file `Incs/common/system/HsCreators.h`, which is part of the Palm OS SDK. For complete reference information, refer to the document *Palm API Guide, Palm OS Platform*.

These APIs work as described:

- `hsNotifyRemSleepRequestEvent` - This notification is sent after `sysNotifySleepRequestEvent` to determine whether an application needs to continue processing while the screen is turned off. The details parameter is `SleepEventParamType` in which `deferSleep` should be incremented if REM sleep is required.
- `hsNotifyRemSleepEvent` - This notification is sent to indicate that REM sleep is starting, due to `hsNotifyRemSleepRequestEvent` being deferred. At this point, the screen is turned off and the autooff timer is primed to expire. This causes the cycle to restart with `sysNotifySleepRequestEvent` and `hsNotifyRemSleepRequestEvent` both being resent.

10.4.3 Detecting REM sleep mode

If you need to determine whether the smartphone is already in REM sleep while handling a REM-request notification, register for the sleep-request notification, as described in **Section 10.4.2**,and use `HsAttrGet()` to obtain the value of `hsAttrDisplayOn`. By the time the REM request is broadcast, the display is already turned off and non-power/application keys are disabled. The sleep reason in the REM request simply reflects whatever the sleep reason was from the preceding sleep request.

10.4.4 LCD on/off notification

Available on:

- Centro™ smartphones
- Treo™ 680, 700p and 755p smartphones

This notification lets an application know when the LCD screen has been turned on or off. To register for this notification, use:

```
SysNotifyRegister(myAppCardNo, myAppDbID, kPalmCreatorIDLcdState, 0,
sysNotifyNormalPriority, 0);
```

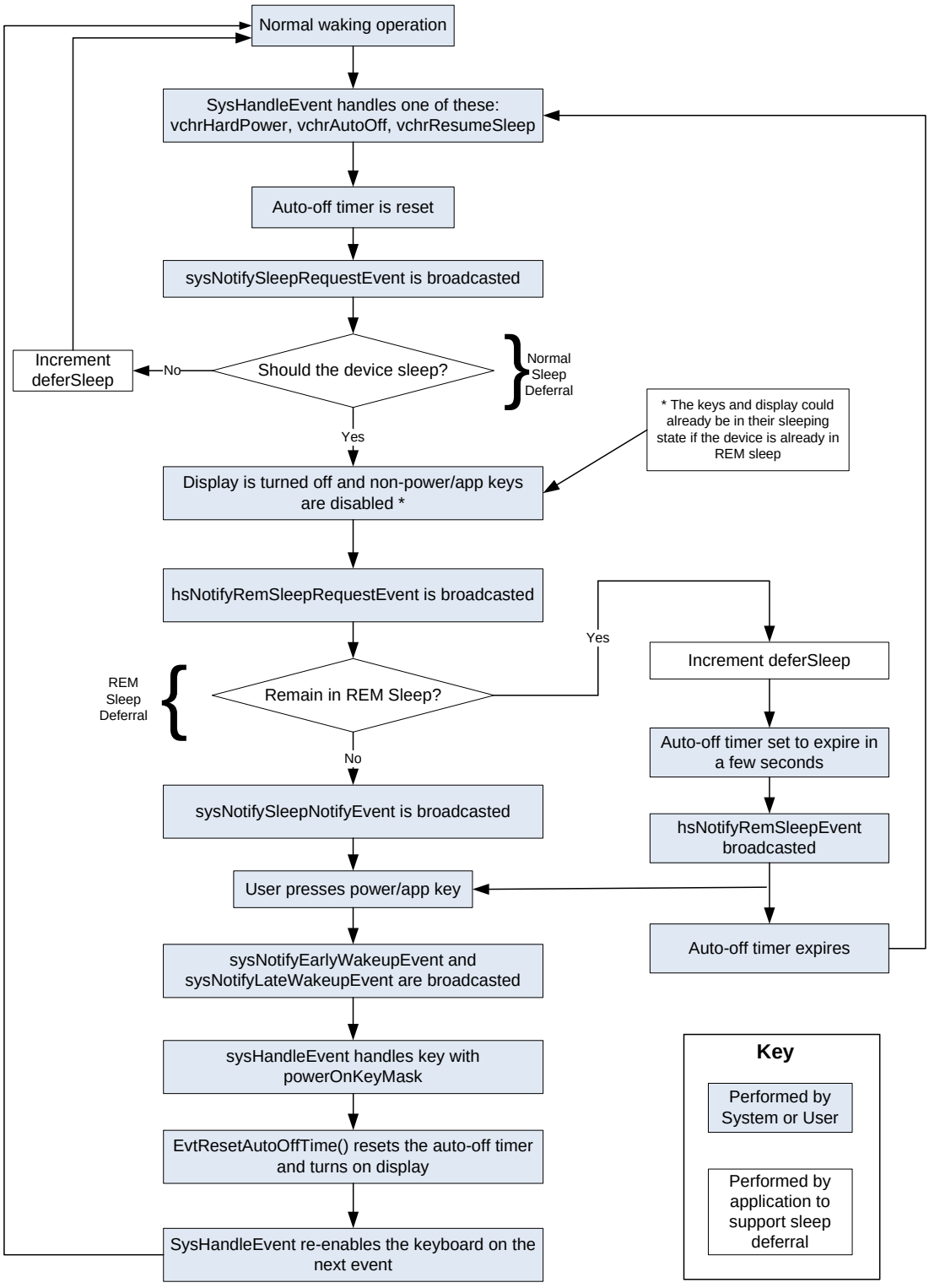
When receiving a notification, use the following code (in PilotMain) to check for the LCD on/off notification:

```
-
  case sysAppLaunchCmdNotify:

    pNotify = (SysNotifyParamType *) cmdPBP;

    switch (pNotify->notifyType)
    {
        case kPalmCreatorIDLcdState:
        if (pNotify->notifyDetailsP == 0)
        {
            // LCD is off
        }
        else
        {
            // LCD is on
        }
        break;
   }
```

The notification is located in the `palmOneCreators.h` header file, and can also be found in the *Palm API Guide, Palm OS Platform*.

Because the display information is sent as a notification, when an application receives this notification, the display state may have already changed. For this reason, if an application is interested in the current state of the display, it may be useful for the application to poll the display state. See **Section 10.4.3** for information on using `HsAttrGet()` to obtain the value of `hsAttrDisplayOn`.

The following figure shows the sleep mode flowchart of events.



10.4.5 Waking up from REM sleep mode

While the system is running in REM sleep mode, it may be necessary to "wake up" and turn the display on, which you can accomplish by calling `EvtResetAutoOffTimer`.

When displaying any UI that must be seen by the user immediately, you should call `EvtResetAutoOffTimer` to ensure that the display is on. However, it's likely that a user won't even pay attention to their smartphone that is in REM sleep mode. If the user must be notified, consider using the Attention Manager instead.

In order to maintain the expected user experience, while in REM sleep mode the keyboard and touch screen are set to behave as if the smartphone is sleeping. That is, the touch screen is disabled and only those keys that would normally wake the smartphone are active. If one of the keys that would normally wake the smartphone is pressed, it generates a key event with the `poweredOnKeyMask` modifier bit set. When `EvtGetEvent` sees a key event with this bit set, it turns the display back on by calling `EvtResetAutoOffTimer`.

A version of REM sleep has been included in the Palm OS software since version 3.5. After waking up from deep sleep, the auto-off timeout has effectively already expired. If the event queue is emptied without the auto-timer ever being reset, a `vchrAutoOff` event is immediately returned from `EvtGetEvent`, causing `SysHandleEvent` to initiate putting the smartphone back to sleep right away. This behavior has not been changed, but has been accounted for in the REM Sleep API. There will not be a barrage of auto-off events if REM sleep is entered from deep sleep without the display ever being turned on. (This happens during mail sync in certain applications, for example.)

10.5 Keyguard API

Available on:

- Centro™ and Treo™ smartphones

The keyguard API prevents a smartphone from being turned on by accident. Sometimes when a smartphone is in a user's pocket or purse, the power button or other key is pressed inadvertently. Applications can set or query the state of the Keyguard feature on the smartphone.

When the Keyguard is active, the digitizer is locked and `EvtGetEvent` will not return any key events that came from the keyboard. Other key events not generated by the keyboard are returned.

To programmatically enable or disable the Keyguard, use `HsAttrSet()` to set the value of `hsAttrKeyboardLocked` to `true`. Do *not* send an `hsChrKeyboardLock` key event to enable the Keyguard.

To query whether the Keyguard is active, use `HsAttrGet()` to get the value of `hsAttrKeyboardLocked`.

NOTE: `hsAttrKeyboardLocked` does not always indicate that the Keyguard dialog box is being displayed. There are states in Keyguard in which key and pen events are filtered and the Keyguard dialog box is not displayed.

To prevent the Keyguard from being enabled, block the virtual character `hsChrKeyboardLock` from being handled by `SysHandleEvent`. In an active application,

check for `hsChrKeyboardLock` between `EvtGetEvent` and `SysHandleEvent`. In an application running in the background, register for `sysNotifyVirtualCharHandlingEvent` notification and mark the notification handled when a virtual character comes through. Blocking `hsChrKeyboardLock` is all that is required. Even if the Keyguard is already enabled, such as through auto-keyguard, the Keyguard is disabled when `hsChrKeyboardLock` is blocked.

Setting the value of `hsAttrKeyboardLocked` to `true` does *not* immediately enable the Keyguard. Instead, an internal state is set and `hsChrKeyboardLock` is enqueued. Only when `hsChrKeyboardLock` reaches `SysHandleEvent` is the Keyguard enabled. At that time, the Keyguard dialog box is displayed, taking control from the active application.

When auto-keyguard is on, the Keyguard is enabled after waking up from REM or deep sleep until either the dialog box is displayed or it is determined that the Keyguard doesn't need to be enabled because the smartphone wasn't asleep long enough. This prevents extra keystrokes from being lost while a smartphone wakes up. If the value of `hsAttrKeyboardLocked` is set to `false` or `hsChrKeyboardLock` is blocked while the auto-keyguard process is occurring, the Keyguard is disabled before the dialog box is displayed.

10.6 Option and Shift key APIs

Available on:

- Centro™ and Treo™ smartphones

The Option and Shift keys increase the range of input possibilities on Palm smartphones. For example, a map application, which uses a 5-way Up key press to scroll up, can use an Option+5-way Up key press to zoom in.

There are APIs available to detect whether Caps Lock or Option Lock is on. An application can also detect if the Option key or the Shift key has been pressed.

There are also APIs available to set Caps Lock or Option Lock and to programmatically simulate Option or Shift key presses.

Also, if an application includes a Graffiti Shift Indicator (GSI) UI object in the application's resource, then Option and Shift key presses can be visually detected.

10.7 MMS helper functions API

Available on:

- Centro™ and Treo™ smartphones

The MMS helper function APIs are only available on Treo and Centro smartphones. The MMS (Multimedia Messaging Service) application is the built-in application that provides an interface for other applications to send and receive MMS messages. You can interface with the MMS application through the Helper API.

The `MMSHelperCommon.h` header file is a public 68k header file that is used with the Default Helper Library. This file contains the type and structure parameters for the MMS Helper Service Class. To handle MMS, register to this service using the Default Helper library.

You can also refer to the MMSReceiver and MMSSender sample codes in the Palm OS SDK to learn how to use this library.

10.7.1 MMS usage model

This section explains how to use the MMS Helper structures with the Palm OS Helper APIs to submit an MMS send request to the MMS application.

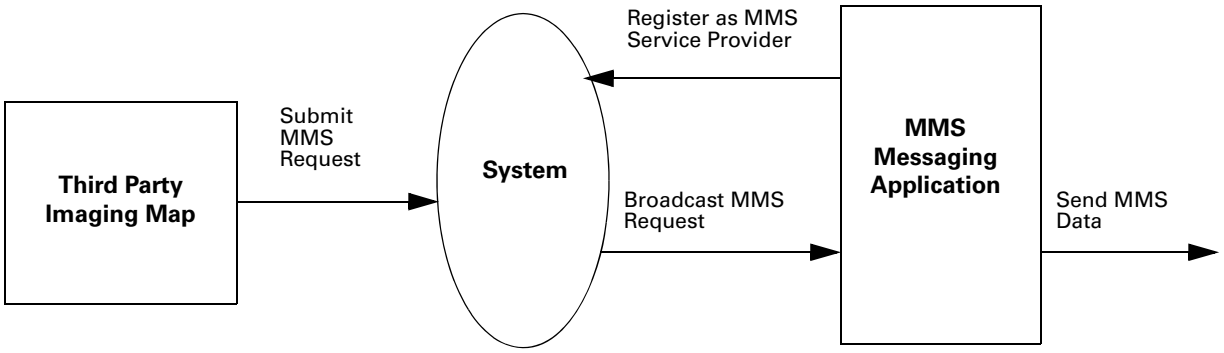
The built-in MMS application is registered in the system as a Helper service class for the MMS service type. When the system receives MMS Helper requests submitted by applications, the system broadcasts a notification to the MMS application to accept and receive the MMS helper data structure from the requesting application.

This gives audio and imaging applications an easy and convenient method for sending picture or audio files using a phone without having to implement the Phone API interface code. The application needs to know only the phone number or email address of the recipient in order to send MMS data to the recipient.

NOTE: Currently, only the built-in MMS application is registered as an MMS Helper service provider. In the future, there may be multiple applications registered as MMS Helper service providers. Be as specific as possible when designing an application request for an MMS Helper service.

A Helper requesting application can specifically request which Helper service provider to invoke by specifying the exact creator ID of that Helper service provider in the `helperAppID` field of the `HelperNotifyExecuteType` structure. Setting `helperAppID` to `0` indicates all Helper service providers.

The following figure shows the MMS helper usage model.



10.7.2 MMS sample code

The following sample code shows how an application submits an MMS send request to an MMS service provider. The implementation first populates the `HelperServiceMMSDetailsType` and `HelperServiceMMSObjectType` structures with the data, address, and message information. It then attaches the detail structure into a `HelperNotifyExecuteType` so that it can be submitted to the system and be broadcast to all the service providers.

`HelperServiceMMSObjectType` is the structure that holds the MMS object. This structure is part of the linked list which is pointed by the object member of `HelperServiceMMSDetailsType`.

`HelperServiceMMSDetailsType` is the structure that helpers of this service class use to send an MMS message

```
{
... ... ...
   case MMSSendButton:
    {
    BitmapPtr bitmapP = NULL;

    HelperServiceMMSDetailsType MMSDetails;
    HelperServiceMMSObjectTypeobject;

    MemSet(&MMSDetails, sizeof(HelperServiceMMSDetailsType),0);
    MemSet(&object,sizeof(HelperServiceMMSObjectType),0);

    bitmapP = MemHandleLock (gLockBitmapH);

    object.pageNum = 1;
    object.tempFileName = NULL;
    object.mimeType = "application/vnd.palm.bmp";
    object.bufferP = bitmapP;
    object.bufferLen = MemPtrSize(bitmapP);

    MMSDetails.object = &object;
    MMSDetails.version = 1;
    MMSDetails.justSend=false;
```

```
MMSDetails.cc = "a cc address";
MMSDetails.subject = "subject text";
MMSDetails.message = "message text";

err = PrvInvokeHelperService(kHelperServiceClassIDMMS,
      “recipient_address@palm.com”, "Helper Sender", &MMSDetails);

MemHandleUnlock(gLockBitmapH);
```

`PrvInvokeHelperService` in the previous code snippet is implemented to invoke the helper service, as shown:

```
HelperNotifyExecuteType execute;

event.actionCode = kHelperNotifyActionCodeExecute;
event.data.executeP = &execute;

PrvBroadcastHelperAction(&event);
```

The MMS service that receives the notification then executes the `helperNotifyActionCodeExecute` request if it is intended for this helper.

```
HelperNotifyEventType* helperParamP;

HelperNotifyExecuteType* executeP;

if (helperParamP->actionCode == kHelperNotifyActionCodeExecute)

{

   // If this is a request for MMS service and it either targets
   //  this specific helper via its helperID or wildcard (0), then
   //  handle it.

      if ((executeP->serviceClassID == kHelperServiceClassIDMMS)
         &&
         ((executeP->helperAppID == prvMyHelperID) || (executeP->helperAppID == 0)) )
      {

   /* further processing */

   }

.
.
.
}
```

10.8 NVFS

Available on:

- Centro™ and Treo™ smartphones
- LifeDrive™ mobile manager
- Tungsten™ T5, Tungsten™ E2, and Palm® T|X handhelds
- Palm® Z22 organizer

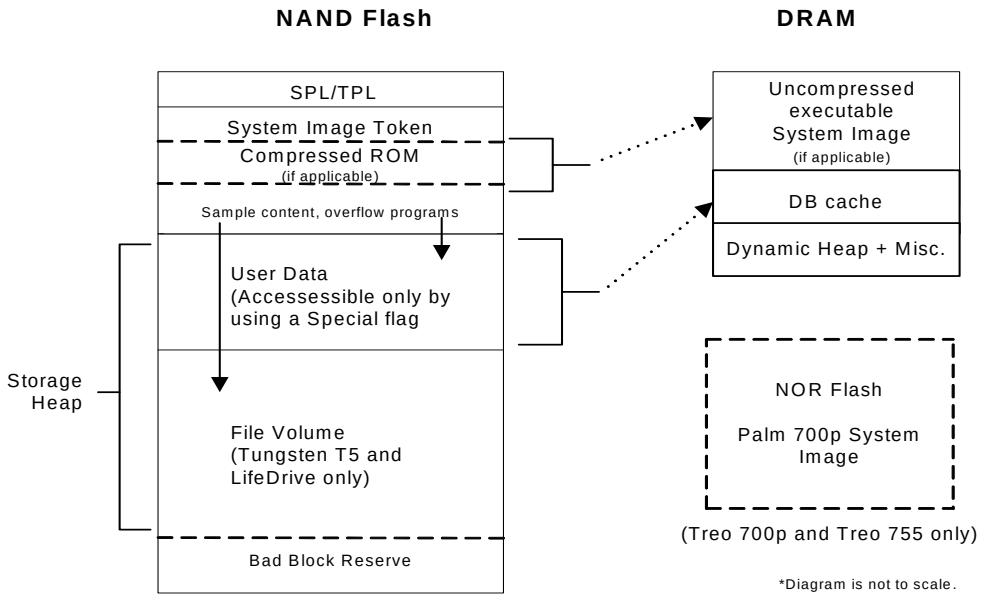
NVFS stands for Non-Volatile File System. Storage in Palm devices traditionally included nonvolatile NOR flash where the Palm OS and built-in applications were stored, and a volatile SDRAM (synchronous dynamic random access memory) where the Dynamic and Storage heaps were stored. If the power was removed, all data in the RAM was deleted.

Treo 650, Treo 680, Treo 700p, Treo 755p, and Centro smartphones, LifeDrive, Palm T|X and Tungsten T5 and E2 handhelds have NAND flash memory, which is nonvolatile. NAND flash houses the storage heap and other data so that it is not erased even if the power is drained from the device. Unlike other devices, Treo 700p and Treo 755p smartphones include both NOR and NAND flash memory.

The main difference between NOR and NAND flash memory is that NOR flash is completely XIP (execute in place), and all addresses in NOR flash are mapped using address pins so that each byte can be accessed. In NAND flash memory, data is accessed in blocks, and only a small section. The section that houses the boot code is XIP.

This means that code instructions cannot be executed out of NAND memory directly, and thus require the system image (rom), as well as any other application, to be executed from DRAM. For more information on NOR and NAND flash memory, see **Section 10.8.1**.

The following figure displays the NVFS memory map for the devices mentioned at the beginning of this section.



For devices where the system image resides in NAND flash memory, copying and uncompressing the system image to DRAM for execution occurs at early boot time. Other applications residing in the Storage Heap are copied upon launch from NAND to DRAM for execution. Both code and data is copied to/from the NAND Storage Heap into the portion of RAM used as a cache, called DBCache. DRAM must also contain another area for an application's local data; this area is called the Dynamic Heap.

Devices where the system image resides in NOR flash memory (Treo 700p and Treo 755p smartphones only) do not go through the copying and uncompressing process.

Because DRAM is of a fixed size, and because it must contain the system image, DBCache, and Dynamic Heap, the size of the uncompressed system image directly affects the amount of DRAM memory available for the DBCache and Dynamic Heap. Since it is desirable to have the DBCache and Dynamic Heap be as large as possible, there is a mechanism that does just that, by reducing the size of the system image.

This is accomplished by having a separate area of NAND which contains compressed versions of applications, libraries, default content, and language overlays. This area is called *Overflow*. This Overflow area is flashed into the NAND during manufacturing, or a system image update, and not changed dynamically. Its size, and content, is set exclusively by Palm.

The Treo 700p and Treo 755p smartphones do not contain a NAND Overflow because it has NOR memory. They contain 64MB NAND, most of which (61.8MB) is available to the user for storage.

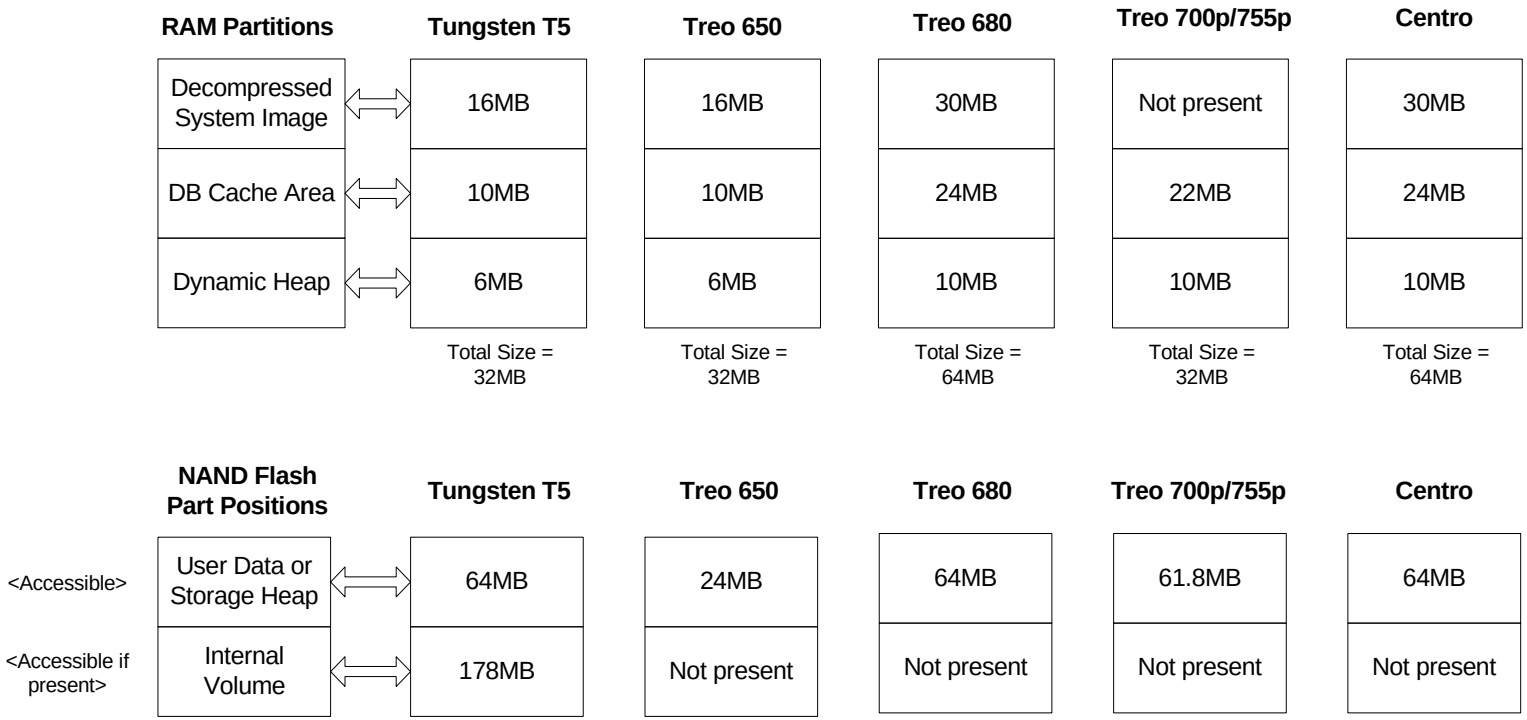
At a hard reset, the system finds NAND Overflow and uncompresses the content from this area, and then saves it in the Storage Heap (which resides on NAND). This decreases the amount of Storage Heap space remaining available (by requiring that uncompressed versions of this data reside in the Storage Heap), but frees up an equivalent amount of memory in the system image, since these applications, libraries, and language overlays would otherwise be part of the system image. This mechanism thus allows more space for the Dynamic Heap and the DBCache on the fixed-size DRAM.

When an application is launched, records and resources are copied from flash, as required, into DBCache. In addition, when a database is opened by an application, it is copied from flash into DBCache and is accessed directly from DBCache. All application access to its databases goes directly to DBCache only; it does not know that there is flash storage behind it. Because the cache is write-back, writes to DBCache do not immediately propagate back to flash. Instead the OS writes back those changes at the following occurrences: the database is closed, the system goes to sleep (auto off or user powers the device off), a call is made to `DmSetDatabaseInfo`, or a call is made to the NVFS API (`DmSyncDatabase`). At these times, the database is scanned for “dirty” records, and those records are written back to flash.

When the system goes to sleep, it makes sure that database changes are committed to flash memory. The system commits changes to NAND flash when it receives a sleep notification and does this at a normal priority. Applications that make database changes in response to sleep notifications should do so at a higher priority so the system can commit the changes. The system also commits changes back to flash memory when the database is closed and when `DmSetDatabaseInfo()` or `DmSyncDatabase()` is called.

The following diagrams illustrate how the RAM and NAND flash are partitioned. All memory sizes shown in the figures are approximate values.

RAM and NAND Flash Partitioning by Device



10.8.1 Differences between NOR and NAND flash memory

The NOR flash or masked ROM houses the system image (ROM) while RAM houses the dynamic and storage heaps. The RAM contains the dynamic heap and a DBCache area that acts as a temporary storage heap. The actual databases are stored in the NAND flash memory as files and are brought into the DBCache area when opened.

For devices where the system image is compressed and stored in NAND flash, during boot time the system image (ROM) is decompressed and brought into RAM storage. Then the Palm OS software starts executing OS code from the RAM. The portion of RAM that has the decompressed system image is read only so that users cannot corrupt OS code (see the preceding figure). The RAM also contains the dynamic heap as in the past. The compressed system image in NAND flash is not visible or accessible to users or developers.

Devices where the system image resides in NOR flash memory, such as Treo 700p and Treo 755p, do not go through the copying and uncompressing process.

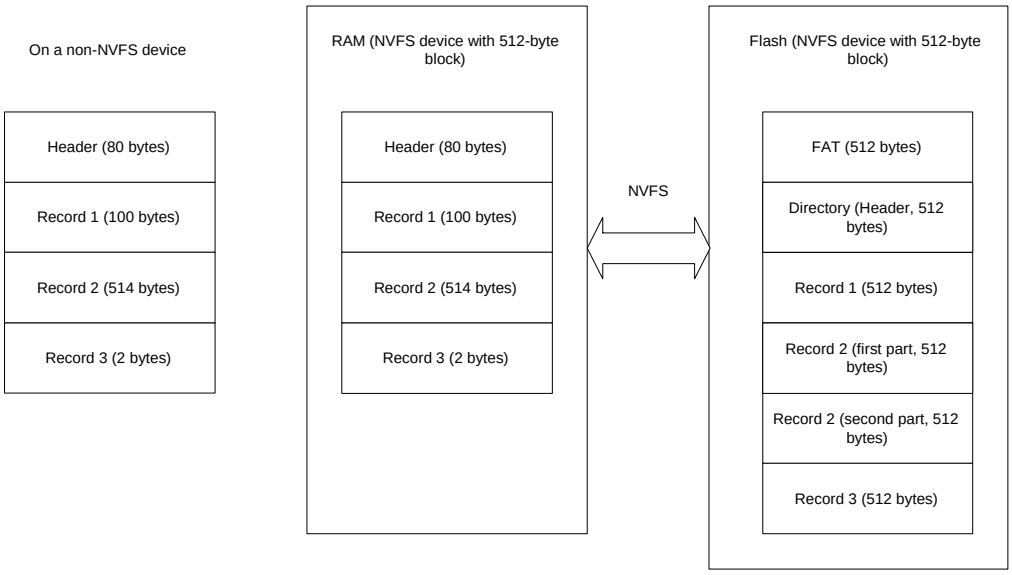
Performing a soft reset wipes the RAM clean and retrieves a fresh load of the system image from the NAND flash. A hard reset additionally erases the storage heap that is in the NAND flash.

The NAND flash is formatted into multiple partitions. Compressed system image is stored in one partition, User Store is stored in a second partition, and there can be an optional third partition as shown in the preceding diagram. The Palm OS software sees these partitions as nonremovable volumes accessible using `VFSMgr` calls. Data in the flash is stored as FAT files. The volume that houses the storage heap is hidden in all devices and can be accessed only by using a special private flag while enumerating the volumes. This volume is known as the *private volume*. Some devices include a partition that can be accessed similar to an SD slot. In Tungsten T5 handhelds, this partition is called INTERNAL. In the LifeDrive mobile manager, it is called LIFEDRIVE. Users can rename this partition like any other volume. To find this partition, look for the nonhidden, nonremovable volume.

The size of the resource database is limited to the size of the DBCache. For large record databases, the OS intelligently purges or flushes data from the cache to free space for new records: If the DBCache becomes full, the OS purges records that are not locked in memory. If the records are modified, they are committed back to flash memory. The purging of data takes place in the background and is seamless to the user. You should be aware that there are some performance issues associated with this method. There may be a noticeable time difference in performance compared to earlier devices when a large amount of data is flushed back from the cache into the NAND flash memory.

10.8.2 Database layout on NVFS devices

The figure below illustrates how the NVFS Database Management Solution works on NVFS devices (with 512-byte blocks) for a database with three records. Also see **Section 10.8.2.1**.



NVFS includes a mechanism for storing Palm OS databases in a FAT (file allocation table) file system. In the NVFS device, the file system is applied to flash (non-volatile) memory, also called the flash drive.

In this FAT file system, each file consists of multiples of 512-byte sectors. Each file is allocated in multiples of four sectors (for FAT16). This means that each file occupies multiples of 2K bytes in the file system. Each Palm OS database is stored in its own file in the file system on the flash drive. Additional overhead in the FAT system for tracking and managing the file system is independent of how the databases are laid out within it.

The header of the database and the record list occupy one or more sectors. RecordInfo entries that do not fit in the first sector become the extended record list and are allocated to other sectors, as needed. The additional sectors do not need to be contiguous.

For example, a database that contained 100 records would look something like this:

Sector 1

Database Header

RecordInfo 1

RecordInfo 2

RecordInfo 3

...

RecordInfo 39

Sector 2
RecordInfo 40
RecordInfo 41
...
RecordInfo 85

Sector 3
RecordInfo 86
RecordInfo 87
...
RecordInfo 100

The Database header includes the name, type, creatorID, and so on. The RecordInfo is a fixed size and includes the recordID, attributes, and other system information but does not include the actual record data, which is variable in length. Note that in this example the third sector is only partially used, leaving some wasted space.

The record data is kept separately, and its layout in the file sectors depends on the version of NVFS in use. Version 5.4.5 (included with the original Treo 650 smartphone and Tungsten T5 handheld) stores the data for each record in one or more sectors. For example, if the first record contained 60 bytes of data and the second record contained 700 bytes of data, the database would look something like this:

Sector 4
RecordData 1 (60 bytes)

Sector 5
RecordData 2 (first 512 bytes)

Sector 6
RecordData 2 (remaining 188 bytes)

In this example, the fourth and sixth sectors are partially used, leaving some space wasted.

10.8.2.1 Database layout

Palm OS 5.4.7 and later allows data from more than one record to share a sector. Logically the sector is divided into 16 sub-blocks of 32 bytes each. Then the record data is stored into a sector based on alignment of the data length with the available sub-blocks of the sector.

The number of sub-blocks is determined by the length of the record data. The length is rounded up to the nearest power of 2 to determine the number of sub-blocks. The placement of the data in the sector depends on this number and an associated alignment within the sector. For example, if the record data fits within a single sub-block, it can be positioned at any of the 16 sub-block alignments in the sector. If the record data requires two sub-blocks, then it can only be placed at the even-numbered sub-blocks, that is, on 64-byte alignments. If four sub-blocks are required, it can only be placed at 128-byte boundaries, and so on. As before, record data longer than 512 bytes will occupy as many complete sectors as required with the remaining bytes, following the rules above.

Using the same example, the database would look something like this:

Sector 4
RecordData 1 (60 bytes, occupying 64 bytes)
RecordData 2 (remaining 188 bytes, occupying 256 bytes)

Sector 5
RecordData 2 (first 512 bytes)

Note that in this example the fourth sector is partially used, leaving some space wasted, but there is no sixth sector used at all. In addition, if a third record is added that has less than 129 bytes of data, it can also be placed in Sector 4.

Generally, the layout with version 5.4.7 will have substantially less unused space for record data storage compared to version 5.4.5.

As records are added, changed, and deleted in the database, holes of unused space will develop in the file sectors. The NVFS system will monitor the total size of these holes, and if it gets too large, the file will be compacted. Compactions will reorganize the database records within the file sectors to fill in the holes and reduce the wasted space.

10.8.3 Programming on devices that have NVFS

The changes in how NAND flash works, as opposed to NOR flash, require some changes in how you program for NVFS devices. Although all applications should run without much modification, some cases may require special handling, such as those that deal with very large databases (over 5MB). Some of these cases are discussed in this section.

10.8.3.1 Checking for NVFS

To check if a device includes NVFS, use the following command:

```
FtrGet (sysFtrCreator, sysFtrNumDmAutoBackup, &returnVal);
```

If the `returnVal` is `1`, NVFS is present on the device. `sysFtrNumDmAutoBackup` is defined in the `PmPalmOSNVFS.h` header file.

10.8.3.2 Database issues

One of the issues with the DBCache area is that if a device loses power (for example, if the battery is removed from the Treo 650 smartphone) when a user is modifying a database, all modifications are lost, because database changes are committed back into the NAND flash only when the database is closed.

NOTE: There is no way to determine programmatically that the battery is about to be removed.

To overcome this problem, you might want your applications to call the `DmSyncDatabase()` API. This API ensures that database changes are committed to NAND flash. (Note that the `DmSyncDatabase()` API has nothing to do with the HotSync® feature.) `DmSyncDatabase()` takes a reference to the open database that needs to be synchronized. To use `DmSyncDatabase()`, you need the `PmPalmOSNVFS.h` header file from the Palm OS SDK.

Use the following guidelines for developing applications for NVFS devices:

- If you call `DmSyncDatabase` or `DmCloseDatabase`, make sure to check for returned errors and clean up appropriately.
- Always check for errors from DataMgr calls that allocate memory, for example, calls such as `DmNewRecord` and `DmResizeRecord`.
- Be aware that the database cache guarantees that individual records are saved atomically. That is, if the device runs out of space in the middle of writing out the 200th modified 1k record in your database, you should expect the first 199 changed records to be saved, but changes to the 200th record will be completely lost rather than partially saved. If your application requires that the entire database be saved atomically (for example, if your application creates another application, or if modifications in one record depend on modifications in another record), your application must be aware of these failure scenarios and guard against them.

If there is space in DBCache while the storage heap is full, database creation and modification may work, but `DmCloseDatabase()` may fail because the database data cannot be committed to the storage heap.

To check the free bytes or size of DBCache, use `MemHeapFreeBytes()` and set the high bit of the heap ID as follows:

```
#define STORAGE_HEAP_ID 1
MemHeapFreeBytes(STORAGE_HEAP_ID | dbCacheFlag, &free, &max);
```

The call `MemHeapFreeBytes(STORAGE_HEAP_ID, &free, &max);` returns the free bytes in the storage in the NAND flash part of the memory (User Store).

To discover the total size of the DBCache area or User Store, you can use the Palm OS API `MemHeapSize()` with the same values for the `STORAGE_HEAP_ID` described previously.

Applications that work with resource databases larger than the size of the DBCache area cannot directly access files used to represent the databases on the internal volume, due to a special file format not readable by the application or application developer. As described in the next section, you can split the data in databases to span multiple databases, or access the file system to read and write to files instead of using databases.

10.8.3.3 Accessing the internal file or private volumes in NVFS

The volume housing the storage heap can be accessed using the `VFSVolumeEnumerate()` function as follows:

```
#define vfsIteratorStart              0L
#define vfsIteratorStop               0xffffffffL

UInt32 volIterator = vfsIteratorStart | vfsIncludePrivateVolumes; //
0x80000000L is the flag passed in to include the private volume...

while (volIterator != vfsIteratorStop)
{
   if ((err = VFSVolumeEnumerate(&otherVolRefNum, &volIterator)) ==
errNone) {
              err = VFSVolumeInfo(otherVolRefNum, &volInfo);
              if (err)
                   goto Done;
              if (volInfo.attributes & vfsVolumeAttrHidden)
              {
              // This is an internal file volume. Perform actions now...
              }
          }
}
```

After you retrieve the volume reference number, the volume can be accessed by `VFSMgr` as with any other volume. If the `vfsIncludePrivateVolumes` flag is omitted, the private volume is not enumerated. So, for example, in a Treo 650 smartphone or later, the preceding code enumerates a private volume and the SD slot's volume. `vfsIncludePrivateVolumes` is defined in the `PmPalmOSNVFS.h` header file.

Applications that must access the private volume directly to store files in the FAT file system can do so if they are having problems working with large databases because of the DBCache size. You should use the private volume sparingly, however, because overuse cuts into the user storage space.

Similarly, you can use the Expansion Manager to see the private volume as a separate internal slot if it is used with the following flag:

```
#define expIteratorStart              0L
#define expIteratorStop               0xffffffffL

UInt32 slotIterator = expIteratorStart | expIncludePrivateSlots; //
0x80000000L is
flag passed in to include private slot;

while (slotIterator != expIteratorStop)
{
          if ((err = ExpSlotEnumerate(&slotRefNum, &slotIterator)) ==
          errNone)
{
             // Perform actions now...
          }
}
```

Other than the private volume, some devices may include a separate internal file volume that can be used for file storage. LifeDrive and Tungsten T5 handhelds, for example, have an internal file volume that is available as an internal drive. The following code enumerates two volumes: the internal file volume and the SD slot volume. You can use `vfsVolumeAttrNonRemovable` to check whether a volume is nonremovable, thus indicating that it is an internal volume. You need the `PmPalmOSNVFS.h` header file from the Palm OS SDK to do so. The ACCESS® SDK may not have the flag defined. You can use the `vfsIncludePrivateVolumes` flag to access the third private volume in a Tungsten T5 handheld.

```
UInt32 volIterator = vfsIteratorStart;

while (volIterator != vfsIteratorStop)
{
   if ((err = VFSVolumeEnumerate(&otherVolRefNum, &volIterator)) ==
errNone) {
          err = VFSVolumeInfo(otherVolRefNum, &volInfo);
          if (err)
            goto Done;
          if (volInfo.attributes & vfsVolumeAttrNonRemovable)
          {
             // This is the internal file volume. Perform actions now...
          }
       }
}
```

Similarly, you can use the Expansion Manager to access the slots by using the capability flags `expCapabilityNonRemovable` and `expCapabilityHidden` from the `ExpCardInfoType` structure to check the slot details. You may need the `PmPalmOSNVFS.h` header file from the Palm OS SDK for these flag definitions.

10.8.3.4 Feature pointer issues

Because feature pointers are allocated in the DBCache, it is possible to run out of space if you open a very large database. Even if the user store in the NAND flash has free space, a feature pointer call such as `FtrPtrResize` can fail if the DBCache is full. Query the free bytes in DBCache and take the appropriate actions as mentioned in the preceding sections if you allocate large feature pointers or if you are using feature pointers while working on large databases.

10.8.3.5 NVFS on Palm OS® 5.4.9

Available on:

- Centro™ and Treo™ smartphones
- LifeDrive™ mobile manager
- Tungsten™ T5, Tungsten™ E2, and Palm® T|X handhelds
- Palm® Z22 organizer

NVFS on Palm OS v. 5.4.9 and later provides the following enhancements:

- Alerts for extreme low memory condition

  The system warns users more aggressively when the device storage heap is low on space. Please note that if the storage heap is already full and there is still data to be flushed, the data will be lost.

- Minimization of memory fragmentation

  The occurrence of `memErrNotEnoughSpace` should be significantly minimized as the result of better compaction of DBCache. This optimization in turn makes it easier for an application to exhaust the available memory. Applications should still check for available memory and chunk size whenever possible.

- Slower sorting

  Optimizations that are done in OS 5.4.9 have caused the sorting performance to be slower as a trade-off, but this should only be visible when a huge number of records are involved.

- `DmQueryRecord` is safe to use again

  The system should now be able to recover invalid handle and read the record data back from the non-volatile memory.

- More aggressive DBCache flushing

  Locked records are now unlocked automatically when a database is closed, including resource record. Closed databases and unlocked records are flushed more aggressively whenever more space is needed. Applications that are implementing callbacks or alarm procedures should be extremely careful and make sure that the code resource is locked at all times and protected with `DMDatabaseProtect()` when expected to run.

- Bug fix for `FtrGet(sysFtrCreator, sysFtrNumROMVersion, &romVersion)`, which now returns the correct OS version and build number.

NOTE: For information on how to lock code resource and protect a database, refer to the ACCESS (formerly PalmSource) Developer Knowledge Base.

Due to some changes in NVFS for the Treo 700p, 755p, and Centro smartphones, when the device runs low on storage space, one of the following three things will happen (depending on what was happening when space ran out):

1. If the device runs out of space during `DmCloseDatabase`, the OS will back out the changes that were made since the last sync and return an error to the calling application. The user will then see an error alerting them that some data could not be saved because of low storage space.
2. If the device runs out of storage while it is flushing the cache to make room for a new allocation (for example, during `DmNewRecord` or `DmResizeRecord`), it is possible that the application that owns the database that is being flushed may not be the application doing the allocation. Since there's no way to notify the owner that the data was lost, there is no safe way to return the system to a stable state after backing out changes. In this scenario, the device will reset, and display the out-of-space alert after the reset.
3. If the device encountered an error during an explicit call to `DmSyncDatabase`, the OS will back out changes that were made since the last sync and return an error such as `vfsErrVolumeFull` to the caller. (No alert.)

10.8.4 Optimizing your application for Palm® NVFS devices

Before the introduction of the Non-Volatile File System (NVFS), data on a Palm device was stored in "volatile" memory that required a constant stream of low power to maintain data. If a device's battery became completely drained, data would be lost until a HotSync operation restored it from a PC.

Beginning with the Treo 650 smartphone and the Tungsten T5 handheld, Palm devices use "non-volatile" memory, which means that they don't need power to store data. A device's battery can be completely drained, or swapped out, and data will remain in the device.

Applications and data may take up more space on NVFS devices than devices with traditional "volatile" memory. For this reason, developing applications for NVFS devices may require some additional work to avoid performance issues.

10.8.4.1 Speeding up actual performance

Applications on most NVFS devices reside in flash memory and must be read from flash memory at first launch to ensure that they are loaded into DBCache in SDRAM. (For LifeDrive mobile managers, applications reside on the hard drive.) Therefore, there is more traffic the first time an application is launched after a reset.

When developing applications for any NVFS device, consider the following methods of optimization:

- Do not use `PrefSetPreferences` and `PrefSetAppPreferences` to store data.

  Many applications use preferences to indiscriminately store application data. On a RAM-based system, this has little overhead, but on NVFS-based systems it has significant overhead.

  Use Features rather than Preferences to store data that does not need to be preserved across a soft reset.

  It is better to make a single `PrefSet` call with all of your changes (at `StopApplication` for instance) than multiple `PrefSet` calls.

Note that a `PrefSet` call will not generate NVFS traffic if the set preference is exactly the same as the existing preference. However, if you are using buffers, make sure they are zeroed so that changing garbage data at the end of the buffer will not cause spurious writes.

These techniques also apply to the Connection Manager and the Exchange Manager.

- Use cache instead of rereading from a file.

  In many instances, applications constantly reread the same information from NVFS files (especially if they use the Media libraries). If your application is reading from a database, that will already be cached in the DBCache, but if you are using raw file data on a read-only basis, consider using a cache in either feature memory or using a FileStream.

  In particular, opening and closing the media library is a time-consuming (2-3 seconds) task. To avoid the cost of continually opening the media library, open the library, generate the bitmaps you will need (such as backgrounds), and cache those bitmaps to feature memory or temporary databases.

- Improve memory management.

  Using a feature pointer instead of a normal memory pointer has significant overhead if you are doing many writes because `DmWrite` is much slower than a normal memory access.

  Using a temporary database (or filestream) to store data can really slow things down because every `DmWrite` must be written to the backing store when the database closes. This is useful when you need to preserve your temporary data across resets and application launches, but can be a performance drain.

  If you are caching a small amount of data (<0.5MB), use heap-based pointers from `MemPtrNew`. If you need to cache a larger amount of data (>0.5MB), use `FtrPtrNew`.

- Use `DmWrite`, `FileWrite`, and `VFSFileWrite` only when necessary.

  Written data must be automatically saved to the disk, so it can become costly if you write little bits of data at a time. Your application will perform better if you make fewer larger writes rather than many small writes.

10.8.4.2 Speeding up perceived performance

- One of the things that users report in terms of performance is that the screen goes white or flickers while applications are loading or doing other significant work.

  Use `WinScreenLock/WinScreenUnlock` to freeze the UI on the screen while you are doing background processing that might make the screen flicker. This is also useful to reduce the appearance of a slow redraw time since you can redraw everything and then put it on the real display in one go.

- Process data at `sysAppLaunchCmdReset` time.

  Much of the time spent by applications launching for the first time after a reset is in recreating default databases. To save time, you may create default databases at `sysAppLaunchCmdReset` time instead of at `sysAppLaunchCmdNormalLaunch` time. That way, much of the initial application launch times are buried within the longer reset time (which already has UI).

- Keep the event loop running.

Slow UI is due in part to the fact that almost all of our libraries and applications are single-threaded. When a long blocking operation is initiated, the event loop (which updates all UI) doesn't run. This means that user input is ignored until the blocking operation is completed.

- Display "*Please wait*" dialogs.

  There are circumstances where you cannot avoid a lengthy delay (connecting to a network service, etc.). Instead of having a non-responsive UI or blank screen, use shell applications to pop up a "*Please wait*" dialog or do a `WinScreenLock()` while an application with many code resources or a lot of startup processing is loading. This shell can be as simple as being in the first code segment, typically running at `sysAppLaunchCmdNormalLaunch` in `PilotMain`. Consider using a pop up a message or progress indicator so the user doesn't think they have crashed or hung while the rest of the application processing is running. Obviously, this is a solution of last resort after exhausting other avenues to speed up an application.

10.9 5-Way Navigator and Keyboard API

Available on:

- Centro™ and Treo™ smartphones
- Tungsten™ and Palm® T|X handhelds
- Zire™ 31 and Zire™ 72 handhelds
- Palm® Z22 organizer

This section describes the software associated with the 5-way navigator button.

In addition to the information provided here, you can refer to the sample code in the Palm OS SDK for information on how to use this module.

10.9.1 5-way navigator terminology

These terms are used in the following section:

- **Object focus mode** refers to the state when a form's focus is enabled and the 5-way keys are used for navigation.
- **Application focus mode** refers to the state when a form's focus is not enabled and Up and Down are used as page-up and page-down keys.
- **Tab order** refers to the horizontal ordering of the objects or, in other words, the order in which objects receive the focus when Right is pressed repeatedly.
- **Action buttons** refers to the command buttons that are lined up along the bottom of a form.
- **Interaction mode** refers to the state when the 5-way keys interact with the object rather than move the focus. For example, a field object is in interaction mode when the 5-way keys move its insertion point, and a list object is in interaction mode when the 5-way keys move a highlight through the list items.

10.9.2 Overview of 5-way navigator

The navigation model is two-dimensional. Left and Right move the focus horizontally, while Up and Down move the focus vertically.

Depending on the object type of the object that has the focus, Center either simulates a tap on the focused object or toggles the interaction mode of the focused object.

To support this functionality, the system was expanded to generate and handle additional navigation events. These include 5-way key events and the focus change events. The events are handled when an application calls `FrmDispatchEvent()`. During such a call, the events are handled in the following manner. (Note that steps 2 and 3 are part of `FrmHandleEvent()`.)

1. The current form's custom event handler receives the event.

   This is when the application can override default navigation behavior. If the handler returns `true`, no more event processing occurs and steps 2 and 3 are never reached.

2. If the event is a key event, the focused object's type is obtained. If the event is a focus change event, the type of the object specified in the event is obtained.

   The handler specific to the object type obtained is then called on the event. (For example, `CtlHandleEvent()` is called if the object is a control.) The various object type handlers have been expanded to handle navigation events that are associated with type-specific navigation behavior. If the handler returns `true`, no more event processing occurs and step 3 is never reached.

3. A generic focus handler is called on the event. This handler is primarily responsible for moving the focus in the form.

10.9.3 Navigation events

The Palm smartphones and the Tungsten T5 handheld not only generate `keyDown` events for keys, but they also generate `keyHold` and `keyUp` events for keys. A `keyHold` event is generated when a key is held for one second and a `keyUp` event is generated when a key is released. `keyHold` and `keyUp` events have the same fields as `keyDown` events—character field, modifiers field, and key code field.

When object focus mode is on, the 5-way button generates `keyDown`, `keyHold`, and `keyUp` events with the following character values: `vchrRockerUp`, `vchrRockerDown`, `vchrRockerLeft`, `vchrRockerRight`, and `vchrRockerCenter`.

When object focus mode is off, the 5-way button generates `keyDown`, `keyHold`, and `keyUp` events with the following character values: `vchrPageUp`, `vchrPageDown`, `vchrRockerLeft`, `vchrRockerRight`, and `vchrRockerCenter`.

The page keys are enqueued when object focus mode is off so that forms that are not 5-way navigator–aware do not lose their paging functionality.

Focus change events are generated as the navigation focus moves in a form. A `frmObjectFocusTake` event is sent when an object should take the focus. The system's internal focus structures are updated when the event is *handled*, not when it is sent. A `frmObjectFocusLost` event is sent after an object has lost the focus. The system's internal focus structures have already been updated when this event is sent. Therefore, it simply initiates the redrawing of the object that lost the focus.

10.9.3.1 Option and Shift modifiers

Palm smartphones have Option and Shift keys. If the Option key is held down while the 5-way button is pressed, the `optionKeyMask` in the 5-way button's key events is set. Similarly, if the Shift key is held down while the 5-way button is pressed, the `shiftKeyMask` in the 5-way button's key events is set.

These masks allow applications to assign secondary features and functionality to the 5-way button on Palm smartphones.

10.9.4 Including objects as skipped objects

Editable fields, tables with fields, pop-up triggers, and selector triggers automatically get navigation focus when the user taps them. If the object is not in the tab order, then the system simply moves the focus to the first object in the tab order when the user presses a directional button after tapping the object. Such a movement of the focus may not make sense to the user.

To ensure that focus movement is always logical to the user, such objects should be included in the tab order but marked as skipped by setting the `kFrmNavObjectFlagsSkip` flag. This flag can be set through `FrmSetNavEntry()`, through `FrmSetNavOrder()`, or in a navigation resource. The object is skipped when the user moves the focus with the 5-way buttons, but the system knows how to move the focus from the object after the user taps it.

10.9.5 Default navigation

If a form does not have a navigation resource, the system determines the navigation order for the form dynamically. It determines whether the form is initially in object focus mode or application focus mode, which UI objects can receive the focus, the tab order, the vertical order, where the focus begins, and whether the focus cycles.

IMPORTANT: To truly support 5-way navigator, an application should have `fnav` resources in its resource file rather than rely on the default navigation order.

10.9.5.1 Initial focus mode

Forms are separated into modal and nonmodal forms. The window of a modal form has the modal flag set. You can check this flag by calling `WinModal()` on the form's window. All other forms are considered nonmodal.

Initially, modal forms are placed in object focus mode, and nonmodal forms are placed in application focus mode. You can programmatically change the mode of a form with `FrmSetNavState()`.

10.9.5.2 UI objects included in the navigation order

All UI objects in a form are included in the navigation order. However, during form initialization, only the following objects are not marked as skipped:

- Usable, enabled controls (controls include command buttons, push buttons, check boxes, pop-up triggers, selector triggers, repeating buttons, sliders, and feedback sliders)
- Usable lists
- Usable, editable fields

NOTE: An application can clear the skipped flag for any object in the form by calling `FrmSetNavOrder()` or `FrmSetNavEntry()`.

If an object is not initially included in the default order, it is not included later if it becomes a valid focus object. For example, if a control that was initially disabled is enabled, it is not included in the default order when it is enabled.

However, objects that were initially included in the default order that later become invalid focus objects are skipped. For example, if a field was initially editable and was later made noneditable, it is skipped.

10.9.5.3 Tab order

The tab order is determined by taking the UI objects that can receive the focus and sorting them by their left coordinate and then their top coordinate. A stable sort is used (insertion sort) so that after sorting:

- Objects are sorted by their top position.
- Objects with the same top position are sorted by their left position.

10.9.5.4 Vertical order

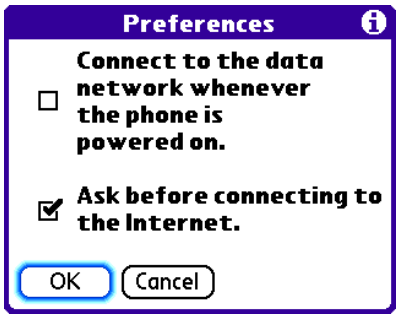
The objects in the vertical order are a subset of those in the tab order. The vertical order is determined by iterating from the first object in the tab order to the last and including only the following objects:

- The first object
- Any subsequent object that is completely below the previously included object

This algorithm results in only the leftmost objects of the form being included in the vertical order.

Because the vertical order may not include all the objects in the tab order, some objects that can receive the navigation focus cannot be navigated to by only using the Up and Down keys. In other words, navigating to objects that are in the tab

order but not in the vertical order will require some use of the Left or Right key. For example, consider the following dialog box:



The check box and the OK button are the leftmost objects in their row and therefore are the objects in the vertical order. To get to the Cancel button from the top check box, the user would have to press Down and then Right. There is no way to get to the Cancel button by using only the Up and Down keys.

If an object that was initially usable and was part of the vertical order is later made nonusable, and an application has not made any changes to the order through the navigation API functions, the vertical order is recalculated using the algorithm previously described. The vertical order is not recalculated when an application has made changes to the order, to ensure that the application's changes are not overridden.

10.9.5.5 Initial focus

If there are any action buttons, the focus is initially given to the leftmost one. If there are no action buttons, the focus is given to the first object in the tab order.

The first action button is identified using the following rules:

- The usable, enabled objects with the greatest top coordinate values are identified. In other words, the objects in the form's bottom row are identified.
- The first action button is the leftmost command button among those objects. If there are no command buttons among these objects, the focus is given to the first object in the tab order.

10.9.5.6 Cycling

On Palm smartphones and the Tungsten T5 handheld, the focus never cycles horizontally. When the focus is on the first object in the tab order, pressing Left does not move the focus to the last object, and when the focus is on the last object, pressing Right does not move the focus to the first object.

Although the focus never cycles vertically on Treo 600 smartphones, the focus does cycle vertically in modal dialog boxes on Treo 650 smartphones and later, and Tungsten T5 handhelds. On Treo 600 smartphones and in nonmodal dialog boxes on Treo 650 smartphones and later, and Tungsten T5 handhelds, when the focus is on the top object in the vertical order, pressing Up does not move the focus to the bottom object, and when the focus is on the bottom object, pressing Down does not move the focus to the top object. In modal dialog boxes on Treo 650 smartphones and later, and Tungsten T5 handhelds, pressing Up while the focus is on the top object *does* move the focus to the bottom object, and pressing Down while the focus is on the bottom object *does* move the focus to the top object.

10.9.6 Custom navigation

Applications can customize navigation by providing a navigation resource for a form, by making navigation API calls, and by handling navigation events.

10.9.6.1 Hex navigation resource

A navigation resource specifies what UI objects in the form are included in the navigation order, what the tab order is, what the vertical order is, whether the focus is initially on or off, where the focus begins, where the focus can move, and what the bottom-left object is. (The bottom-left object information is needed to cycle the focus from the top of the form.)

IMPORTANT: As mentioned before, an application should include `fnav` resources in its resource file rather than rely on default navigation order.

For the system to detect a navigation resource, the navigation resource must be in the same database as its associated form. For the system to detect that a form has a navigation resource, the navigation resource must be in the same database as the form.

Technically, an application only needs a navigation resource if the behavior is not the correct behavior for a form. However, the creation of navigation resources for all forms that have navigation is recommended because the default navigation order may be different on various platforms and some platforms may not even provide a default navigation order.

The resource is a hex resource of type `formNavRscType` (`fnav`). It is defined as a 68K format (big endian) resource.

The term "hint" is appended to the name of those fields that are used only by some platforms. For example, the `bottomLeftObjectIDHint` is used only by those platforms that have their focus cycle from the top of the form to the bottom. If an application specifies a value for this field, it will run properly on platforms that do cycle and platforms that do not cycle.

The navigation resource has a header section and a list-of-objects section. The format of the header is described in the *Palm API Guide, Palm OS Platform*.

10.9.6.2 PilRC navigation resource

The PilRC resource compiler supports a navigation resource format that is easier to create than the HEX resource just described. Many of the fields required by the hex resource are optional in the navigation resource and a navigation map that allows UI objects to be specified in row-column fashion is supported. This format is supported starting with PilRC 3.0 and is documented in the manual packaged with PilRC. PilRC can be obtained at **pilrc.sourceforge.net**.

10.9.6.3 Objects that become nonusable

The system skips over nonusable objects when a user moves through the tab order. Objects that reside in the same position in the form and are alternately shown should therefore be placed next to each other in the tab order and have the same above and below objects.

If an object in the vertical order becomes nonusable, and the user navigates up to it, the object above receives the focus instead. The down case works the same way, except that the object below the nonusable object receives the focus. If you want an application to behave differently, use the API calls to explicitly set which object should replace the nonusable object in the vertical order.

10.9.6.4 Handling navigation events

System navigation behavior is executed when `FrmHandleEvent()` receives a navigation event. Because `FrmDispatchEvent()` calls a form's custom event handler before calling `FrmHandleEvent()`, a form can easily override default navigation behavior by handling navigation events in its custom event handler.

As explained earlier, the system's internal focus structures are updated when a `frmObjectFocusTake` event is handled, not when it is sent. Therefore, an application must explicitly call `FrmSetFocus()` on the event's associated object if it handles a `frmObjectFocusTake` event.

A `frmObjectFocusLost` event is sent after an object has lost the focus. The internal focus structures will have already been updated when this event is sent. Therefore, an application does not have to do anything besides implement its desired custom behavior if it handles a `frmObjectFocusLost` event.

With tables and gadgets, applications must intercept navigation events. These UI object types have minimal default behavior, if any.

For example, when an application has a table that is included in the focus order, the application might perform the following actions in its custom handler:

- `frmObjectFocusTake` event for table:

  Call `FrmSetFocus()` on the table, highlight the first row of the table, and return `true`.

- `frmObjectFocusLost` event for table:

  Unhighlight row in table and return `true`.

- An Up `keyDown` event when table has the focus:

  If the highlight is not on the table's top row, move the highlight up a row, and return `true`.

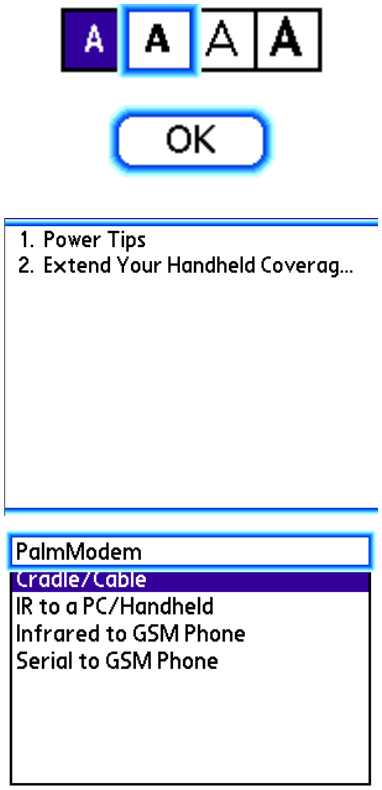
- A Down `keyDown` event when table has the focus:

  If the highlight is not on the table's bottom row, move the highlight down a row, and return `true`.

In general, it is assumed that most modal forms do not alter the default navigation behavior, while most nonmodal forms do. That is why navigation is automatically enabled only for modal forms. Nonmodal forms are usually the main views of an application and therefore require more custom behavior.

10.9.7 Focus treatment

The functions that draw UI objects are updated to know how to draw the new visual states introduced by navigation. The system uses a blue square ring, a blue rounded ring, or blue bars to indicate that an object has the focus:



`CtlDrawControl()` checks whether the control it is drawing has the navigation focus, and draws a focus ring around the control if it does.

`FldDrawField()` checks whether the field it is drawing has the navigation focus. The function then checks whether the field is in interaction mode. If it is not in interaction mode, the function draws focus bars above and below the field and does not draw the insertion point. If it is in interaction mode, the function draws the field normally and draws the insertion point.

If `LstDrawList()` is drawing an embedded list, it checks whether the list has the navigation focus. The function then checks whether the list is in interaction mode. If it is in interaction mode, the function draws a focus ring around the temporarily selected item. If it is not in interaction mode, the function draws a focus ring around the entire list.

API functions are provided if you want to draw focus rings around objects other than controls, fields, and lists in your application. On the Treo 600 smartphone, the functions are `HsNavDrawFocusRing()`, `HsNavRemoveFocusRing()`, and `HsNavGetFocusRingInfo()`. On the Treo 650 smartphone and later, and the Tungsten T5 handheld, the functions are `FrmNavDrawFocusRing()`, `FrmNavRemoveFocusRing()`, and `FrmNavGetFocusRingInfo()`.

The system ensures that no more than one ring is ever drawn on a form. If a ring is being drawn and there is already a ring on the form, the system removes the ring already displayed on the form before drawing the new ring. A ring drawn with `HsNavDrawFocusRing()` or `FrmNavDrawFocusRing()` should never be directly erased. If you want your application to remove the ring, it should call `HsNavRemoveFocusRing()` or `FrmNavRemoveFocusRing()`.

On Tungsten T5 handhelds, when the Active Input Area is collapsed or expanded, the system automatically removes any focus ring before the area is collapsed or expanded and, after the area has been collapsed or expanded, sends a `frmObjectFocusTake` event with the ID of the object that had the focus ring.

10.9.8 Navigational API, Button Mapping, and behavioral differences between Palm smartphones and Tungsten™ T5 handhelds

This section describes navigational differences between the Treo 600 smartphones and later, and the Tungsten T5 handheld.

10.9.8.1 Palm OS® features

Treo 600 smartphones and later, and Tungsten T5 handhelds, set the `hsFtrIDNavigationSupported` feature. The feature's creator is `hsFtrCreator`. The value of the feature is the version number of the Palm Navigation API. On Treo 600 smartphones the version is 1, and on Treo 650 and Treo 700p smartphones and later and Tungsten T5 handhelds the version is 2.

Treo 650, Treo 700p, Treo 755p, and Centro smartphones and Tungsten T5 handhelds also set the `sysFtrNumFiveWayNavVersion` feature. The feature's creator is `sysFileCSystem`. The value of the feature is the version number of the ACCESS (formerly PalmSource) navigation API. On Treo 650, Treo 700p, Treo 755p, and Centro smartphones and Tungsten T5 handhelds, the version is 1. Version 2 of the Palm Navigation API and version 1 of the ACCESS Navigation API are the same.

Treo 600 smartphones with software version 1.12, Treo 650, Treo 700p, Treo 755p, and Centro smartphones, and Tungsten T5 handhelds set the s`ysFtrNumUIHardwareFlags` feature. The feature's creator is `sysFileCSystem`. The value of this feature is a bit field that describes what hardware is available on the device. The bit definitions used with the feature's value are as follows:

- `sysFtrNumUIHardwareHard5Way` - The device has a 5-way rocker
- `sysFtrNumUIHardwareHasThumbWheel` - The device has a thumb wheel
- `sysFtrNumUIHardwareHasThumbWheelBack` - The device has a thumb wheel with a Back button
- `sysFtrNumUIHardwareHasKbd` - The device has a dedicated keyboard

On Treo 600 smartphones with software version 1.12, Treo 650, Treo 700p, Treo 755p, and Centro smartphones, and Tungsten T5 handhelds, the `sysFtrNumUIHardwareHas5Way` is set. On Treo 600 smartphones with software version 1.12 and on Treo 650, Treo 700p, Treo 755p, and Centro smartphones, the `sysFtrNumUIHardwareHasKbd` is also set.

NOTE: The feature not being set on the original Treo 600 smartphone software was an oversight. This problem was fixed with software update 1.12 available from the Palm Customer Support download area.

The Tungsten T5 handheld no longer supports the `navFtrVersion` feature supported on Zire handhelds and earlier Tungsten handhelds.

10.9.8.2 Functions

Treo 600 smartphones and later support `HsNavDrawFocusRing()`, `HsNavRemoveFocusRing()`, `HsNavGetFocusRingInfo()`, and `HsNavObjectTakeFocus()` calls. Treo 650 smartphones and later, LifeDrive, Palm T|X, and Tungsten T5 handhelds support `FrmNavDrawFocusRing()`, `FrmNavRemoveFocusRing()`, `FrmNavGetFocusRingInfo()`, and `FrmNavObjectTakeFocus()` calls.

Except for the prefix differences in their names, these functions work exactly the same way. Applications should transition to using the `FrmNav` API calls, because the `HsNav` calls are deprecated and remain only on Treo 650 smartphones for Treo 600 smartphone backward compatibility.

IMPORTANT: Do not make `HsNav` calls on Tungsten T5 handhelds. `HsNav` calls made on Tungsten T5 handhelds will fail, most likely with a Sys0505 error, which means that the module that exports the function is not on the handheld.

Because Treo 600 smartphones support only the `HsNav` version of these calls and Tungsten T5 handhelds support only the `FrmNav` version of these calls, applications intended to run on both devices must check their context before making these calls. The suggested method is to check the version number of the `hsFtrIDNavigationSupported` feature and decide whether to make an `HsNav` call or a `FrmNav` call based on the version value. Specifically, `HsNav` calls should be made if the version is `1`, and `FrmNav` calls should be made if the version is `2`. The decision about what call to make must be made at runtime. For example:

```
if (FtrGet (hsFtrCreator, hsFtrIDNavigationSupported, &version) == 0)
  {
    if (version == 1)
      HsNavObjectTakeFocus (formP, objID);
    else // if version >= 2
      FrmNavObjectTakeFocus (formP, objID);
  }
```

10.9.8.3 Button Mapping

This section describes the button mapping schemes for Treo smartphones.

10.9.8.3.1 Virtual Keys on Treo Smartphones

Traditional Virtual Keys for Palm Devices
vchrHard1
vchrHard2
vchrHard3
vchrHard4
vchrHardPower
vchrPageUp
vchrPageDown
vchrLaunch (usually silk-screened on non-keyboard devices)
vchrMenu (usually silk-screened on non-keyboard devices)
vchrCalc (usually silk-screened on non-keyboard devices)
vchrFind (usually silk-screened on non-keyboard devices)

Treo Specific Virtual Keys
hsChrOptHard1
hsChrOptHard2
hsChrOptHard3
hsChrOptHard4
vchrHard11
hsChrOptHard11
hsChrOptHardPower

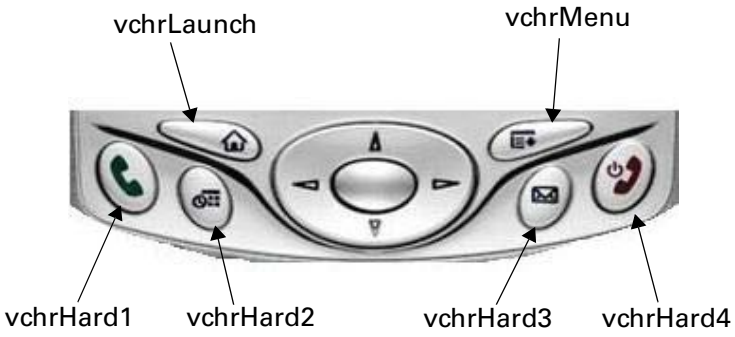
10.9.8.3.2 Button Mapping for Treo™ 600

The following table shows how the keyboard driver maps virtual characters to device keys for the Treo 600 smartphone.

NOTE: `EvtGetEvent` converts `vchrPageUp/Down` into `vchrRockerUp/Down` if navigation is enabled for the active form.

Treo 600		
Traditional Keys	**vchrHard1**	Phone key (1st key in app row)
	vchrHard2	Calendar key (2nd key in app row)
	vchrHard3	Mail/Msg key (3rd key in app row)
	vchrHard4	Screen key (4th key in app row)
	vchrHardPower	Power key (on top of device)
	vchrPageUp	Rocker-up key
	vchrPageDown	Rocker-down key
	vchrLaunch	Home key (bottom row of keyboard)
	vchrMenu	Menu key (bottom row of keyboard)
	vchrCalc	None
	vchrFind	Option + Left Shift key
Treo-specific Keys	**hsChrOptHard1**	Option + Phone key
	hsChrOptHard2	Option + Calendar key
	hsChrOptHard3	Option + Mail/Msg key
	hsChrOptHard4	Option + Screen key
	vchrHard11	None
	hsChrOptHard11	None
	hsChrOptHard Power	Option + Power key
	vchrMenu	Menu key (Right, flattened key in top row)
	vchrCalc	None
	vchrFind	Option + Left Shift key

10.9.8.3.3 Button Mapping for Treo™ 650



The following table shows how the keyboard driver maps virtual characters to device keys for the Treo 650 smartphone.

NOTE: `EvtGetEvent` converts `vchrPageUp/Down` into `vchrRockerUp/Down` if navigation is enabled for the active form.

Treo 650		
Traditional Keys	**vchrHard1**	Phone/Send key (1st key in app row)
	vchrHard2	Calendar key (2nd key in app row)
	vchrHard3	Mail/Msg key (3rd key in app row)
	vchrHard4	Power/End app key (4th key in app row)
	vchrHardPower	None
	vchrPageUp	Rocker-up key
	vchrPageDown	Rocker-down key
	vchrLaunch	Home key (Left, flattened key in top row)
	vchrMenu	Menu key (Right, flattened key in top row)
	vchrCalc	None
	vchrFind	Option + Left Shift key
Treo-specific Keys	**hsChrOptHard1**	Option + Phone/Send key
	hsChrOptHard2	Option + Calendar key
	hsChrOptHard3	Option + Mail/Msg key
	hsChrOptHard4	Option + Power/End app key
	vchrHard11	None
	hsChrOptHard11	None
	hsChrOptHard Power	None

10.9.8.3.4 Button Mapping for Treo™ 680, Treo™ 700p, Treo™, and 755p

The following table shows how the keyboard driver maps virtual characters to device keys for the Treo 680, Treo 700p, and Treo 755p smartphones.

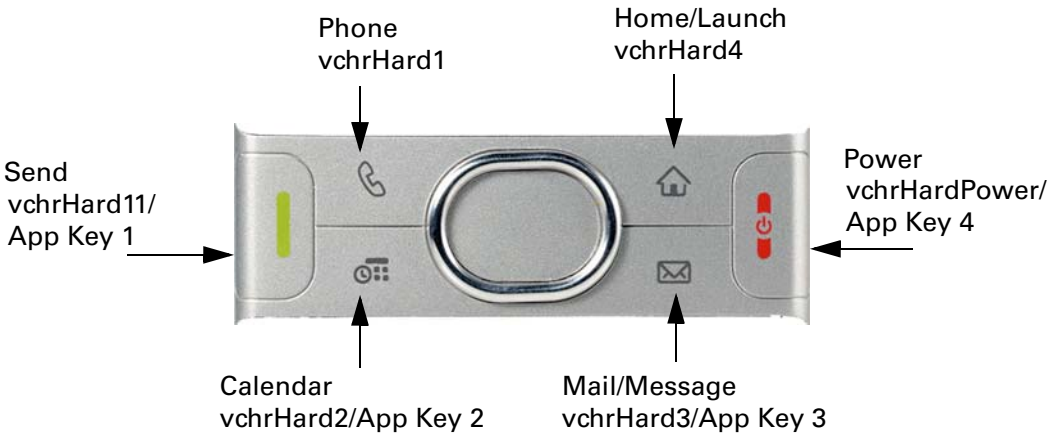
NOTE: `EvtGetEvent` converts `vchrPageUp/Down` into `vchrRockerUp/Down` if navigation is enabled for the active form.

Treo 680, Treo 700p, and Treo 755p smartphones		
Traditional Keys	**vchrHard1***	Phone (1st key in app row)
	vchrHard2	Calendar key (2nd key in app row)
	vchrHard3	Mail/Msg key (3rd key in app row)
	vchrHard4*	Home key (4th key in app row)
	vchrHardPower*	Power/End key (Right, flattened key in top row)
	vchrPageUp	Rocker-up key
	vchrPageDown	Rocker-down key
	vchrLaunch*	Option + Home key
	vchrMenu**	Menu key (bottom row of keyboard)
	vchrCalc	None
	vchrFind	Option + Left Shift key
Treo-specific Keys	**hsChrOptHard1***	Option + Phone key
	hsChrOptHard2	Option + Calendar key
	hsChrOptHard3	Option + Mail/Msg key
	hsChrOptHard4*	None
	vchrHard11*	Send key (Left, flattened key in top row)
	hsChrOptHard11*	Option + Send key
	hsChrOptHardPower*	Option + Power/End key

* Indicates differences from Treo 650 smartphones.
** The vchrMenu key is in a different location on the device, but functionality is the same.

It is important to note that launching an application in response to pressing an application key occurs in `SysHandleEvent` after a `sysNotifyVirtualCharHandlingEvent` notification is sent. Therefore, any application that processes key events before `SysHandleEvent` and/or handles `sysNotifyVirtualCharHandlingEvent` notifications could prevent application launching from working properly.

10.9.8.3.5 Button Mapping for Centro™ smartphones



The following table shows how the keyboard driver maps virtual characters to device keys for the Centro smartphone.

NOTE: `EvtGetEvent` converts `vchrPageUp/Down` into `vchrRockerUp/Down` if navigation is enabled for the active form.

Centro smartphone		
Traditional Keys	**vchrHard1**	Phone
	vchrHard2	Calendar key (App key 2)
	vchrHard3	Mail/Msg key (App key 3)
	vchrHard4	Home key
	vchrHardPower	Power/End key (App key 4)
	vchrPageUp	Rocker-up key
	vchrPageDown	Rocker-down key
	vchrLaunch	Option + Home key
	vchrMenu	Menu key (bottom row of keyboard)
	vchrCalc	None
	vchrFind	Option + Left Shift key

Centro-specific Keys	**hsChrOptHard1**	Option + Phone key
	hsChrOptHard2	Option + Calendar key
	hsChrOptHard3	Option + Mail/Msg key
	hsChrOptHard4	None
	vchrHard11	Send key (App key 1)
	hsChrOptHard11	Option + Send key
	hsChrOptHardPower	Option + Power/End key

It is important to note that launching an application in response to pressing an application key occurs in `SysHandleEvent` after a `sysNotifyVirtualCharHandlingEvent` notification is sent. Therefore, any application that processes key events before `SysHandleEvent` and/or handles `sysNotifyVirtualCharHandlingEvent` notifications could prevent application launching from working properly.

10.9.8.3.6 Option plus App key Mapping for Centro™ smartphones

Pressing the Option key plus Launcher behaves the same as pressing the Launcher key alone. Pressing the Option key plus App key 2, 3 or 4 launches applications associated by the user through Button Preferences panel in the Home Menu.

The following table shows the default Option Key plus App Key mapping.

Option + App Key	Application
Option + Phone	Web Browser
Option + Calendar	World Clock
Option + Message	VersaMail

10.9.8.3.7 Button Mapping scheme in Palm® smartphones

The button mapping scheme of the Treo 680, Treo 700p, Treo 755p, and Centro smartphones differ from the Treo 650 in the following significant ways:

1. Phone Application and Send Key

   On the Treo 650 smartphone, the Phone Application and Send functionality were on the same key. On the Treo 680, Treo 700p, Treo 755p, and Centro smartphones this has changed; Send functionality is no longer associated with the Phone Application key. To associate behavior with the Send key, you should now use `vchrHard11` instead of `vchrHard1`.

2. Home Key

   On the Treo 650 smartphone, the Home key was associated with the `keyLaunch keyCode` (or the `keyBitExt2Launcher` bit). The Home key on the Treo 680, Treo 700p, Treo 755p, and Centro smartphones are associated with the `keyHard4 keyCode` (and the `keyBitHard4` bit) instead. The Home key generates `vchrHard4` key events when it is not modified by Option. When it is modified by the Option key, the Home key generates `vchrLaunch` key events.

   It is important to note that instead of enqueueing `hsChrOptHard4` key events when the Launcher key is modified by an Option, `vchrLaunch` is enqueued. This is necessary because some games that perform pre-`SysHandleEvent` processing will only exit if they see a `vchrLaunch` key event.

NOTE: Because `vchrLaunch` is enqueued instead of `hsChrOptHard4`, any functionality associated with `hsChrOptHard4` will be lost on the Treo 680, Treo 700p, Treo 755p, and the Centro.

The Home key will switch to Launcher in two instances:

1. When it is unmodified by the Option key

   In this case, the Home key generates a `vchrHard4 keyDown` event, `SysHandleEvent` broadcasts a `sysNotifyVirtualCharHandlingEvent` notification, and then `SysHandleEvent` switches to Launcher if the notification is unhandled.

2. When it is modified by the Option key

   In this case, the Home key generates a `vchrLaunch keyDown` event, `SysHandleEvent` broadcasts a `sysNotifyVirtualCharHandlingEvent` notification, and then `SysHandleEvent` switches to Launcher if the notification is unhandled.

Additionally:

- Events generated by the Home key will have `keyHard4` in the keyCode field
- Passing the `keyLaunch keyCode` to `PmKeyKeysPressed`, `PmKeyStop`, or `PmKeyEnable` has no effect.
- Passing the `keyLaunch keyCode` to `PmKeyKeyCodeToChrCode` returns 0 for the character and modifiers.
- Passing the `keyHard4 keyCode` to `PmKeyKeysPressed`, `PmKeyStop`, or `PmKeyEnable` affects the behavior of the Home key.

- Passing the `keyHard4 keyCode` to `PmKeyKeyCodeToChrCode` returns `vchrHard4` if `optionKeyMask` is not set in the passed-in modifiers and `vchrLaunch` if `optionKeyMask` is set in the passed-in modifiers.
- Passing `vchrHard4` or `vchrLaunch` to `PmKeyChrCodeToKeyCode` returns `keyHard4`.
- Neither `vchrHard4` nor `vchrLaunch` can be remapped to a different application through the user interface or programmatically.

3. Power Key

   The dedicated power key, which was removed from the Treo 650, has been brought back on the Treo 680, Treo 700p, Treo 755p, and Centro. The dedicated power key enqueues `vchrHardPower`.

4. PmKeyAttrGet

   Several attributes have been added to `PmKeyAttrGet` that will allow an application to determine the functionality that is mapped to specific keys:

   - `pmKeyAttrGetLauncherKeyCode` returns the keyCode of the Home key
   - `pmKeyAttrGetMenuKeyCode` returns the keyCode of the Menu key
   - `pmKeyAttrGetSendKeyCode` returns the keyCode of the Send key
   - `pmKeyAttrGetEndKeyCode` returns the keyCode of the End key
   - `pmKeyAttrGetPhoneKeyCode` returns the keyCode of the Phone key

   Please note that these attributes are only available in devices that have PmKey Library API version 3 or later. Therefore, when you call `PmKeyAttrGet`, do one of the following things:

   1. Verify that the PmKey library is version 3 or later by using the following code:

      ```
      (sysGetLibAPIVersionMajor(pmKeyLibVersion) >= 3)
      ```

   2. Fail gracefully if `PmKeyAttrGet` returns `pmErrNotSupported`.

NOTE: Due to the location of the Home key, if you cannot exit a game through the Home key, Option + Home key, or the game menus, try pressing and holding the Side key. This should launch the application associated with the Side key and allow you to exit the game.

10.9.8.4 Associating custom behavior with the Center button

Treo 600 smartphones, Treo 650 smartphones and later, Centro smartphones, and Tungsten T5 handhelds generate the following key events for Center button actions:

Action	Treo 600 smartphone	Treo 650 and later Palm smartphones, and Tungsten T5 handheld
Press	`keyDown` event with `chr=vchrRockerCenter`, `keycode=keyRockerCenter`, and `modifiers=commandKeyMask`.	`keyDown` event with `chr=vchrHardRockerCenter`, `keycode=keyRockerCenter`, and `modifiers=commandKeyMask`.
Continuous Press	`keyDown` event with `chr=vchrRockerCenter`, `keycode=keyRockerCenter`, and `modifiers=autoRepeatKeyMask | commandKeyMask`.	`keyDown` event with `chr=vchrHardRockerCenter`, `keycode=keyRockerCenter`, and `modifiers=autoRepeatKeyMask | commandKeyMask`.
Held for one second or longer	`keyHold` event with `chr=vchrRockerCenter`, `keycode=keyRockerCenter`, and `modifiers=commandKeyMask`.	`keyHold` event with `chr=vchrHardRockerCenter`, `keycode=keyRockerCenter`, and `modifiers=commandKeyMask`.
Release	`keyUp` event with `chr=vchrRockerCenter`, `keycode=keyRockerCenter`, and `modifiers=commandKeyMask`.	`keyUp` event with `chr=vchrHardRockerCenter`, `keycode=keyRockerCenter`, and `modifiers=commandKeyMask`. (Consumed by `SysHandleEvent` if the system handled `keyHold` of `vchrHardRockerCenter`.) `keyDown` event with `chr=vchrRockerCenter`, `keycode=0,` and `modifiers=commandKeyMask`. (If the `keyDown` event `vchr=vchrHardRockerCenter` is not handled and the system does not handle the `keyHold` event of `vchrHardRockerCenter`.)

On Treo 600 smartphones and later, associating custom behavior with the Center button simply entails handling the `vchrRockerCenter keyDown` event. Applications can handle the other events as well, although most applications will not need to. If an application handles the other events, it is responsible for making sure that multiple actions are not triggered by the Center button. For example, an application that performs an action on `keyUp` should ensure that no action is performed on `keyDown`.

On Treo 600 smartphones, by default, a Center button press and hold does nothing different from a Center button press. On Treo 650 smartphones and later, and Tungsten T5 handhelds, however, if the Center button is pressed and held, the Attention dialog box is displayed.

As such, on Treo 650 smartphones and later, and Tungsten T5 handhelds, an application cannot associate an action with the press of the Center button because it is not yet known whether the Center button is going to be pressed or pressed and held. If an application associates an action on press and then the button is pressed and held, two actions will be triggered by the Center button. On Treo 650 smartphones and later, and Tungsten T5 handhelds, actions should occur on the *release* of the Center button and only if the Center button was not held.

To minimize the changes required to make Treo 600 smartphone applications work with Treo 650 smartphones and Tungsten T5 handhelds, on these devices a `vchrRockerCenter keyDown` event is generated on Center button release rather than on press, and it is only generated on release if the Center button was not held.

This means that an application can safely handle any `vchrRockerCenter keyDown` event it receives without the Center button triggering multiple actions. The application need not check to see if it is running on a Treo 600 smartphone, nor does it need to determine whether the Center button is held before handling the event. This also means that a continuous press `keyDown` event, a `keyHold` event, and a `keyUp` event for `vchrRockerCenter` are not generated on Treo 650 smartphones and Tungsten T5 handhelds. See the next section for what new events are generated on Treo 650 smartphones and Tungsten T5 handhelds.

10.9.8.5 Center button events for Treo™ 650 smartphones and later, and Tungsten™ T5 handhelds

`vchrHardRockerCenter` key events are generated on Treo 650 smartphones and later, and Tungsten T5 handhelds in the same fashion that `vchrRockerCenter` key events are generated on Treo 600 smartphones. Only Treo 650 smartphone and later, and Tungsten T5 handheld applications that need more information on the state of the Center button than what the `vchrRockerCenter keyDown` event provides need to handle `vchrHardRockerCenter` key events.

The system will not handle a `vchrHardRockerCenter keyHold` event if a form's custom handler, the handler associated with a form by `FrmSetEventHandler`, handles a `vchrRockerHardCenter keyDown` event. This prevents an action from occurring on both the press and hold of the Center button.

If an application did not handle the `vchrHardRockCenter keyDown` event and the system did not handle the `vchrHardRockerCenter keyHold` event, a `keyDown` event with `chr=vchrRockerCenter`, `keycode=0`, and `modifiers=commandKeyMask` is generated on release.

Even though an application can handle Center presses, doing so is not recommended because it prevents the Attention dialog box from being displayed when the Center button is held.

10.9.8.6 Page scrolling

10.9.8.6.1 Page scrolling on Treo™ 600 and Treo™ 650

On Treo 650 smartphones and Tungsten T5 handhelds, paging through lists of records or through lines of text is easier than it is on Treo 600 smartphones. On Treo 600 smartphones, when the focus is on a multi-line field that is not in interaction mode, Up and Down move the focus off the field and to the object above or below. On Treo 650 smartphones and Tungsten T5 handhelds, Up and Down still move the focus to the object above or below if the top or bottom line of text is showing. However, if the top or bottom line of text is not showing, then Up and Down pages the field's text up or down. Only after the top or bottom is reached and Up or Down is released and pressed again does the focus move to the object above or below. Users can still easily move the focus off the field by pressing the Center button if the field is in interaction mode, and then pressing Left or Right. Focus bars above and below a field convey that Up and Down behave in this manner.

Navigation behavior for tables is to be completely implemented by third-party applications. For tables in Palm applications, the same paging behavior the system uses for multi-line fields is used for Up and Down. Additionally, if a table uses Center to open a record, Left takes the table out of interaction mode. Without this extra functionality added to the Left button, there would be no easy way to move the focus off a table.

For consistency's sake and to ensure a good user experience, third-party applications should follow these navigation conventions in their tables as well.

10.9.8.6.2 Page Scrolling on Palm® smartphones

Treo 680, Treo 700p,Treo 755p, and Centro smartphones include page scrolling behaviors that are different from the Treo 650. The main difference is the FIRST page scroll. The FIRST page scroll behavior (where the page is scrolled completely) is replaced with a behavior that moves the focus to the bottom item on the page.

Page scrolling behaviors in Treo 680, Treo 700p, Treo 755p, and Centro smartphones are as follows:

1. When the user quickly presses and releases the Up or Down button, the focus scrolls a line.
2. When the user presses and holds the Up or Down button, the focus scrolls a page.
3. When the user presses and continues to hold the Up or Down button, the focus scrolls a page repeatedly.
4. When the user holds the 5-way rocker up or down for 0.5 seconds, the list will scroll 1 page and focus will be placed on the bottom item on the page (if the user is scrolling down).
5. When the user holds the 5-way rocker up or down for an additional 0.4 seconds, the list will start page scrolling every 0.4 seconds. Focus will remain on the bottom item on the page as paging occurs.

10.9.8.7 Navigation macros

The navigation macros supported on Tungsten and Zire handhelds also work on Treo 650 smartphones and Tungsten T5 handhelds. They continue to be supported to help minimize the code paths that an application needs to take to run on multiple devices. The details of these macros are thoroughly documented in the `palmOneNavigator.h` header file.

10.9.9 Tips and troubleshooting

10.9.9.1 Navigation order

- You must create a navigation resource any time they have a form whose initial navigation order is not the default navigation order. `FrmSetNavOrder()` and `FrmSetNavEntry()` are not intended to replace the use of navigation resources. A form that should initially have a custom navigation order should always have a navigation resource.

  Having the navigation information available at the time the system initializes the form is much cleaner than having the system initialize the form with the default navigation order and then having the order changed when the application performs its own form initialization. `FrmSetNavOrder()` and `FrmSetNavEntry()` are mainly for dynamically created forms or forms with navigation orders that change sometime after form initialization.

- For forms that are "navigation-aware" (that is., forms that have a navigation resource and/or call navigation API functions), the vertical navigation order is not automatically updated as object attributes are changed, as objects are included in the order, or as objects are excluded from the order.

  You may expect that the vertical order will automatically be updated because the vertical navigation order of forms that are not "navigation-aware" *is* automatically updated. However, it is not prudent to automatically determine the navigation order for all forms. Automatic updating of navigation order is applied only when necessary—when a form does not know about navigation.

  If a form knows about navigation, it is the developer's responsibility to specify the proper vertical order through a navigation resource and then update the vertical order as needed.

- Developers can enable basic navigation for existing applications by simply creating navigation resources for the application's forms and including the resources into the application's existing .prc.

- Pop-up lists do not technically receive the navigation focus. When they are not popped up, they are not usable and therefore cannot receive the focus. When they are popped up, the rocker keys have dedicated functionality—regardless of whether object focus mode is on or which object has focus. Therefore, there is no need to put pop-up lists in a custom navigation order (although no problems arise if they are placed in the order).

  If a form is using the default navigation order, the pop-up list will be included in the order but will most likely be marked as skipped (because the list will most likely not be usable when the form is initialized).

10.9.9.2 Focus

- Although `FrmSetFocus()` gives the focus to the specified object, it does not redraw the object. To give focus to a system-supported navigation object (controls, fields, or lists), an application should call `HsNavObjectTakeFocus()` on the object. `HsNavObjectTakeFocus()` sends a `frmObjectFocusTake` event for the object and `FrmHandleEvent()` processes the event by calling `FrmSetFocus()` on the object and redrawing the object.
- The effects of calling `FrmSetFocus()` to set the navigation focus will be lost if `FrmSetFocus()` is called in response to a `frmOpen` event. This is because a `frmObjectFocusTake` event that sets the form's initial navigation focus is sent just after the `frmOpen` event.

  To properly give an object the initial focus, a navigation resource with the object specified as the initial focus object should be provided.
- If the object with the navigation focus is hidden, the form will be in a state where there is no navigation focus. The application is responsible for setting the new focus after it hides the focused object.

  If the application fails to do this and the user presses a directional rocker key when there is no focus on the form, the focus will be moved to the first object in the tab order.

10.9.9.3 Focus rings and redraw problems

- An application may run into redraw problems when it controls the drawing and/or removal of focus rings (when they directly call `HsNavDrawFocusRing()` and/or `HsNavRemoveFocusRing()`).

  Drawing focus rings around an object and properly restoring an object when it loses the focus ring is very tricky. The rings can be drawn over an object's frame, over another object, and over pixels directly drawn to the screen. When the system draws and removes the ring, it takes these possibilities into account and also contends with clipping rectangles and objects that change appearance between receiving and losing the focus ring. The system manages these complications fairly well when it is controlling the drawing and removal of rings.

  For an application to properly handle these complications, developers should have a basic understanding of the ring drawing and removal mechanism. Before a ring is drawn, the bits behind the ring are saved. When a ring is removed, the bits behind the ring are restored, the object is redrawn, and the portion of any object that was behind the ring is redrawn.

  Therefore, it is important that an application always draw an object in its normal state *before* drawing a focus ring around it. If the appearance of an object with the focus ring needs to change (that is, if an object's bounds needs to change) or if what's behind the focus ring needs to change (for example, if the background color of the form needs to change), an application should remove the ring, make the changes, draw the changes, and then draw the ring again.

10.9.9.4 Fields

- To give a field the insertion point, `FldGrabFocus()` should be called. `FldGrabFocus()` will take care of enabling the insertion point and putting the field into interaction mode. Similarly, to take the insertion point away from a field, `FldReleaseFocus()` should be called. `FldReleaseFocus()` will take care of disabling the insertion point and taking the field out of interaction mode.
- Since navigation causes fields to constantly receive and lose the insertion point, it is necessary to always set the proper shift state for a field when it receives the insertion point. The `hsNotifySetInitialFldStateEvent` notification accomplishes this task. An application that has a field that should always have a particular shift state should register for the notification.

  When registering, it should pass the field's form pointer as the user data for the notification. When it receives the notification, it should compare the active form pointer with the form pointer passed as the user data. If the pointers match, it should then call `FrmGetFocus()` to see which object has the focus. If it is the field that it wants to set the shift state for, the application should set the shift state and then mark the notification as handled.

10.10 Handspring® extensions

Available on:

- Centro™ and Treo™ smartphones

The Palm Treo smartphone was originally designed by Handspring, and Handspring created extensions to the Palm OS to support Palm smartphones. You can find the details of the new APIs in the *Palm API Guide, Palm OS Platform* available at **http://pdn.palm.com**.

10.11 Tips and Tutorials

Available on:

- Centro™ and Treo™ smartphones
- LifeDrive™ mobile manager
- Tungsten™ T5, Tungsten™ E2, and Palm® T|X handhelds
- Palm® Z22 organizer

A framework for applications to display formatted help content is included in the Palm OS SDK. This help content serves as an on-device training tool designed to provide customers with a quick introduction to a device, application, or accessory.

10.11.1 Terminology

10.11.1.1 Tips

Tips are a collection of one-screen HTML pages (`Lessons`) that provide succinct usage tips for an application. These Lessons are collected into one Topic.

Tips are accessed from a menu item in the application.

10.11.1.2 Tutorial

A Tutorial is a collection of several Topics.

Tutorials are accessed through an icon in the Applications View instead of directly from an application. A Tutorial consists of a menu, and a collection of ten or fewer Topics tied together in a logical manner. For example, the Tutorial that comes preinstalled on Palm smartphones contains Topics that provide customers with basic instruction on using their Palm smartphone.

On the Treo 600 smartphone, the Applications icon for the Tutorial is labeled "Tutorial." On the Treo 650 smartphone and later, and Centro smartphones, the Applications icon for the Tutorial is labeled "Quick Tour."

10.11.1.3 Topic

A Topic is a collection of Lessons on a common theme.

A Tips file generally has one Topic, while a Tutorial file usually contains multiple topics. Tutorial files generally have a menu page that lets the user select which Topic they want to see. For example, the first Topic in the Tutorial that comes preinstalled on the Palm smartphone, "Getting Started," covers basic familiarity with the Palm smartphone buttons and behavior. When you build an XML file, a Topic is defined by Sequence tags.

10.11.1.4 Lesson

A Lesson, sometimes referred to as a page, is a single screen within a Topic. It consists of text, images, or both. A single idea is usually conveyed by one Lesson, but more complex ideas may require multiple Lessons.

10.11.2 Content

10.11.2.1 Topic titles

Topic titles should fit on a single line—generally, 25 characters or less—and should clearly describe the logical connection between enclosed Lessons. Standard title capitalization rules should be used. When creating a Tips file, the Topic title should be the name of the application the Tips are about.

10.11.2.2 Lesson text

Most Lessons contain text. Standard manual writing guidelines should be used when writing text. Text should be limited to no more than 30 words per Lesson.

10.11.2.3 Lesson images

Whenever possible, images should be used to ground the text with specific examples of the device, application, or accessory in action. See the image creation guidelines for more information.

10.11.3 Tips and Tutorial structure

Before being converted into a Palm OS® PRC file, Tutorials and Tips are collections of HTML files and images organized in a directory. Components include:

- A folder to contain the complete contents of the Tips or Tutorial. This folder is referred to as the *root content folder*.
- A menu page that lists all the available topics. This menu page is contained in an HTML document that is usually named index.html. Tips files that contain only one topic do not need a menu page.
- An HTML page for each Lesson in each Topic:
  - If the HTML page references external content such as images or JavaScript, they must also be present within the root content folder, unless the external content is already in the device's ROM.
  - References to external content must be relative to the root content folder, not to any subfolders within the content. For this reason, it's frequently easiest to put all the content files into one folder and reference files by file name only.
  - HTML pages may also reference the shared content library in the device's ROM. See the shared content description document for more information.
  - A style sheet and JavaScript library that provide the standard Palm Tips and Tutorial styles and formatting are included in the shared content library in the device's ROM. We strongly recommend that all Tips and Tutorial content use these files. See the shared content description document for more information.
- An XML document, usually named tips.xml, that defines the elements and structure of the Tutorial or Tips.

10.11.3.1 Menu document

Tutorials have a menu document written in HTML (usually called index.html) that lets users select a topic to view. Tips files normally contain only one topic, so they do not have menu documents. The menu document contains the following elements.

10.11.3.1.1 Head

```
<html>
<head>
```

```
<meta name="HandheldFriendly" content="True">
<title>Tutorial Main Menu</title>
   <link rel="stylesheet" href="common/inc/style.css" type="text/css"
title="test_style">
   <script language="javascript" src="common/inc/tutorial.js"></script>
</head>
```

Where:

- `<meta name="HandheldFriendly" content="True">` prevents the rendering engine from transforming the content.
- `<title>Tutorial Main Menu</title>` is customizable.

  The content within the title tags should reflect the title of the Tutorial.
- `<link rel...>` and `<script language...>` tell the document which style sheet and JavaScript source file to use.

  These should refer to the shared style sheet and JavaScript files in the shared content library in the device's ROM as shown earlier.

10.11.3.1.2 Body

```
<body onLoad="setPage('menu','');">
```

Where:

- `setPage` is a JavaScript routine that performs cookie actions appropriate to the type of page.
  - The first parameter defines what actions should be taken. The first parameter for a menu page should always be titled "menu."
  - On a menu page, the second parameter should be empty.

10.11.3.1.3 Icon

```
<!-- icon -->
<div id="icon">
<img src="common/img/icon_menu.gif" height="42" width="35" border="0">
```

Where:

- `<div id="icon">` positions the icon.
- `<img src="common/img/icon_menu.gif">` points to the menu icon in the shared content library in the device's ROM. All menus should use this icon.

10.11.3.1.4 Header and Content

```
<!-- header -->
<div id="menu_frame">
<img src="common/img/spacer.gif" height="42" width="35" border="0" class="r_float">
<p class="title">Tutorial Main Menu</p>

<!-- content -->
<img src="common/img/spacer.gif" height="10" width="125" border="0"><br>
<a class="menu" href="get_start_1.html"><img src="common/img/bullet_empty.gif"
name="hsgs" width="16" height="8" border="0">Getting Started</a><br>
<a class="menu" href="keyboard_1.html"><img src="common/img/bullet_empty.gif"
name="hskb" width="16" height="8" border="0">Keyboard Basics</a><br>
<a class="menu" href="top_ten_1.html"><img src="common/img/bullet_empty.gif"
name="hstt" width="16" height="8" border="0">Top 10 Fun Features</a><br>
<a class="menu" href="upgrade_1.html"><img src="common/img/bullet_empty.gif"
name="hsup" width="16" height="8" border="0">Upgrading From a Palm OS Device</a><br>
```

Where:

- `<div id="menu_frame">` defines the position and background image for the menu frame. This should not be changed.
- `<p class="title">Tutorial Main Menu</p>` defines the color and position of the title that will appear on the Tutorial menu page. Replace "Tutorial Main Menu" with the title of your Tutorial. When testing your Tutorial, make sure your title fits in the allotted space.
- Each Topic that is to appear in your menu should be coded as follows:
  `<a class="menu" href="get_start_1.html"><img src="common/img/bullet_empty.gif" id="ft01" width="16" height="8" border="0">Getting Started</a><br>`
  - Replace `get_start_1.html` with the name of the HTML document that defines the first Lesson in the Topic.
  - Replace `hsgs` with a code uniquely identifying the Topic. This code is used on the last Lesson of a Topic, so make a note of it for future reference. This is used to update the bullet to a checkmark once the Topic is completed by the user. See the built-in Quick Tour for an example of this behavior.
  - Replace `Getting Started` with the name of the Topic.

10.11.3.1.5 Footer

```
<!-- footer -->
<span id="menu_footer">Scroll Up or Down and press Center to select a topic</span>
</body>
```

This footer text should appear at the bottom of every menu. Do not change this section.

10.11.3.2 Lesson document

To make it easier to understand the structure, Lesson documents should be given a consistent naming structure. For example, the Lessons in the "Getting Started" Topic are named `get_start_1.html`, `get_start_2.html`, `get_start_3.html` and so on.

10.11.3.2.1 Head

```
<html>
<head>
<meta name="HandheldFriendly" content="True">
   <title>Getting Started</title>
   <link rel="stylesheet" href="common/inc/style.css" type="text/css"
   title="test_style">
   <script language="javascript" src="common/inc/tutorial.js"></script>'
```

Where:

- `<meta name="HandheldFriendly" content="True">` prevents the rendering engine from transforming the content.
- `<title>Getting Started</title>` is customizable.

  The content within the title tags should reflect the title of this Topic. All Lessons in the same Topic should use the same title. Tips files should use the name of the application that the Tips are about as the title of all Lesson documents.
- `<link rel...>` and `<script language...>` tell the document which style sheet and JavaScript source file to use.

  These should refer to the shared style sheet and JavaScript files in the shared content library in the device's ROM as shown in the example. The `<script language...>` tags are necessary only on the last Lesson of a Topic.

10.11.3.2.2 Body

```
<body onLoad="setPage('end','hsgs');">
```

Where:

- The last Lesson of a Tutorial Topic should have a body tag like the one shown in the example.
- `setPage` is a JavaScript routine that performs cookie actions appropriate to the type of page.
  - The first parameter defines what actions should be taken. If this is the last Lesson in the Topic, the first parameter should be set to `end`.
  - The second parameter should be the same as the code used to uniquely identify this Topic on the menu page.
  - This is used to update the bullet to a checkmark once the Topic is completed by the user. See the built-in Quick Tour for an example of this behavior.
- All other Lessons should have a regular `<body>` tag.

10.11.3.2.3 Icon

```
<!-- icon -->
<div id="icon">
<img src="common/img/icon_gs.gif" height="42" width="35" border="0">
<div id="icon"> positions the icon.
```

`<img src="common/img/icon_gs.gif"...>` points to the icon used for this Topic. Replace `common/img/icon_gs.gif` with the path to your icon image relative to the root folder. All Lessons in the same Topic should use the same image. See the attached image guidelines for more information.

10.11.3.2.4 Header and Content

```
<!-- header -->
<img src="common/img/spacer.gif" height="42" width="35" border="0" class="r_float">
<p class="title">Getting Started</p>

<!-- content -->
<img src="common/img/spacer.gif" height="5" width="125" border="0"><br>
<P>
<B>Congratulations on your purchase!</B>
</P>

<P>
The following tips will help you get started.
</P>

<img src="common/img/key_fiveway.gif" width="51" height="34" class="cap_none" border="0"
style="margin: 26 5 0 5">

<P>
Press the Center navigation button to go to the next page.
</P>

<!-- arrows and text -->
<img src="common/img/arrow_down_long.gif" width="8" height="30" border="0"
style="position:absolute; top:83; left:126;">
</div>

</body>
```

Where:

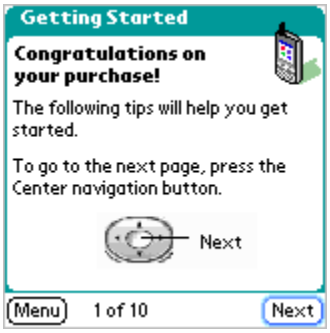
- `<div id="content_frame">` defines the position and area in which the content will appear. This should not be changed.
- `<p class="title">Getting Started</p>` defines the title for the Topic that will appear at the top of the screen. Replace "Getting Started" with your Topic title. The same title should be used for all Lessons in this Topic. Tips files should use the name of the application that the Tips are about as the title of all Lesson documents.
- All content should appear between `<!-- content --> <img src="common/img/spacer.gif" height="5" width="125" border="0"><br>` and `</div>`.

10.11.3.2.5 Lesson Content Formatting

There are several alternate ways of presenting content. In general, any Lesson may consist of the following:

- Text that flows vertically and horizontally in the main content frame
- Images that flow vertically within the main content frame and float right
- Images that flow vertically within the main content frame and float left
- Images that flow vertically within the main content frame and are centered
- Text or images that are removed from the flow of the main content frame and are absolutely positioned

- Text that appears in a graphic box on top of another image, otherwise known as a "callout."



In this example, all text but the word "Next" flows vertically and horizontally in the main content frame. The image of the navigation button flows vertically within the main content frame and floats right. The word "Next" and the arrow pointing to the center button are removed from the flow of the main content frame and are absolutely positioned. The HTML that defines this content is as follows:

```
<!-- content -->
<img src="img/spacer.gif" height="5" width="125" border="0"><br>
<p>
<b>Congratulations on your purchase!</b> The following tips will help you get started.
</p>

img src="img/fiveway.gif" width="51" height="34" class="capp_top" borders="0">

<p>
To go to the next page, press the center navigation button.
</p>

<!-- arrows and text -->
<img src="img/arrow_down_long.gif" width="8" height="30" border="0"
style="position:absolute; top:63; left:125;">
<span style="position:absolute; top:52; left:126;">Next</span>
```

Where:

- The text that will appear within the main flow of the content frame must be set off with paragraph tags (`<p>` and `</p>`).
- Because HTML aligns paragraphs and images at the top of their respective blocks, `<img src="img/fiveway.gif"...>` appears after the first paragraph to align with the top of the second paragraph. The path to the image file, as always, is defined from the root folder.
- `<img...class="cap_top">` defines the margins, which are extended—in this case on top—to make room for a caption. The following are predefined classes and their margins. If you need larger margins, you will need to define them within the `img` tag using the style attribute.
  - `cap_none: top:5; right:5; bottom:0; left:5; float: right`
  - `cap_top: top:30; right:5; bottom:0; left:5; float: right`
  - `cap_left: top:5; right:5; bottom:0; left:30; float: right`

- `<img src="img/arrow_down_long.gif"...>` defines the path from the root folder to the arrow image. You'll find predefined arrows in the attached template documentation; use those as a template if you create your own. Within the `img` tag is the style attribute: `<img...style="position:absolute; top:52; left:126;">`. Use this syntax to place elements exactly, always defining distance from the top-left corner at 0,0 pixels.
- `<span style="position:absolute; top: 52; left: 126;">` is used to absolutely position text. Always define distance from the top-left corner at 0,0 pixels.

This is an example of a callout. The following code generates the callout:

```
<span style="position: absolute; top: 40; left: 60;">
<table border="0" width="80" cellspacing="0" cellpadding="0" align="left" >
    <tr>
        <td><img src="common/img/pull_tl.gif" width="6" height="7" border="0"></td>
        <td><img src="common/img/pull_t.gif" width="68" height="7" border="0" ></td>
        <td><img src="common/img/pull_tr.gif" width="6" height="7" border="0" ></td>
    </tr>
    <tr bgcolor="#CCFFCC">
        <td><img src="common/img/pull_l.gif" width="6" height="55" border="0" ></td>
        <td>When on a call, press Center to Hang Up, or Right to choose another option.</
td>
        <td><img src="common/img/pull_r.gif" width="6" height="55" border="0" ></td>
    </tr>
    <tr>
        <td><img src="common/img/pull_bl.gif" width="6" height="7" border="0" ></td>
        <td><img src="common/img/pull_b.gif" width="68" height="7" border="0" ></td>
        <td><img src="common/img/pull_br.gif" width="6" height="7" border="0" ></td>
    </tr>
</table>
```

Where:

- `<span style="position: absolute; top: 40; left: 60;">` and `</span>` absolutely positions the entire callout on the screen.

  Always measure to the top and left of the block being positioned from the top- left corner of the screen at 0,0 pixels.

- `<table border="0" width="80" cellspacing="0" cellpadding="0" align="left">` starts the table that will draw the callout.

  The width can be adjusted to accommodate content, but should be no wider than 90 pixels.

- In the first row, `<td><img src="common/img/pull_tl.gif" width="6" height="7" border="0"></td>` defines the image of the top-left corner of the callout, `<td><img src="common/img/pull_t.gif" width="68" height="7" border="0"></td>` defines the image for the top bar of the callout, and `<td><img src=" common/img/pull_tr.gif" width="6" height="7" border="0">` defines the image for the top-right corner of the callout in the shared content library in the device's ROM.

  Always use these graphics when creating a callout. Adjust the width of `pull_t.gif` so that the width of all three images equals the width of the table.

- In the second row, `<tr bgcolor="#CCFFCC">` defines the background color of the callout.

  Always use #CCFFCC for the callout background color. `<td><img src=" common/img/pull_l.gif" width="6" height="45" border="0" ></td>` defines the left-side vertical bar of the callout, and `<td><img src=" common/img/pull_r.gif" width="6" height="45" border="0" ></td>` defines the right-side vertical bar of the callout in the shared content library in the device's ROM. The height of these two elements should be the same, and should be adjusted to encompass all of the text in the second cell: `<td>You'll see the number as you dial at the top of the screen.</td>`.

- In the third row, `<td><img src=" common/img/pull_bl.gif" width="6" height="7" border="0"></td>` defines the image of the bottom-left corner of the callout, `<td><img src=" common/img/pull_b.gif" width="68" height="7" border="0"></td>` defines the image for the bottom bar of the callout, and `<td><img src=" common/img/pull_br.gif" width="6" height="7" border="0">` defines the image for the bottom-right corner of the callout in the shared content library in the device's ROM.

  Always use these graphics when creating a callout. Adjust the width of `pull_b.gif` so that the width of all three equals the width of the table.

Some content may be in the form of bulleted lists. Because HTML doesn't support indented lists, use the following code to align numbered lists:

```
<p class="hanging_indent">
1. Press <img src="img/home_key.gif" width="21" height="23" border="0"><br></p>

<img src="img/keypad.gif" width="66" height="56" border="0" class="cap_none">

<p class="handing_indent">
2.Dial numbers directly from the keyboard dialpad.
</p>

<p class="hanging_indent">
3. Press Center to place the call.
```

Where:

- Each list item should be enclosed in a `<p class="hanging_indent">` and `</p>` tag.
- Each step is numbered individually.

10.11.3.3 XML document

An XML document, usually named tips.xml, catalogs the elements and structure of the Tutorial. It tells the PRC file creation utilities which content should be added into the Tips or Tutorial PRC file. It also defines the ordering of the lessons and the structure of the topics. It is an XML-formatted file that is processed by createTipsRsc.exe to create a standard Palm OS XRD resource file. For information about this process or clarifications about tips.xml syntax, examine the source code for createTipsRsc.exe.

All XML tags and tag attributes are case sensitive. Filenames in the tips.xml file as well as within the HTML content are case sensitive. To make things simpler and easier, all file names within the tips.xml file and within all tags in the content should be in lowercase letters.

The header of the XML document is formatted as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<Tutorial Name="Device_Tutorial " Base="Tutorial_Tips" Description="Device
Tutorial">
```

Where:

- `<?xml version="1.0" encoding="UTF-8"?>` defines the document type. This should not be changed.
- `<Tutorial Name="Device_Tutorial" Base="Tutorial_Tips" Description="Device Tutorial">` needs to be edited.

  Replace `Device_Tutorial` in both the Name and Base attributes with the name of the PRC database that will be created by the Tips and Tutorial creation utilities. Replace `Device Tutorial` with the human-readable name of the Tutorial file you want to create when the XML document is compiled.

```
<Page Base="1" Default="1" File="index.html"/>
```

- Tutorials may contain pages that are not part of a Topic, such as the main menu page. The syntax shown demonstrates how this is accomplished:
  - `Base="1"` tells the creation utilities that this is a page that is shown directly and may load other content, as opposed to included pages such as images or style sheets.
  - `Default="1"` tells the creation utilities that this is the page that should be displayed when this Tutorial is first opened. Only one page in a tips.xml file should have the Default attribute set.

- Because the content in a Tips file is usually all in one Topic, Tips files usually do not have this form of the Page tag in their tips.xml files.
  - In a Tips file, the first page of the Topic should be flagged `Default="1"`

```
<Sequence Menu="index.html">
    <Page File="get_start_1.html"/>
    <Page File="get_start_2.html"/>
    <Page File="get_start_3.html"/>
    <Page File="get_start_5.html"/>
    <Page File="get_start_6.html"/>
    <Page File="get_start_7.html"/>
    <Page File="get_start_8.html"/>
    <Page File="get_start_9.html"/>
    <Page File="get_start_10.html"/>
    <Page File="get_start_11.html"/>
    <Page File="get_start_12.html"/>
    <Page File="get_start_13.html"/>
</Sequence>
```

- The sequence tag defines one topic. Because a Tips file usually contains one Topic, it will usually contain one pair of Sequence tags. Tutorials usually contain multiple Topics, so they will usually contain multiple pairs of Sequence tags.
- For a Tutorial, the sequence tag looks like this: `<Sequence Menu="index.html">`. The Menu attribute tells the system where to navigate when the Menu button is tapped. If you've named the menu page something other than index.html, you will need to change it here.
- For a Tips file, the sequence tag looks like this: `<Sequence Done="1">`. The Done attribute tells the system to show a Done button instead of a Menu button. The Done button, when clicked, exits Tips and returns the user to the calling application.
- `<Page File="get_start_1.html"/>` tells the creation utilities what HTML pages contain the Lessons for this Topic. Order is important in this section; the Lessons will be shown in the same order in which the Page tags appear within the Sequence tag.

```
<Page File="inc\style.css"/>
<Page File="inc\tutorial.js"/>
<Page File="inc\active_call.gif"/>
```

- All other files used by the Tutorial or Tips file—other than shared content already in the device's ROM—need to be defined here using the relative path from the root folder. For instance, all images used by the Lesson pages must be included here. Order is not important in this section.

10.11.4 Converting Tips and Tutorial content in a PRC file

10.11.4.1 What you need

Use the provided utilities and guidelines to create Tips and Tutorials. The utilities are provided in the Palm OS® SDK under TipsTutorialsUtil.zip. The contents of the TipsTutorialsUtil.zip file are as follows:

- Content developed using the guidelines in this chapter.
- The tips.xml file that defines the content in the shared content library in the device's ROM. This file should be copied into the root content folder.
- The createTipsRsc.exe program. The Perl source code for this program is also included for your reference.
- The Palm-BinTool.exe program.
- The PalmRC.exe program.
- A creator ID for the resulting PRC file. We highly recommend that the Tips and Tutorial PRC file have a different creator ID from that of the main application. This is the same type of creator ID as a standard Palm OS application and can be obtained from the ACCESS (formerly PalmSource) Developer Network website at **http://www.access-company.com/developers/**.

10.11.4.2 Converting content into an XRD resource file

To convert the content into a Palm XRD file, open a command prompt and change directory (cd) to the directory with the tips.xml file. (The tips.xml file is described in the guidelines.) Then run `createTipsRsc`. The syntax for the `createTipsRsc` command is as follows:

```
createTipsRsc -i tips.xml -o tips.xrd -c CREATORID -s
shared_content.xml -p c:\bin\palm-bintool.exe -f
```

Where:

- `-i` specifies the input XML file.
- `-o` specifies the output XRD file.
- `-c` specifies the creator ID for the final PRC file. `CREATORID` is a filler. You should obtain an authentic one from ACCESS.
- `-s` specifies the XML file that describes the shared content library in the devices ROM.
- `-p` specifies the full path to the palm-bintool.exe file.
- `-f` is an optional flag that sets the backup flag on the final PRC file. This tells the HotSync operation to back up this file and to restore it automatically after a hard reset.
- `-l` is an optional flag that sets the current locale. For instance, `esES`.

10.11.4.3 Converting a Palm XRD resource file into a PRC

To convert the XRD file into a PRC, run PalmRC with the following syntax:

```
PalmRC.exe tips.xrd -p ARM -overlayFilter BASE -target 4.0 -o tips.prc
```

Where:

- `tips.xrd` is the name of the input XRD file.
- `tips.prc` is the name of the output PRC file.

Do not modify any of the other command-line parameters. The resulting PRC file can then be placed onto the device through a normal HotSync operation.

10.11.5 Displaying Tips and Tutorial content

Use the methods described in this section to display Tips and Tutorial content.

10.11.5.1 Displaying application tips

To display tips within an application, launch the Blazer® web browser in Tutorial mode. The command block should be a pointer to a null-terminated string that is the name of the PRC database containing the application tips to be displayed. After the browser exits, the calling application is relaunched automatically.

```
#define myappTipsDbName                 "MyApp_Tips"
#define myappTipsDbNameLength           10

void DisplayTips() {
    Char* startPage;
    startPage = MemPtrNew( myappTipsDbNameLength + 1 );
    MemPtrSetOwner(startPage, 0);
    StrCopy( startPage, myappTipsDbName );
    AppLaunchWithCommand( hsFileCBlazer3,
sysAppLaunchWebBrowserTutorialMode, startPage );
}
```

10.11.5.2 Displaying a Tutorial

There is little programmatic difference between displaying Tips and displaying a Tutorial. Tips are displayed by launching the Blazer® web browser in Tutorial mode from within an application, whereas a Tutorial is displayed by launching the Blazer web browser in Tutorial mode from a stub application that appears in the Applications View. This stub application has two functions:

- Provide a user-visible entry point into the Tutorial by displaying an icon in the Applications View.
- Launch the Blazer web browser in Tutorial mode with the command block pointing to a null-terminated string that is the name of the PRC database that contains the Tutorial to be displayed.

The following is an example of a stub application:

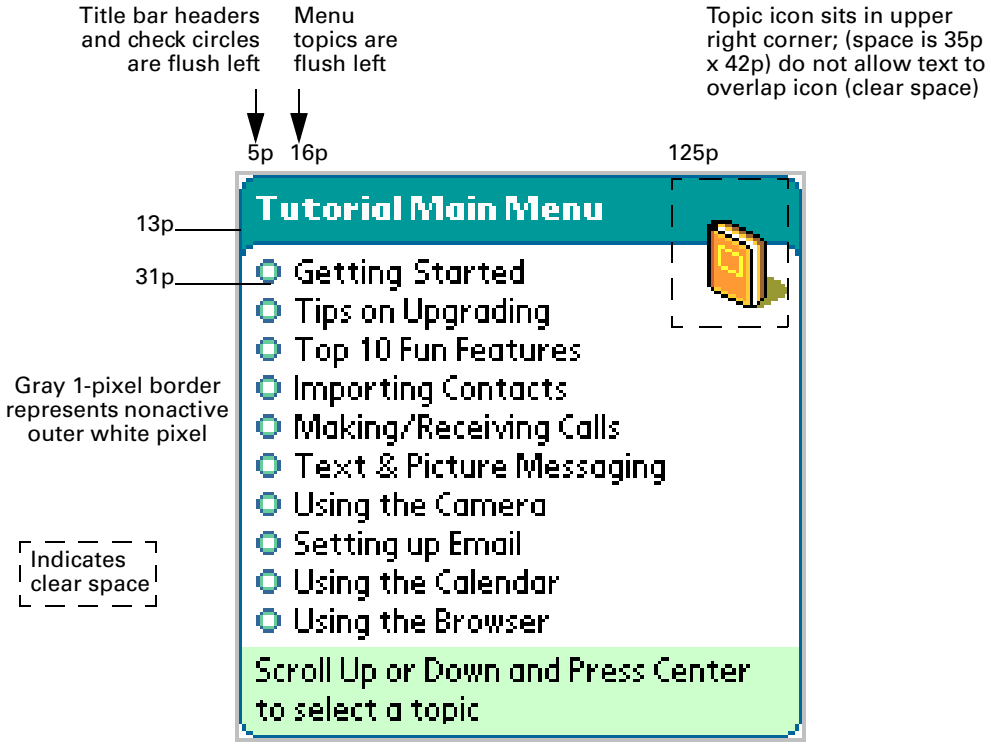
```
#define Tutorial_Start_Page "Tutorial_Tips"
#define Tutorial_Start_Page_Length 13

UInt32  PilotMain(UInt16 cmd, MemPtr cmdPBP, UInt16 launchFlags)
{
  if (cmd == sysAppLaunchCmdNormalLaunch) {
    Char* startPage;
    startPage = MemPtrNew(Tutorial_Start_Page_Length + 1);
    StrCopy(startPage, Tutorial_Start_Page);
    MemPtrSetOwner(startPage, 0);

    AppLaunchWithCommand(hsFileCBlazer3,
sysAppLaunchWebBrowserTutorialMode, startPage);
   }

  return 0;
}
```

10.11.6 Graphic element design guidelines

This section details the graphic element design guidelines.

10.11.6.1 Tutorial main menu

The Tutorial main menu is as follows:



Icons for upper right corners: Each icon used per chapter should be drawn in Photoshop at 30/60 degree increments using 2x1 stepping. Each icon should face left and have a cast shadow to the right.

This page intentionally blank

10.11.6.2 Tutorial content pages

Content Page: large graphic & pull quotes
Full screens should be approximately 70% and placed on the left side of the screen. Pull quotes should be placed to the right as shown, with arrow pointing to area that text is referencing. Use arrows (and circles) that exist in the template or request a set from Palm.

Same clear space
(32p x 42p) applies
on content pages

Content Page: graphic (tall) & text
All graphics, except when they completely fill the screen, should be placed in the lower right corner. Use arrows with or without text to point out information and circles to call out specific details. Small graphics should be enlarged so that they are clear.

Same clear space
(35p x 42p) applies
on content pages

Content Page: graphic (wide) & text
All graphics, except when they completely fill the screen, should be placed in the lower right corner. Crop the images so that they are recognizable.

Content Page: Text only
Clear space for the icon still applies. For "scroll up or down" use the same format from the Main Menu.

All button or key graphics should appear as shown in this screen.

Do not crop buttons like this:

Content Page: graphic (centered) & text
Certain graphics will look better when centered. This is the exception, not the rule. Note that elements such as buttons or keys from the device should be cropped as shown above.

Graphics: All icons used in the upper right corner should be drawn in Photoshop. All other placed images, such as screen shots or device images, must be taken from the original source files and reduced to fit into the space allowed per page. All images should be saved out of Photoshop using "save for web" at actual size in either JPEG or GIF format, whichever is smaller. For JPEG, quality setting should be 20 or 30, depending on the image, with 0 blur. For GIF, preferences should be set to GIF 32 No Dither.

All device images are the property of Palm, and may be used with permission.

10.11.6.3 Images that are in the shared content library in the device's ROM

The images available directly from the device's ROM for your own Tips and Tutorials can be found in the utility section of the Palm OS SDK available at **pluggedin.palm.com**.

10.12 Full-Screen Writing API

Available on:

- LifeDrive™ mobile manager
- Tungsten™ T5, Tungsten™ E2, and Palm® T|X handhelds
- Zire™ 31, Zire™ 72, and Palm® Z22 organizers

This section contains reference information for the full-screen writing feature that is provided programmatically through the GoLCD (Graffiti® 2 writing on LCD) Manager API. You can use the functions in this API to enable or disable full-screen writing, enable and disable the Graffiti 2 shift indicator (GSI) as a control for full-screen writing, enable and disable Graffiti 2 inking, and even change the colors of Graffiti 2 inking and the GSI.

The GoLCD API is declared in the header file `PalmGoLCD.h`.

Full-screen writing allows users to enter Graffiti 2 characters in the application area of a handheld's display as well as in the Graffiti 2 writing area.

By setting Graffiti 2 Preferences, users can enable full-screen writing and choose whether to show, or *ink*, the Graffiti 2 strokes in the application area.

In each application the availability of full-screen writing is indicated by a shaded, rectangular Graffiti 2 shift indicator (GSI) in the lower-right corner of the display. The user can tap the GSI to turn full-screen writing off and on. The GSI appears as an outline when full-screen writing is off, and as a solid rectangle when it is on. In addition, the shift indicator, punctuation-mode indicator, and shift-lock indicator appear superimposed on the shaded rectangle when the user draws the appropriate Graffiti 2 strokes to activate those modes.

Graffiti 2 strokes in the application area are distinguished from taps on application controls, depending on the duration and direction of tap-and-hold events. GoLCD interprets pen events in the writing bounds of the application area, which you can set using `GoLCDSetBounds`. If the pen is held down for a certain length of time and travels significantly across the screen in the writing bounds area, GoLCD enters `goLcdGraffitiMode` and interprets all pen events as Graffiti 2 strokes. GoLCD exits `goLcdGraffitiMode` and stops interpreting pen events as Graffiti 2 strokes when the pen is lifted from the screen for a certain amount of time. For more information, see the *Palm API Guide, Palm OS Platform*.

This page intentionally blank

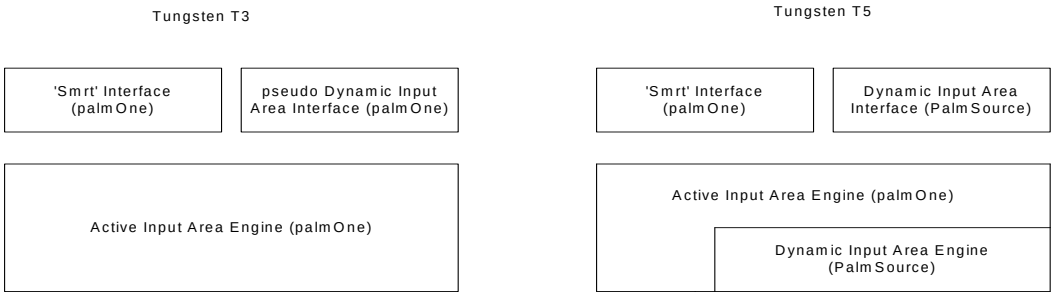
10.13 Dynamic Input Area (DIA)

Available on:

- Tungsten™ T3 and Tungsten™ T5

Dynamic Input Area (DIA) aware applications that worked on Tungsten T3 handhelds may behave differently on Tungsten T5 or later handhelds. This section explains the DIA-related differences between Tungsten T5 and Tungsten T3 handhelds.

The ACCESS® (formerly PalmSource) DIA and the Palm Active Input Area (AIA) were initiated independently at around the same time to achieve similar goals, providing a status bar and a dynamic replacement for the silk-screened Graffiti input area. The AIA ran atop Palm OS 5.2, while the DIA originally targeted Palm OS 6. Early in the course of their development, many of the public APIs were unified with the goal of easing the transition to Palm OS 6.



Overview of the Input Area System on Tungsten T3 and T5

As shown in the figure, when the initial delivery of the ACCESS DIA was accelerated to appear with Palm OS 5.3, Palm created a compatibility layer to mimic the behavior of the DIA APIs from Palm OS software 5.3, while running atop the Palm AIA on Palm OS software 5.2. The goal was, and continues to be, to provide support for the ACCESS APIs, regardless of the implementation underneath.

The most significant difference between the AIA and DIA models is in the scope of "tall screen awareness"—the AIA model is application-centric, using a "Smrt" resource to flag an entire application as "tall-screen–aware," and the DIA model is form-centric, using an API call during the setup of a form to flag that form as "tall-screen–aware." Newer applications from Palm are written to use the DIA model, but legacy code still exists, using the AIA model.

Tungsten T5 handhelds using Palm OS 5.4 software include DIA support in the OS, so the problem shifted from providing forward compatibility—handhelds adding a pseudo-DIA interface to Palm OS 5.2 software—to providing backward compatibility for our legacy code. As you might imagine, this is not a simple feat; it's somewhat like having a car with two steering wheels. There's only one input area, but two concurrent models for controlling it.

This page intentionally blank

CHAPTER 11

11. Applications

This chapter details the features and APIs available in some of the Palm® applications.

11.1 Web Browser API

This section describes the various technical features of the Palm web browser system that allow website designers and programmers to deliver a better web experience.

Available on:

- Centro™ and Treo™ smartphones
- Tungsten™ T5, Tungsten™ E2, and Palm® T|X handhelds
- LifeDrive™ mobile manager

The Palm Blazer® web browser is an application designed for devices that use Palm OS®. It can access multiple Internet content formats, including HTML, xHTML, cHTML, and WML. With the web browser, a user can download web pages from the Internet and view them on the compact device screen. The key features of the web browser are as follows:

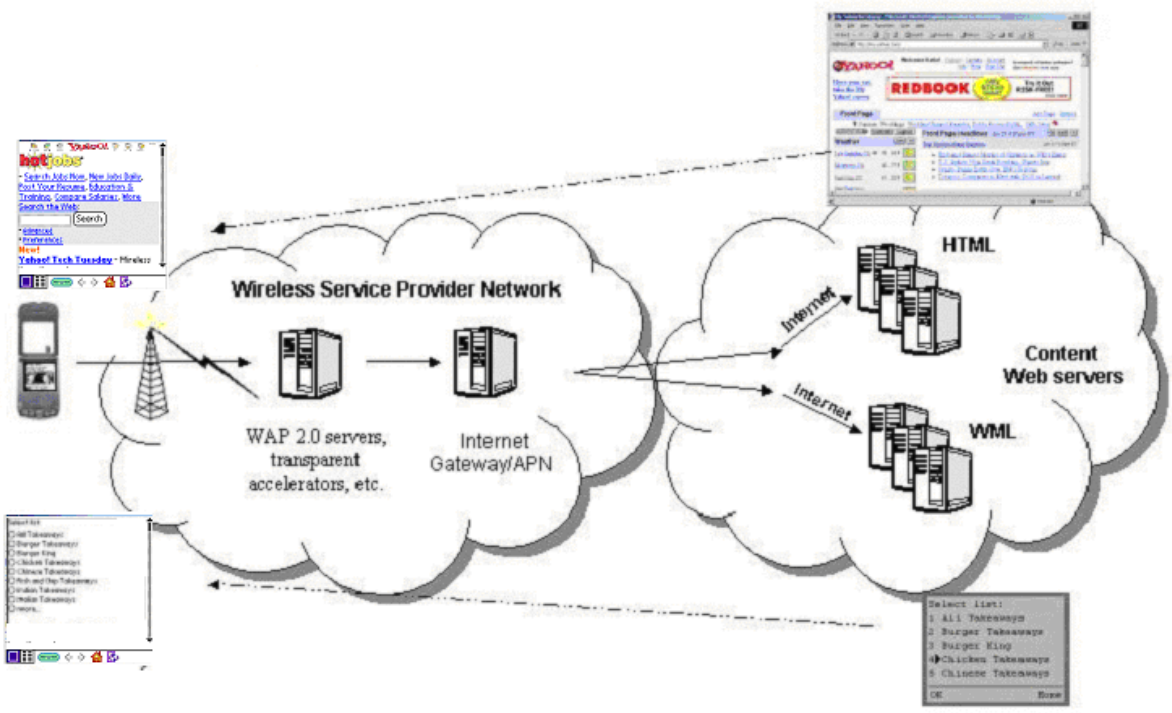
- An intuitive graphical interface
- Support for multiple markup languages
- Support for secure websites
- Choice of display modes:
  - Optimized mode for optimization of content for Palm OS devices
  - Wide Page mode for the display of content similar to that of a desktop browser
  - Normal or fast mode for page rendering
- 5-way navigator support
- A variety of UI features including History, Saved Pages, and Beaming Bookmarks
- Ability to download applications, ring tones, images, and more
- Support for streaming content in Blazer 4.5

11.1.1 How the web browser works

Versions of the web browser before Blazer version 3.0 were built as a proxy solution. The proxy server adapted pages from the remote web servers and streamed content to the client. The client application received the content from the proxy server and displayed the web page on the device's screen. Since Blazer 3.0 however, the Palm web browser is a proxyless, client-only browser. (It is capable of using a standard HTTP proxy in the same way a desktop browser would use a proxy.)

There are several reasons why this architecture was chosen. For one, having a client-only solution allows devices to access more wireless service provider services, such as downloads and mobile content. Also, device processing power has reached the point where the content optimizations can be accomplished efficiently enough on the client.

The figure below shows how the current proxy-less web browser accesses web pages.



The Blazer web browser uses the Palm OS Network Library for Bluetooth® wireless technology, WiFi, and the Palm NetMaster Library for GSM/CDMA to make the connection to the wireless service provider infrastructure.

11.1.2 Web browser feature overview

This section covers the features of the web browser. This information is useful for understanding the software requirements, technical features, and user interface elements.

11.1.2.1 Protocol stack support

The web browser uses the internet standard HTTP stack for communications. While the browser does support WML, there is no support for the WAP 1.x protocol stack.

11.1.2.2 Overview of key web browser features

The web browser incorporates a number of key technical features described in this section.

11.1.2.2.1 Intuitive graphical interface

The web browser's user interface takes advantage of the device's screen size. The web browser provides one-touch access to bookmarks, navigation, the home page, and new web pages. It also gives users a familiar look and feel comparable to that of a desktop browser.

11.1.2.2.2 Support for multiple markup languages

The web browser supports several markup languages, as well as cascading style sheets and JavaScript. This allows users to access a wide variety of web and wireless content from a single browser and gives content providers flexibility in determining what type of markup language to use. The Palm web browser supports the following markup languages:

Markup Language
HTML 4.01
XHTML 1.1
XHTML Mobile Profile
cHTML (iHTML)
WML 1.3
DHTML
DOM

For special tags that the browser supports, see **Section 14.1.9 on page 277**.

11.1.2.2.3 Support for secure websites

The web browser supports the following security features:

- End-to-end security using SSL 3.0
- 128-bit encryption using the RC-4 algorithm
- RSA-based key exchange
- RSA-based digital signature verification for verifying the authenticity of signed certificates, signed code, and so forth.
- MD5 and SHA-1 secure hash algorithms
- SSL 3.0 with server-side authentication only
- Support for X.509 certificates
- Indication of a secure connection by a lock icon in the toolbar or URL bar.

11.1.2.2.4 Optimization of content for device screens

The web browser optimizes the content to take advantage of device characteristics such as screen size and navigation.

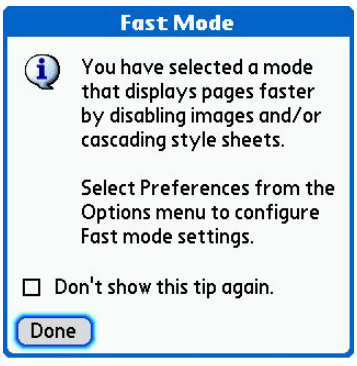
11.1.2.2.5 Normal or fast mode for rendering pages

Blazer v. 4.3 and later allows a user to choose how they want their pages to be rendered: normal or fast. Normal mode renders a page the same way that previous versions rendered pages. Fast mode is by default rendering with no cascading style sheets and no images. While in fast mode, users can still download individual images by tapping an image and holding it for longer than one second.

Users can set preferences for what fast rendering does on their device. In addition, there is a normal/fast mode quick switch (the icon looks like a lightning bolt) available on the Blazer main form page, where the optimized/wide page quick switch used to be. The optimized/wide page mode setting is now only in the Blazer “options” menu.

Select mode

Set preferences



11.1.3 Download manager

The web browser incorporates download management technology that allows users to download files over a wireless network. There are a variety of applications for this technology:

- Posting applications, ring tones, images, and so forth for your users to download
- Incorporating mechanisms in your application to either auto-check for updates and download them, or allow the user to manually download updates

11.1.3.1 Content support in Blazer® 3.0 and Blazer® 4.0

When a user attempts to download a file, the download manager first checks with the Exchange Manager to see if a content handler has registered for that type of file. If a content handler is registered, then the download proceeds. If no content handler registered is, different versions of Blazer behave differently.

- Blazer 3.0 notifies the user that they must first install a content handler for that file type before they will be able to download the file.
- Blazer 4.0 allows all non-DRM-protected content to be saved to an expansion card, if one is present, even if there is no Exchange Manager handler for that content type.
- For Blazer 4.5, see **Section 11.1.3.2 on page 204**.

For example, if the user tries to download a MIDI ring tone, because there is an application in the device's ROM that handles MIDI, the download proceeds. However, if the user attempts to download an Adobe® Acrobat® file, unless they have an application installed on their device that handles Acrobat files and that has registered with the Exchange Manager as a handler for Acrobat files, the user is not allowed to proceed with the download unless the user is running Blazer 4.0 and has an expansion card inserted in their device. If the user is running Blazer 4.0 and has an expansion card inserted in their device, the user could download the Acrobat file directly to the expansion card. For devices with an internal drive (for example, Tungsten T5 handhelds and LifeDrive mobile managers), in the absence of an expansion card, the file is downloaded directly to the internal volume.

In order to avoid memory problems, Blazer 4.0 limits the size of downloaded files to 2MB. If a user tries to download a bigger file, an error message is displayed. This restriction has been removed for devices with Blazer 4.1. and later (currently only LifeDrive mobile managers). Without this restriction, if the file is less than 2MB and the device has sufficient memory in the DBCache, the file is downloaded through the Exchange Manager. If the file is larger than 2MB or there isn't enough space in the DBCache, users will see a dialog informing them that the file will be downloaded to the internal drive via VFS Manager.

For your convenience, Blazer 4.0 lists every type currently registered in the Exchange Manager in the HTTP Accept header.

11.1.3.2 Content Support in Blazer® 4.5

11.1.3.2.1 Caching rules

Blazer 4.5 includes some rules for caching. Specifically:

1. Any web page that exists in Blazer history will be read from cache at first, and Blazer will not access the network unless it does not have a cache entry for that page. Only if there is no cache entry for a page will Blazer access the network.
2. The no-cache message is ignored. No-store and SSL caching rules are respected.

11.1.3.2.2 Streaming content

Blazer 4.5 has added support for streaming content. The web browser can stream content in two ways:

1. It can search the web page for an embedded object. If it finds an `<EMBED>` or `<OBJECT>` tag on the page with a MIME-type that an application is registered for, then Blazer will display a Play button. Links are then passed to the streaming application via the exchange manager when a user clicks those links.
2. Streaming content may also be presented as a direct link. If an application registered for streaming that type of content, the user will be presented with a dialog box that allows the user to choose whether to play the content, save it to the device, or save it to the card.

11.1.3.2.3 Link protocol types

You can use the RTSP or HTTP URL scheme in an application to link to streaming content:

- **RTSP -** Using RTSP links, an application will redirect the user immediately to the registered streaming content. The user will not be presented with a dialog or a play button.
- **HTTP -** Using HTTP links, if an application is registered for streaming the type of content to be downloaded, an application will present a dialog box that allows the user to play the content, save it to the device, or save it to the card. If no streaming application is registered to handle the MIME-type or file extension, an application will present only the save option.

For more information on audio and video streaming in Blazer® 4.5, see **Section 14.1.4.4 on page 264**

11.1.3.2.4 Supported formats for streaming content

The following table lists formats supported by Blazer 4.5 for streaming, but please note that all codecs will not be available for all devices. For a list of codecs available by device, see **Section 5.1.4 on page 56**.

Scheme	Formats
URL scheme	http://
	mms://
	rtsp://
Media File Formats	.3GP
	.3G2
	.MP4, .M4A
	.MP3
	.ASF, .WMA, .WMV
Video codecs	MPEG4
	H263
	WMV
	AVC H.264
Audio codecs	AMR
	QCELP
	MP3
	WMA
"Wrapper" file formats	SDP
	ASX
	WAX, WVX
	M3U

NOTE: For Windows Media video codecs to work properly on the Treo 700p and Treo 755p smartphones, the compression profile should be set to "Baseline" or "Simple". This must be changed from the default "Main" profile using a compression tool.

11.1.3.3 Download restrictions

The web browser includes download restrictions listed in the following sections.

11.1.3.3.1 JPEG images

Normal behavior for a browser is to render JPEG images within the browser, as opposed to downloading them and saving them on the device. Therefore, an image must be specially flagged if it is to be downloaded rather than rendered in the browser. There are two ways to flag an image for download:

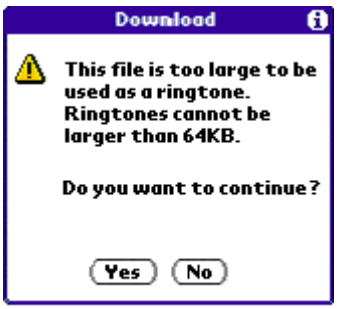
- Use a descriptor file to precede the image. The download manager treats any images called by descriptor files as a download, and does not render them in the browser.
- Flag the image with a special MIME type. If the web browser sees an image with the following MIME type, it will know the image is for download, not rendering: **x-handspring-image/jpeg**.

The only images officially supported for download are JPEG images. All other images are rendered in the browser.

In Blazer 4.0 and later, the user can also tap-and-hold most images to bring up a Save As dialog box that allows the user to save the image to the Exchange Manager or to an expansion card (if present).

11.1.3.3.2 Ring tones

The Palm Ring Tone manager can only play ring tones that are under 64KB. If the user attempts to download a ring tone that is larger than 64KB, they are presented with the following message:



11.1.3.3.3 Multiple file downloads

There are several schemes for supporting multiple file downloads. The most common is to combine the files in a ZIP file and then extract that file onto the device. In order for this to work, the user must first have an application on the device that can extract ZIP files and that registers for ZIP files with the Exchange Manager.

Alternatively, the Nutshell installer from Ecamm Network (**www.ecamm.com**) works well. You can use the installer to package multiple PRC and PDB files into a single PRC file. When the user runs the resulting installer PRC, the installer unpacks the PRCs and PDBs and installs them on the device.

11.1.3.3.4 Digital rights management

The download manager provides some digital rights management functionality. Specifically, wireless service providers can specify domains from which downloads will be forward locked, which means that the file cannot be beamed, sent, and so forth. Forward lock applies to several multimedia types. It does not apply to PRC applications. PRC applications have built-in digital rights management, and those rights management schemes (for example, tying usage to HotSync® ID) should be employed at the content developer's discretion.

11.1.3.4 Streaming code example

The following example code will allow you to determine if a streaming application is available, and if so, to stream to that application. To use this code, you must know how to determine the MIME-type or file extension of the content to be streamed.

```
/**
*Check to see if a file can be streamed.
*
*Inputs:
*   Char* mimeType: mimetype of content or NULL
*   Char* extension: extension of content or NULL
*   UInt32* creatorId: empty
*
*Outputs:
*   Boolean: true if streaming app. found, else false.
*   UInt32 *creatorId: creator id of the streaming application if found
*/
Boolean HasStreaming(Char *mimeType, Char *extension, UInt32 *creatorID, Boolean
*nullMime)
{
   Boolean isStreamingApp = false;
   if(nullMime)
      *nullMime = false;

   if (mimeType) //valid mimetype
   {
      if ((errNone == ExgGetDefaultApplication(creatorID, exgRegStreamingTypeID,
            mimeType)) ||
         (errNone == ExgGetDefaultApplication(creatorID, exgRegStreamingType1ID,
          mimeType)) ||
         (errNone == ExgGetDefaultApplication(creatorID, exgRegStreamingTypeSkipUIID,
          mimeType)))
      {
         isStreamingApp = true;
      }
      else if (extension)
      {
         if ((errNone == ExgGetDefaultApplication(creatorID,
              exgRegStreamingExtensionID, extension)) ||
            (errNone == ExgGetDefaultApplication(creatorID,
             exgRegStreamingExtension1ID, extension)) ||
            (errNone == ExgGetDefaultApplication(creatorID,
             exgRegStreamingExtensionSkipUIID, extension)))
         {
            isStreamingApp = true;

            if(nullMime)
               *nullMime = true;
         }
      }
   }
   else if (extension) //valid extension
   {
      if ((errNone == ExgGetDefaultApplication(creatorID, exgRegStreamingExtensionID,
           extension)) ||
         (errNone == ExgGetDefaultApplication(creatorID, exgRegStreamingExtension1ID,
          extension)) ||
```

```
          (errNone == ExgGetDefaultApplication(creatorID,
           exgRegStreamingExtensionSkipUIID, extension)))
       {
          isStreamingApp = true;
       }
   }

   return isStreamingApp;
}

/**
*Pass url to streaming application.
*
*Inputs:
*   Char* url: url to be passed to streaming application
*   Uint32 creatorId: creator id received from query to exchange manager in case
*                   streaming content supported
*
*Outputs:
*   Err: any error received from exchange manager transactions
*/
Err StreamToExgMgr(Char *url, UInt32 creatorId, Char *documentExtensionP, Char
*mimeType)
{

   UInt32 bytes;
   UInt32 bytesSent;
   ExgSocketType exgSocket;
   Err error;
   Char* tempUrlPtr;
   EventType event;

   Int extLen = 0;
   if (!url)
      return memErrInvalidParam;

   if(documentExtensionP)
   {
      extLen = StrLen(documentExtensionP);
   }

   tempUrlPtr = url;
   bytes = StrLen(url) + sizeOf7BitChar('\0');


   //create exchange manager socket
   MemSet(&exgSocket, sizeof(exgSocket), 0);
   exgSocket.length = bytes;
   exgSocket.target = creatorId;//application to receive message
   exgSocket.goToCreator = creatorId;//application to launch after mesg received
   exgSocket.localMode = true;//limit to local application
   exgSocket.noStatus = true;//don't display progress dialog
   if (mimeType != "text/plain")
      exgSocket.type = StrDup(mimeType);//set the type of the file who's url is being
      passed

   exgSocket.name = MemPtrNew(StrLen("filename.") + extLen + sizeOf7BitChar('\0'));
```

```
    StrCopy(exgSocket.name, "filename.");

    if(extLen > 0)
    {
        StrCat(exgSocket.name, documentExtensionP);
    }


    if (NULL == exgSocket.name)
    {
        if (NULL == exgSocket.type)//this should not happen
        {
            ErrNonFatalDisplay("Trying to pass a file with invalid mimetype and
            extension");
        }

        exgSocket.name = StrDup("InvalidExtension");

        if (NULL == exgSocket.name)//out of memory?
            return 1;
    }

    //send data to target application
    error = ExgPut(&exgSocket);

    if(error == exgErrDeviceFull || error == exgMemError)
    {
        //MJP: attempt flushing the dbcache and try again
        PrvForceDBCacheFlush();
        error = ExgPut(&exgSocket);
    }
    if (errNone != error)//check for error
    {
        if (exgSocket.type)
            MemPtrFree(exgSocket.type);
        if (exgSocket.name)
            MemPtrFree(exgSocket.name);
        return error;
    }

    //loop send until entire url is sent
    while ((errNone == error) && (bytes > 0))
    {
        bytesSent = ExgSend(&exgSocket, tempUrlPtr, bytes, &error);
        bytes -= bytesSent;
        tempUrlPtr = tempUrlPtr + bytesSent;
    }

    error = ExgDisconnect(&exgSocket, error);

    if (exgSocket.type)
        MemPtrFree(exgSocket.type);
    if (exgSocket.name)
        MemPtrFree(exgSocket.name);

    return error;
}
```

11.1.4 Launching the web browser on Palm™ smartphones

If you want your application to take the user directly to a web page, the web browser has the ability to launch a specific URL. The following code sample shows how to launch the web browser and go to a specific web page.

```
static void
LaunchBlazerWithURL(Char* urlP)
{
   Err err = 0;
   UInt16 cardNo;
   LocalID dbID;
   DmSearchStateType searchState;
   Char* url;

   // first check if web browser is installed
   err = DmGetNextDatabaseByTypeCreator(true, &searchState, sysFileTApplication,
      hsFileCBlazer3, true, &cardNo, &dbID);
   if (err)
      {
         // Display appropriate error dialog...
         return;
      }
   // ok, now let's call the web browser with the URL. Must first copy the URL,
   // because it will be disposed of by the system after the browser exits
   url = MemPtrNew(StrLen(urlP)+1);
   if (!url)
      {
         return;
      }
   StrCopy(url, urlP);
   // set the memory owner to zero, so it is not deleted
   // by the system when we switch apps
   MemPtrSetOwner(url, 0);
SysUIAppSwitch(cardNo, dbID, sysAppLaunchCmdGoToURL, url);
```

11.1.5 Launching the web browser in minimal mode

The web browser includes a minimal UI mode. This mode lets you display a web page with the simplest possible UI. The following code sample shows how to launch the web browser in minimal mode, and go to a specific web page.

```
static Boolean doWebBrowserMinimalMode( Char * url )
{
    webBrowserMinimalModeLaunchInfo* minimalModeInfo;
    UInt16 currentFormId;
    Char* tempStr;

    minimalModeInfo = MemPtrNew ( sizeof(webBrowserMinimalModeLaunchInfo) );

    // Since we're calling into ARM, byte-swap the pointers
    minimalModeInfo->launchUrl = (Char*)ByteSwap32(url);

    tempStr = MemPtrNew( 5 );
    StrNCopy(tempStr, "Done", 5);

    // Since we're calling into ARM, byte-swap the pointers
    minimalModeInfo->doneButtonLabel = (Char*)ByteSwap32(tempStr);

    MemPtrSetOwner (tempStr , 0);
    MemPtrSetOwner (minimalModeInfo, 0);
    MemPtrSetOwner (url, 0);

    AppLaunchWithCommand( hsFileCBlazer3, sysAppLaunchWebBrowserMinimalMode,
minimalModeInfo );

    return true;
}
```

11.2 VersaMail® application API

Available on:

- Centro™ and Treo™ smartphones
- Tungsten™ T5, Tungsten™ E2, and Palm® T|X handhelds
- LifeDrive™ mobile manager

This section provides reference information for the VersaMail application Device APIs. You can use these APIs to create attachment plug-ins for VersaMail attachments, add email to VersaMail folders programmatically, create background network connections, and so forth.

11.2.1 Before using the VersaMail® Device APIs

The VersaMail APIs assume a working knowledge of the following:

- VersaMail application itself
- Palm OS programming
- Palm OS 68K runtime environment

11.2.2 Overview of the VersaMail® Device APIs

The VersaMail Device APIs consist of five main components that are documented in the remaining portions of this section:

- VersaMail Account Configuration

  This component allows you to distribute a PDB file to multiple users to set their initial VersaMail configuration automatically.

- Adding Outgoing Emails to VersaMail Folders

  This component allows you to add email messages to VersaMail folders programmatically. It can be used to distribute Welcome email, corporate information email, application information email, and so forth.

- VersaMail Font library

  This component allows you to set the fonts displayed in the VersaMail Font Picker dialog box, as well as get and set information about fonts programmatically.

- VersaMail Attachment Plug-ins API

  This component allows you to create a plug-in for a type of attachment (for example, BMP attachments) that allows users to send and view VersaMail email attachments.

11.2.2.1 VersaMail® Account Configuration

In an enterprise environment, it is often useful to set up all users with one or more baseline VersaMail account configurations. You can do so by distributing a database called __MMDevice.pdb. This feature will work in the VersaMail application, version 2.0 or later. For VersaMail 3.1 and later, the __MMDevice database version must be `0x1003`.

11.2.2.2 Overview of the MMDevice database

The MMDevice database consists of various records that set default values in VersaMail accounts. Each record follows the same basic syntax. When the VersaMail application is started on a handheld, it looks for the __MMDevice.pdb, sets account information based on the values therein, and then deletes __MMDevice.pdb. You should not confuse the file __MMDevice.pdb with _MMDevice.pdb (single underscore), which is usually created when an account configuration has changed.

__MMDevice.pdb is a case-sensitive name. It should have type and creator codes of `asc3` and a version number of 4.2 in hex (`0x0420`).

You can use the VMAccConfig application in the samples folder of the Palm OS SDK to create an __MMDevice.pdb database automatically.

11.2.2.2.1 MMDevice database record syntax

The syntax for each record in the MMDevice database is as follows:

```
set <key> <account slot> <value>
```

with one or more spaces or tab characters between each field.

- `set`

  This string literal is required at the beginning of each record.

- `key`

  A string referencing the particular account parameter you want to set. For more information, see **Section 11.2.2.3 on page 215**.

- `account slot`

  You should set this value to `0`. The account will be added to the end of the list of existing accounts. The VersaMail application accepts only a total of 8 accounts.

- `value`

  The value you want to set for the particular account parameter.

  The value is not parsed other than for keys that must be numeric. Trailing and leading whitespace are trimmed, however.

  To specify the default value for a record key, simply omit the entire record key line. If you include the record key line with a blank value, the key's value is explicitly set to an empty string that may or may not be valid for a particular record key.

For example, the following record would set the incoming mail server for the next account in the list to `mail.mac.com`:

```
set incomingServer 0 mail.mac.com
```

NOTE: The `title` key is mandatory. See **Section 11.2.2.3.12 on page 215** for more information.

11.2.2.3 MMDevice database record keys

The following MMDevice record keys are valid. The keys are not case sensitive and, unless otherwise noted, default to an empty value.

11.2.2.3.1 apn

The string that specifies the default access point name (APN) to be used for the account. This refers to a service setting set in the Network preferences panel. If not specified, the default service is used.

11.2.2.3.2 connectionType (deprecated in VersaMail 3.1)

How the connection is made with the email server. You can set it to `SyncOnly`, `PalmWireless`, or `ModemDialup`. For `ModemDialup`, the dial-up settings must be configured through the system's Network preferences panel.

11.2.2.3.3 emailAddress

The fully qualified address for the email account.

11.2.2.3.4 incomingPort

The TCP/IP access port for the incoming email server. The default value is blank, but the default TCP/IP access port for POP servers is `110`, and the default access port for IMAP servers is `143`. This value must be specified if the `serverType` value is specified. For more information, see **serverType (deprecated in VersaMail 3.1)**.

11.2.2.3.5 incomingServer

The server for incoming email. The incoming server is usually a POP or IMAP server.

11.2.2.3.6 outgoingPort

TCP/IP access port for the outgoing email server. Typically, this is the SMTP port `25`.

11.2.2.3.7 outgoingServer

The server for outgoing email. The outgoing server is usually an SMTP server.

11.2.2.3.8 password

The password for the account.

11.2.2.3.9 replyTo

The fully qualified email address for the Reply-To header of outgoing mail.

11.2.2.3.10 rootMailbox

For an IMAP email account, this specifies the root prefix of the account. This is typically not needed with most IMAP servers.

11.2.2.3.11 serverType (deprecated in VersaMail 3.1)

The protocol of the incoming email server—`POP`, `PalmDotCom`, `Enterprise`, or `IMAP`. The default for this key is `POP`. If you specify this key, you must specify the `incomingPort`, as well. For more information, see **incomingPort**.

11.2.2.3.12 title

The title of the account as shown in the VersaMail application. For example, "Personal Mail." You must enter a value for this record key.

11.2.2.3.13 useEncryptedPassword

Whether the account uses an encrypted password. Specify `YES` or `NO`. The default is `NO`.

11.2.2.3.14 useEsmtp

Whether the account requires authenticated SMTP. Specify `YES` or `NO`. The default is `NO`.

11.2.2.3.15 userName

The username for the account.

11.2.2.3.16 samcreatorid (only in VersaMail 3.1)

The creator ID of the service access module that provides the service. This key has been introduced to decide which protocol should be used with a new account. Valid choices are `'pop3'`, `'Imap'` or `'ExAs'`. The value for the key must be set or the account will default to use `pop3`.

NOTE: Values are case sensitive and require the surrounding single quotes (`''`) as part of the value string.

11.2.2.3.17 samtypeid (only in VersaMail 3.1)

The type ID of the service access module. For VersaMail 3.1, this must be `'appl'` (with the apostrophes).

11.2.3 Adding outgoing email to VersaMail® folders

This section describes the various methods used to add outgoing email to VersaMail folders. You can use the launch codes provided by the VersaMail application directly or use the Exchange Manager or Helper Notification methods documented in the *Palm OS Programmer's Companion, Volumes I and II* and the *Palm OS Programmer's API Reference*.

The direct and Exchange Manager methods work in the VersaMail application, version 2.0 or later. The Helper Notification method works in the VersaMail application, version 2.5 or later.

11.2.3.1 Overview of adding email to the Outbox

There are three basic methods for adding email to the VersaMail Outbox:

- Using the Exchange Manager

  An application can use the Exchange Manager Send command or the _send URL send scheme to sublaunch a Compose email message form. The Compose form allows the user to add text to the message and save the message in the VersaMail Outbox or Drafts folder for later sending. Using the Send command displays a list box that gives the user a choice of Bluetooth® wireless technology, SMS, or the VersaMail application to send the email. If the VersaMail application is not on the device, the list box does not contain the VersaMail option. Similarly, on devices that don't have the Bluetooth® wireless technology or SMS option, the VersaMail application is automatically launched.

The following sample code shows how an application can use the Exchange Manager to sublaunch a Compose email message form:

```
/* Use the Exchange Manager to create a New mail message */
/* Lets you add an attachment */

ExgSocketType exgSocket;
Err err = errNone;

Char      *textBuf = "test";
UInt32    size = StrLen(textBuf) + 1;

// it's important to initialize the structure to null values

MemSet(&exgSocket,sizeof(exgSocket),0);
exgSocket.description = "Testing";

/* A box pops up with an option to pick SMS, VersaMail application, etc.
 * You may not see the VersaMail option in the list if the VersaMail
 * application is not installed on the device.
 */

   exgSocket.name = "?_send:Sample.txt";

   /* send is important here */

   exgSocket.type = ".txt";
   err = ExgPut(&exgSocket);

 if (err == 0) {
       ExgSend(&exgSocket,textBuf,size,&err);
       ExgDisconnect(&exgSocket,err);
  }
```

For more information on using the Exchange Manager method, see the *Palm OS Programmer's Companion, Volume II* and the *Palm OS Programmer's API Reference*.

- Using Helper notifications

  This method supports a `SysUISwitch` from the launching application to a full VersaMail email Compose form and back again. The full compose form is opened when the appropriate control is tapped in the launching application, and once the user has saved or sent the message, they are returned to the launching application.

  For more information on the Helper notifications method, see the *Palm OS Programmer's Companion, Volume I,* the *Palm OS Programmer's API Reference,* and the sample code `AddEmail` in the Samples folder of the Palm OS SDK.

- Programmatically using the strategies documented in this chapter

  You must use the strategies documented in this chapter to allow a third-party application to add outgoing email messages to a VersaMail folder.

  All of the strategies documented in this chapter use VersaMail launch commands.

11.2.3.1.1 VersaMail launch commands

There are three basic launch commands that you can use:

- `sysAppLaunchCmdAddRecord`

  This launch command creates a basic message without an attachment. You can use `MailAddRecordParamsType` to simply add the message to the VersaMail Outbox or `MailAddRecordsParamsTypePlus` to add the message to a different VersaMail category, such as Drafts.

  The attachment functionality of `MailAddRecordsParamsTypePlus` is ignored by the VersaMail application when it is used with `sysAppLaunchCmdAddRecord`. To use the attachment functionality of `MailAddRecordsParamsTypePlus`, you must use the next launch code, `MMPRO_ADD_MESSAGE_WITH_ATTACHMENT`, instead.

  `sysAppLaunchCmdAddRecord` is available in the `systemMgr.h` header file, `MailAddRecordsParamsType` is available in the `AppCmdLaunch.h` header file, and `MailAddRecordsParamsTypePlus` is available in the `PalmVMLaunch.h` header file.

- `MMPRO_ADD_MESSAGE_WITH_ATTACHMENT`

  This launch command works much like `sysAppLaunchCmdAddRecord`, except you can use `MailAddRecordsParamsTypePlus` to specify both a category and an attachment.

  `MMPRO_ADD_MESSAGE_WITH_ATTACHMENT` is available in the `PalmVMLaunch.h` header file.

- `MMPRO_LAUNCH_CODE`

  This launch command actually launches the VersaMail application itself and presents the user with a full email compose form. The `MMPRO_LAUNCH_CODE` command uses the `MMProLaunchStruct` data structure to add attachments smaller than 64KB and `MMProLaunchStruct2` to add attachments larger than 64KB.

  `MMPRO_LAUNCH_CODE`, `MMProLaunchStruct`, and `MMProLaunchStruct2` are available in the `PalmVMLaunch.h` header file.

11.2.4 VersaMail® Font library

This section provides information on how to use the VersaMail Font library. You can use the Font library to set the VersaMail Font Picker user interface and to work directly with font information on a device. More information on the Font library can be found in the *Palm API Guide, Palm OS Platform*.

11.2.4.1 Checking whether the Font library is present

To check whether the Font library is present and loaded on a handheld, use `SysLibFind` to check for the library name `PalmSGFontLib` and a `0` error return value.

The VersaMail Font library is not included in the current versions of the VersaMail application. The library has to be retrieved, and you must perform a HotSync® operation to install it on a device. The library is available in the Palm OS SDK.

Also, the Font library requires the font databases—PalmSGHiResFonts.pdb for high-resolution devices and PalmSGLowResFonts.pdb for low-resolution devices. The font databases are available in the VersaMail application, version 2.5 or later, and in the Palm OS SDK.

11.2.4.2 Using the Font library

The Font library is loaded during many handheld events, such as HotSync operations, but in order to use the functions in the Font library, you need to obtain and store the library reference number. You can use standard methods to maintain easy access to the reference number, such as globals with associated accessor routines or Palm OS `FtrSet` and `FtrGet` calls. For more information, see `VMFontOpen`.

Also, before you use any of the Font Picker user interface functions, you must initialize the Font Picker user interface data structure using `InitFontUI`. You currently can apply only one style to a font. For example, you can apply bold or italic to a font, not bold and italic.

11.2.5 VersaMail® Attachment Plug-ins API

This section provides information on how to create a plug-in so that users can view and send email attachments using your plug-in in the VersaMail application. The VersaMail Attachment Plug-ins API is declared in the header file `PalmVMPlugin.h`.

This feature works with the VersaMail application, version 2.6 or later. The Palm OS SDK includes a sample plug-in for TXT attachments.

11.2.5.1 Overview of how the VersaMail® application handles plug-ins

The handling of email attachments in the VersaMail application is controlled by the VersaMail Plug-in Manager.

When a user attempts to view a particular attachment in the VersaMail application, the VersaMail Plug-in Manager first tries to find the appropriate plug-in given the attachment's MIME type. If multiple compatible plug-ins are found, the VersaMail application displays a list so the user can choose the plug-in they want to use.

If no plug-in is found, the Plug-in Manager attempts to find an application registered with the Exchange Manager as the default application to handle the particular MIME type. If no Exchange Manager application is found, the message "No viewer for this attachment type" is displayed and the attachment can be viewed with the plain text viewer.

When a user attempts to send a particular attachment in the VersaMail application, all installed plug-ins are queried to find out what type of attachment each of them supports. When an appropriate plug-in is found, the VersaMail application queries the plug-in to display a list of possible attachments that can be sent. When a user selects an attachment from this list, the VersaMail application calls the appropriate plug-in to provide the data required to send the attachment.

11.2.5.2 Overview of plug-in design

A plug-in should be designed as a Palm OS PRC file with the application type "mmpl" that supports the specific VersaMail launch commands detailed in this section. Because plug-ins are seamless to the user and should not appear to be separate programs, plug-ins should be created without an associated program icon. Furthermore, when a plug-in has finished viewing or sending an attachment, it should return control to the VersaMail application, not to the Applications View or to any other application.

The VersaMail application builds a dynamic list of all available plug-ins so that the user can install or remove them as desired. A separate plug-in should be provided for each type of file attachment. If there are multiple plug-ins that support the same type of file attachment, the user is provided with a list of possible plug-ins.

The VersaMail launch command that plug-ins must support are as follows:

- `MMPRO_PLUGIN_GET_INFO_LAUNCHCODE`

  This launch command is sent by the VersaMail application when it is trying to get information about what plug-ins are available to send a particular attachment.

- `MMPRO_PLUGIN_QUERY_LAUNCHCODE`

  This launch command is sent by the VersaMail application to get a list of possible attachments a user can send.

- `MMPRO_PLUGIN_EXTENDED_QUERY_LAUNCHCODE`

  This launch command is sent by the VersaMail application to get a more complete description of possible attachments a user can send. This launch command is provided for applications in which a simple, short description of a possible attachment is not enough information for a user to understand what the attachment is. For example, in Date Book, a simple date without the information associated with that date might not be enough information for a user to determine if they want to send the date as an attachment.

- `MMPRO_PLUGIN_SEND_LAUNCHCODE`

  This launch command is sent by the VersaMail application when a user has selected an attachment to send. The plug-in prepares the attachment so that it can be sent in response.

- `MMPRO_PLUGIN_RECEIVE_LAUNCHCODE`

  This launch command is sent by the VersaMail application when an attachment of the appropriate type is received. It allows the plug-in to do whatever you would like the plug-in to do with a received attachment. For example, you might want to display the attachment in a window that includes a Done button that users can tap when they have finished viewing the attachment.

This page intentionally blank

CHAPTER 12

12. Developing SDIO Applications for Palm® Handhelds

This chapter provides information on writing Palm OS® applications that interact with SDIO hardware. Because there is a wide range of possible SDIO devices, it focuses solely on those aspects of program design specific to the Palm OS application, Palm handhelds, and the SDIO slot driver.

This chapter provides:

- Pointers to the software, hardware, and documentation you'll need to create your application
- Aspects of the Palm OS that you'll use when writing your application
- Programming guidelines specific to SDIO applications
- Pointers relative to creating the SDIO card itself

Much of an SDIO application is dictated by the hardware with which it interacts. However, because SDIO is a standard, and because these SDIO applications run on the Palm OS, all such applications have a number of traits in common. This commonality is the subject of this chapter.

12.1 SD, SDIO, and MultiMediaCard specifications

The SD Card Association (SDA) publishes the *SDIO Card Specification*, which is based on and refers to the SDA document titled *SD Memory Card Specifications, Part 1, PHYSICAL LAYER SPECIFICATION*. Both of these documents provide essential foundation material for the contents of this document. You should be familiar with the *SDIO Card Specification* and with those parts of the *SD Memory Card Specifications* that document card modes, card initialization, interrupts, registers, and card reading and writing. Depending on the SDIO hardware with which you are working, additional sections of the *SD Memory Card Specifications* document may be of interest.

The SD Card Association's website can be found at **http://www.sdcard.org/**. You'll need to be a member in order to obtain the specifications from the SD Card Association.

NOTE: Creating Palm OS applications that can use and exchange data from other products via SD Memory cards is outside the scope of this document. However, to make sure that data can be interchanged with present and future SD products, refer to the appropriate SD Association specification depending on the type of application.

For developers working with MultiMediaCards, the MultiMediaCard Association's website can be found at **http://www.mmca.org/**. The MultiMediaCard specifications are available from the MultiMediaCard Association to its members.

The SDIO slot driver has been written to accommodate the following specifications:

- MultiMediaCard memory cards, V1.4 to V3.0
- SD memory cards, Part 1, V1.0 (and the supplement to part 1)

SDIO V1.0

12.2 Palm OS® SDK

General Palm OS programming concepts from ACCESS Co. LTD are documented in the *Palm OS Programmer's Companion*. Reference documentation for the APIs can be found in the *Palm OS Programmer's API Reference*. Both of these documents are installed as part of the Palm OS Software Developer's Kit (SDK), which can be found at the ACCESS Developer Network, **http://www.access-company.com/developers/**. SDIO applications are not supported on versions of the Palm OS prior to 4.0.

Although you'll want to be familiar with a number of different aspects of Palm OS programming, pay particular attention to the portions of the *Companion* and *Reference* that cover the Expansion and VFS Managers; these chapters show you how to read and write expansion media, including SD memory cards.

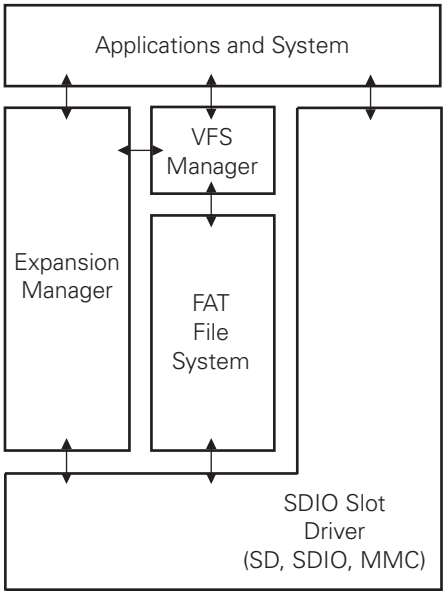
12.3 Software architecture of an SDIO application

Palm OS applications that interact with SDIO cards make use of the functions provided by the Expansion Manager, the VFS Manager, and the SDIO slot driver. Before you can write such a Palm OS application, you should have an understanding of how your application will interact with these and other features of the Palm OS software.

The following figure presents a simplified view of how the SDIO slot driver relates to your applications, the Expansion Manager, and the VFS Manager. Unlike other Expansion Manager slot drivers, the SDIO slot driver exposes its APIs to applications. Because it also lies beneath the Expansion and VFS managers, you access SDIO hardware through a combination of Expansion Manager, VFS Manager, and SDIO slot driver calls. Note that you use the VFS Manager with a given SDIO card only if there is an SD or SDIO file system present on that card.

The VFS Manager APIs are used for all file system access on an expansion card. When inserted, SD memory and SDIO CSA memory is mounted as file system memory. Therefore, access to these memory areas is done using the VFS Manager

APIs. Details of accessing data on file systems can be found in the standard Palm OS documentation on Expansion Manager and VFS APIs.



12.3.1 Expansion Manager

The Expansion Manager is a software layer that manages slot drivers on Palm OS handhelds. The Expansion Manager is not solely responsible for support of expansion cards; rather, it provides an architecture and higher-level set of APIs that, with the help of low-level slot drivers and file system libraries, support various types of media.

The Expansion Manager:

- broadcasts notification of card insertion and removal
- plays sounds to signify card insertion and removal
- mounts and unmounts card-resident volumes

NOTE: Some of the functions provided by the Expansion Manager are designed to be used by slot drivers and file systems and are not generally used by third-party applications.

For a detailed explanation of the functions that make up the Expansion Manager, see the "Expansion Manager" chapter in the *Palm OS Programmer's API Reference* at **http://www.access-company.com/developers/documents/palmos/palmos.html**.

12.3.2 VFS Manager

The VFS (Virtual File System) Manager provides a unified API that gives applications access to many different file systems on many different media types, including SD media. The VFS Manager is used for all file system access on an expansion card. In the case of an SDIO card, the VFS Manager is typically used to access any function CSA memory. The data stored in CSA memory is structured as a FAT12/16 file system

(FAT 12/16/32 for the LifeDrive™ mobile manager) and is therefore ideally suited for access by the VFS Manager.

Combo cards may contain SD memory that is also accessed through the VFS Manager APIs.

For a detailed explanation of the functions that make up the VFS Manager, see the "Virtual File System Manager" chapter in the *Palm OS Programmer's API Reference* at **http://www.access-company.com/developers/documents/palmos/palmos.html**.

12.3.3 SDIO slot driver

To simplify the interaction with the SDIO hardware, Palm has created an SDIO slot driver. It replaces the Palm OS SD/MultiMediaCard slot driver, which isn't SDIO-aware, and consists of data structures and functions that allow you to easily manage power, interrupts, and data on the SDIO card.

The SDIO slot driver controls all media supported by an SD expansion slot, including SD media, MultiMediaCard media, and SDIO media.

An examination of the functions provided by the SDIO slot driver shows that it implements most of the software functionality outlined in the SDIO Card Specification. It does not, however, support the following:

- SDIO Suspend/Resume Operation
- SDIO Read Wait Operation
- SDIO RW Extended Block Operation in "forever" mode

12.3.4 Notification Manager

The Palm OS Notification Manager allows applications to receive notification when certain system-level or application-level events occur. Although the Notification Manager has many uses, developers of SDIO applications should particularly take note of the fact that you use it to detect card removal by registering for a `sysNotifyCardRemovedEvent` (see **Section 12.4.3 on page 228**).

12.4 Guidelines for SDIO applications

All SDIO applications need to be aware of the power needs of the SDIO card. As well, they need to be able to handle interrupts generated by the card, and must be aware of when an SDIO card is inserted or removed from the handheld's SD slot. The following sections discuss these and other SDIO-application-specific topics.

12.4.1 Power management

When the handheld awakes from sleep mode, it doesn't turn the card on. Only when there is a request to access the card does it turn the card on.

12.4.1.1 Turning on card functions

You can turn on a given SDIO card function explicitly with `SDIOSetPower`. Be aware that you, as an application developer, are responsible for managing card power.

You must ensure that the total of all function hardware that is active does not draw in excess of the SDIO-specified maximum of 200ma.

Perform the following steps to explicitly turn on an SDIO card function:

1. Disable SDIO interrupts with `SDIODisableHandheldInterrupt`—even if your application doesn't use interrupts.
2. Verify that there is sufficient current available to power the card function. To aid in the power management process, the SDIO slot driver provides three functions: `SDIOGetCurrentLimit`, `SDIOSetCurrentLimit`, and `SDIORemainingCurrentLimit`.

   NOTE: These three functions do not detect or limit current draw, check the battery level, or reflect how much energy the battery has left.

   The current limit for each function can be obtained by calling `SDIOGetCurrentLimit` or changed by calling `SDIOSetCurrentLimit`. Prior to enabling power to a given function, call `SDIOGetCurrentLimit` to determine how much power it will draw, and compare it to the value returned from `SDIORemainingCurrentLimit`, which indicates how much current can be spared.
3. Turn the function on using `SDIOSetPower`.
4. Reenable interrupts by calling `SDIOEnableHandheldInterrupt`.

   After turning off an SDIO card function (with `SDIOSetPower`), be sure to call `SDIOSetCurrentLimit` and set its current limit to zero.

   When a card is removed, all of the in-memory current limits are automatically set to zero.

12.4.1.2 Auto power off

The `SDIOSetAutoPowerOff` function allows you to specify an amount of time after which the power and data signals to a given function on an SDIO card should be turned off. You specify this time interval in system ticks; there are `SysTicksPerSecond` ticks per second. To disable the auto-power-off feature, simply call this function and supply a tick count of zero.

To obtain the current auto-power-off settings for a given SDIO card function, use `SDIOGetAutoPowerOff`.

12.4.1.3 Callbacks

The SDIO slot driver allows your application to register callback functions that will be invoked whenever the corresponding event occurs on the SDIO card. Several of these callbacks relate to power management.

Whenever the Palm handheld is about to be put to sleep, the callback function corresponding to `sdioCallbackSelectSleep` is called. Just after the handheld wakes, the function corresponding to `sdioCallbackSelectAwake` is called. These callback functions can be called from either an interrupt routine or a non-interrupt routine; as a result interrupts may be disabled or enabled. In either case, they should always be as fast as possible.

Whenever SDIO card power is turned on or is about to be turned off, the callback function corresponding to `sdioCallbackSelectPowerOn` or `sdioCallbackSelectPowerOff`, respectively, is called. While processing these functions, never call `SDIOSetPower` in order to turn an SDIO card's power on or off. These functions can be called from within an interrupt handler, so they should be as fast as possible.

12.4.2 Interrupt handling

An SDIO card is capable of interrupting the host device into which it is inserted—in this case, the Palm handheld. The SDIO slot driver allows you to register a callback function that is called whenever the card interrupts the handheld.

Register for the interrupt callback by calling `SDIOSetCallback` and specifying that you are registering for `sdioCallbackSelectInterruptSdCard`. In your callback function, be sure to reset the interrupt source to prevent the interrupt callback from being called again inadvertently.

Whether or not you have registered an interrupt callback function, you can enable or disable the SDIO interrupt on the handheld by calling `SDIOEnableHandheldInterrupt` or `SDIODisableHandheldInterrupt`. Note that these functions only affect interrupts on the handheld; they do not turn on or off interrupts on the SDIO card itself.

These functions are implemented as an incrementing counter, making them re-entrant. For instance, for every call to `SDIODisableHandheldInterrupt` there must be an equal number (or more) of calls to `SDIOEnableHandheldInterrupt` in order to re-enable interrupts.

By default, when the card is inserted interrupts on the handheld are enabled, but are disabled internally until an interrupt callback is set with `SDIOSetCallback`. Note that in order to receive the SDIO interrupt, power to the card must be on, even if the handheld is asleep.

12.4.3 Detecting card insertion and removal

Applications that depend on the presence of the SDIO card in the slot should register for a `sysNotifyCardRemovedEvent`, which is broadcast when the user removes the card from the SD slot.

Be sure to unregister for the `sysNotifyCardRemovedEvent` notification and any SDIO callbacks when your application terminates.

For more information on registering and unregistering for notifications, see the "Notification Manager" chapter in the *Palm OS Programmer's API Reference*. The "Expansion" chapter of the *Palm OS Programmer's Companion*, vol. I discusses the various notifications that are issued when a card is inserted or removed, or when a volume is mounted or unmounted.

Both documents can be found at the ACCESS Developer Network at **http://www.access-company.com/developers/documents/palmos/palmos.html.**

12.4.4 Auto Run

When a card is inserted into the SD slot, after it has been initialized any file system memory present on the card is mounted by the Expansion Manager. This includes all SD memory, in the case of a standard SD card or SDIO combo card, and all SDIO Function CSA memory for functions 0-7.

After mounting of the file systems, the SDIO slot driver broadcasts a series of Auto Run (`sysNotifyDriverSearch`) notifications. These notifications are sent in an attempt to locate function- or card-specific drivers, and allow those drivers that are already on the handheld to launch themselves.

The typical sequence of events after a card is inserted is as follows:

1. Power is applied to the card.
2. The card is initialized according to the SDIO, SD, or MultiMediaCard specification, as appropriate.
3. Information about the card (tuples, clock speed, CSD, CID, etc.) is read.
4. Any recognized file systems are mounted.
5. `sysAppLaunchCmdCardLaunch` is sent to `start.prc` on each mounted file system.
6. The Auto Run notifications (`sysNotifyDriverSearch`) are sent.
7. `sysAppLaunchCmdNormalLaunch` is sent to `start.prc` on each mounted file system.

For SDIO cards, one Auto Run notification is broadcast for the SD memory portion of a combo card, and an additional notification is broadcast for each card function (up to 7). For SD memory and MultiMediaCard memory cards, only one such notification is sent. The notifications are sent starting with SD memory, followed by function 7 (if there is one) and proceeding to function 1 as appropriate.

The `notifyDetailsP` field of the `SysNotifyParamType` structure that accompanies the Auto Run notification points to an `AutoRunInfoType` structure. Each driver that has registered for `sysNotifyDriverSearch` should examine the contents of the `AutoRunInfoType` structure to determine if it is the driver that should control the inserted card. If so, the driver should then check the `SysNotifyParamType` structure's `handled` field. If `handled` is set to `true`, another driver has received the broadcast and will control the card. If `handled` is set to `false`, the driver should set it to `true` to indicate that it will control the device.

12.5 Developing the SDIO peripheral

An SDIO application is only as good as the hardware with which it interacts. The following sections provide some tips for the creation of an SDIO peripheral to be used with a Palm handheld.

12.5.1 Specifications

When developing an SDIO peripheral, it is extremely important that you following the specifications identified in **Section 12.1 on page 223**. Be sure to pay close attention to the power restrictions, as the Palm handheld isn't able to deliver more power to an SDIO peripheral than the specification maximum.

12.5.2 SDIO slot driver

A Palm handheld running Palm OS 4.0 or 5.0 supports SD/MultiMediaCard expansion cards. If the SDIO slot driver is installed, it will also support SDIO expansion cards. In both cases, only one file system can be mounted for a given expansion card. Future versions of the Palm OS will likely lift this restriction, allowing up to seven file systems to be mounted for an SDIO expansion card.

In order to support SDIO peripherals, handhelds running Palm OS 4.0 must either be flash-upgraded to a version of the OS that supports SDIO, or must have the SDIO slot driver separately installed in RAM. The SDIO slot driver can be downloaded from the Palm website and installed as a PRC file in RAM on Palm OS 4.0 devices. After a soft reset, the slot driver in RAM is recognized and takes precedence over the SD/MultiMediaCard slot driver in ROM.

NOTE: Devices based on Palm OS 5.0 have the SDIO slot driver built into the ROM.

You can verify whether a given slot driver is "SDIO-aware" by calling `SDIOAPIVersion`. This function returns `expErrUnimplemented` if the specified driver doesn't support SDIO, or errNone if it does. If the driver does support SDIO this function also returns the slot driver version number through the `versionP` parameter.

To remove the SDIO slot driver from RAM, you must perform a hard reset of the handheld. You cannot delete the SDIO slot driver using the Application Launcher's "Delete" function. Note that to avoid having the SDIO slot driver reinstalled on the handheld during the next HotSync operation, you must remove the slot driver PRC from the Backup directory of your desktop computer.

12.5.3 SDIO card initialization and identification on Palm OS

The process of identifying and initializing an SDIO card is specified in the *SDIO Card Specification*. One of the first steps in developing an SDIO card is to have the card identify itself as an SDIO card to the host.

12.5.3.1 Identification

Identification of a card is done only once, at the time the card is inserted in the handheld's SD slot. Information obtained from the card during the identification phase is retained in the handheld's memory until the card is removed. Among other things, this information includes:

- the type of card in the slot
- what the card contains
- the card's limits
- data read from tuples

By default SDIO cards power-off automatically after a certain amount of inactivity. This behavior can be modified with the `SDIOSetAutoPowerOff` function.

12.5.3.2 Initialization

A card is initialized every time it is turned on. The SDIO slot driver follows the appropriate initialization flowchart—SD mode or SPI mode—from the "SDIO Card Initialization" section of the *SDIO Card Specification* to initialize the card.

During the initialization phase, the handheld operates within the range of SD or SPI clock frequencies specified in the SD Memory Card Specifications (from zero to 400kHz). The actual clock frequency used depends upon the model of the Palm handheld.

The TPLFID_FUNCTION tuple, located immediately after the CISTPL_FUNCID tuple in the CIS for function 0, contains the TPLFE_MAX_TRAN_SPEED byte. This byte indicates “the maximum transfer rate per one data line during data transfer”; essentially, the maximum clock frequency that the card can support. As soon as this tuple is read, the SDIO slot driver increases the clock speed to the highest possible frequency that doesn’t exceed the maximum specified in TPLFE_MAX_TRAN_SPEED.

12.5.4 Code Storage Area (CSA)

In order for an SDIO card’s Code Storage Area (CSA) to be readable by the Palm OS software, the CSA should be in FAT12/16 format (FAT 12/16/32 for the LifeDrive™ mobile manager), and any drivers, data, or applications that the peripheral would like to be automatically detected by the Palm handheld should reside in the `/Palm` and `/Palm/Launcher` directories. Once the CSA area is mounted, applications may access any data within the CSA, irrespective of the directory in which that data resides.

This page intentionally blank

PART III

Debugging

This part of the guide provides details on how to debug problems with the code you create.

This page intentionally blank

CHAPTER 13

13. Debugging

This chapter details how to debug problems with Palm APIs using tools and utilities available for Palm smartphones.

13.1 Overview

Available on:

- Centro™ and Treo™ smartphones

The following sections explain the tools and methods available for developing and debugging applications including:

- Simulators
  - Release Simulator
  - Debug Simulator
- DebugPrefs
- Metrowerks CodeWarrior
- PalmDebugger
- Garnet OS Developer Suite (also called Palm OS Developer Suite, or PODS)

13.1.1 Hardware requirements

There is no USB support for debugging on Treo 600 smartphones, so you will need to use a serial cable for debugging.

The Treo 650, Treo 700p, and Treo 755p smartphones support USB debugging for 68K applications.

For serial debugging on the Treo 650, Treo 700p, and Treo 755p smartphones, you will need the Treo 650 smartphone serial cable adapter and a Treo 600 smartphone serial cable.

13.1.2 Simulator vs. on-device debugging

When debugging problems with your applications, it may be more effective to use the simulator on a PC or actual on-device debugging, depending on the situation.

Simulators are useful for:

- Single stepping through 68K source code
- Checking for memory leaks
- Checking basic functionality
- Checking quick changes to your code (for example, to the user interface)

- Discovering if a crash is caused by an inappropriate API call (such as passing on the wrong parameter)
- Checking database and recording status

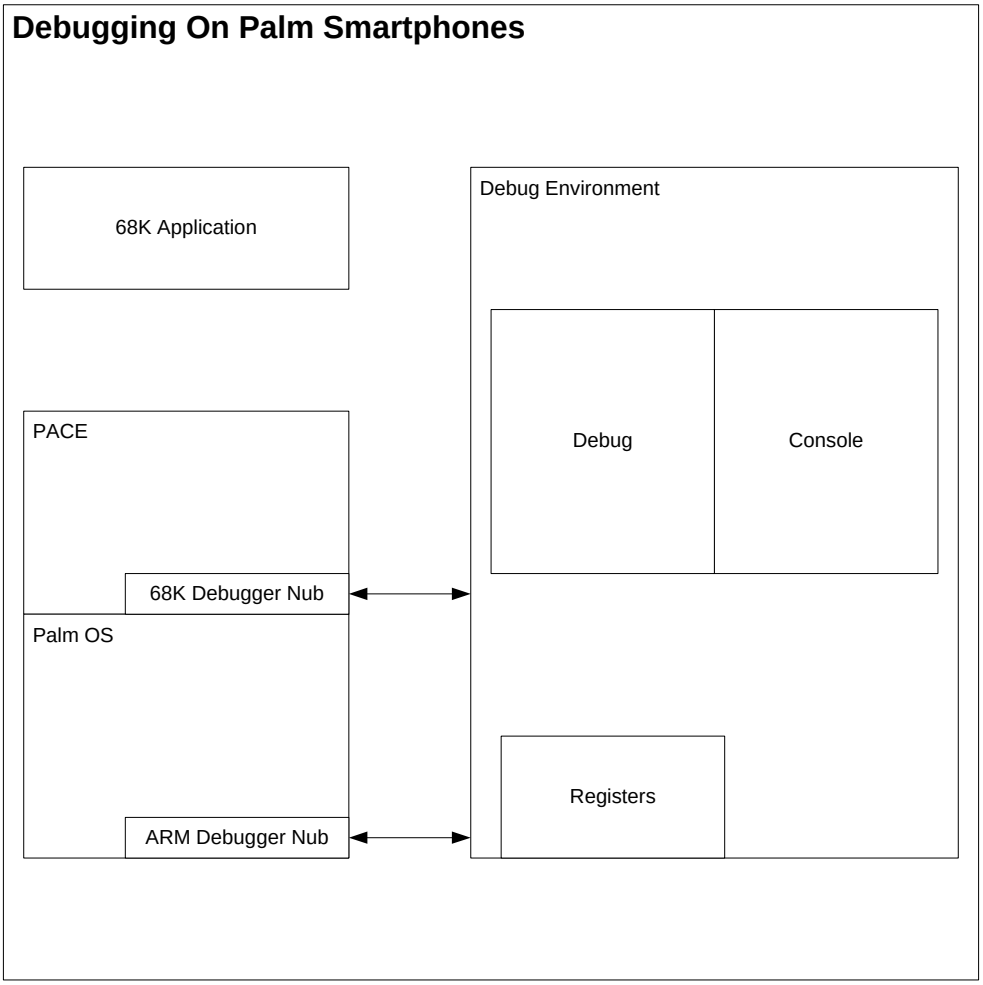
On-device debugging is useful for :

- Issues related to the radio (such as voice, data, or Bluetooth)
- Checking for memory corruption
- Watching variables
- Obtaining stack traces
- Testing the speed of the application and connections

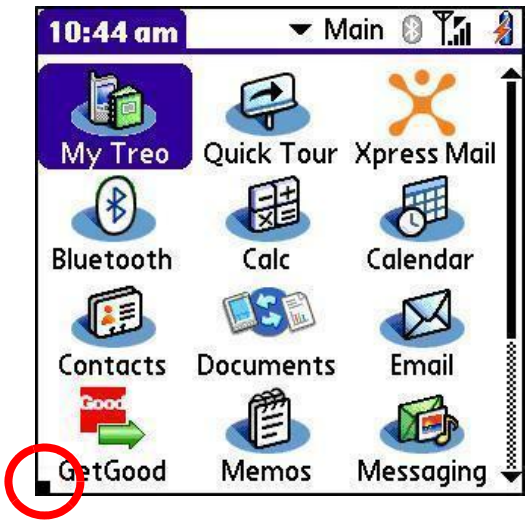
13.1.3 Debugger modes

The following section assumes that you understand some background information on the Palm OS 5.0 architecture and environment, including details of the relationship between the 68k and PACE layers.

The following figure shows the various debugger nubs and where they are located within the Palm OS on a Palm smartphone.



13.1.3.1 Debug mode

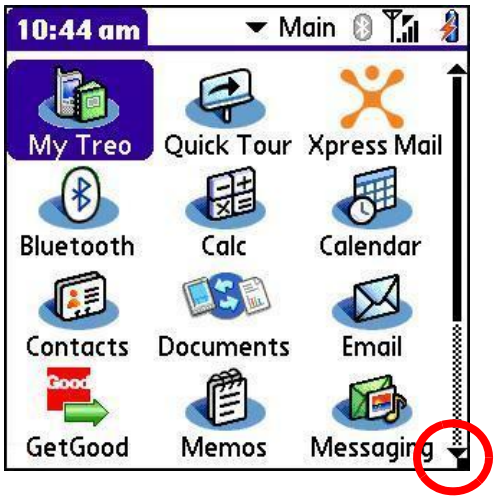


The 68K PACE Debug mode is active if you see a blinking block in the lower-left corner of the screen. In debug mode, there is no response to keyboard or touch screen input. In debug mode, the device is ready to connect to PalmDebugger and receive debugging commands.

To connect, select the shortcut character, enter a period (.), and then press **Option-1**.

To exit console mode, soft reset the device by removing the battery and replacing it, or do a pin reset.

13.1.3.2 Console mode



The 68K PACE Console mode is active if you see a blinking block in the lower-right corner of the screen. In console mode, the device works as usual, and will be ready to connect to the CodeWarrior debugger or PalmDebugger for 68k source code debugging.

To connect, select the shortcut character, enter a period (.), and then press **Option-2**. Alternatively, press **Shift+HotSync** to activate 68K console mode.

To exit console mode, soft reset the device by removing the battery and replacing it, or do a pin reset.

NOTE: The PACE debugger nub stops the PACE environment, not the whole system.

13.2 Simulators and emulators

Simulators and emulators are tools that help developers write applications for Palm mobile devices. At the most basic level, simulators and emulators are software programs that enable the execution of an application written for a different computer environment. They allow developers to test and use their applications when they do not have a device.

The most useful feature is the ability to view changes to an application on the fly (for example, to the user interface). Simulators and emulators assist with many tasks, and are useful for developing and debugging applications for Palm devices running Palm OS.

13.2.1 What's the difference?

Specifically, a simulator is a software application that imitates another system. For example, a Palm simulator will allow you to run an application written for a mobile device running Palm OS on a PC. Just keep in mind, when using a simulator, you may see slight differences in the application's behavior in certain classes of error messages compared to the actual behavior of a target system. For Palm OS simulators, this difference is due to the fact that simulators are compiled to run on a PC, and don't attempt to emulate ARM hardware. But even with these differences, simulators are still useful tools for software development.

An emulator is a software application that goes a step further than a simulator by attempting to duplicate the exact behavior of a system, including its hardware. Palm OS emulators (for Palm OS 4 and earlier) emulate 68k-based hardware and run Palm ROM system images. The focus of an emulator is an exact representation of one system on another. For this reason, in theory, an emulator is a closer approximation to actually running an application on a device than a simulator.

The following table lists some differences between simulators and emulators:

Task	Simulator	Emulator
Imitate software	●	●
Emulate hardware	-	●
Run target device binary	-	●
Represent memory accurately	-	●

13.2.2 Where can I get them?

Palm's device-specific simulators are available for download from PDN on the device's home page. For example, to find a simulator for the Palm Centro smartphone, from the PDN home page, navigate to **Develop > Devices > Palm Centro**. You'll find the simulator in the Quick Links section of the Centro device page. Click the link to download the simulator you need. To find all available simulators, go to the PDN Knowledge Base home page, then from the **Category** drop-down menu, select **Downloads**. Sort the results by Summary, and all available simulators will be presented on one page alphabetically.

13.2.3 Palm OS simulators and emulators

Generic simulators and emulators for the Garnet 5.4 operating system are provided by ACCESS at the ACCESS Developer Network website.

At the Palm Developer Network (PDN) website, Palm, Inc. provides simulators and emulators that include Palm differentiations not included in the ACCESS tools. Device-specific simulators are available for all current Palm devices, while Palm provides emulators only for Palm devices running Palm OS 4 and earlier.

Two versions of Simulators are provided for each device: the Release Simulator and the Debug Simulator.

- The Release Simulator simulates the production device, where fatal errors are not displayed to end users.
- The Debug simulator is a tool for developers that displays all the otherwise hidden fatal and non-fatal alerts and errors that a production device and the Release Simulator would skip. Developers should use the Debug Simulator for best results.

13.2.4 Release Simulator

This section describes the Palm smartphone version of the Palm OS® simulator.

The Palm OS smartphone simulators simulate Treo UI, 5-way navigation, and basic functionality of the production device. It also allows for network redirecting to your local host TCP/IP.

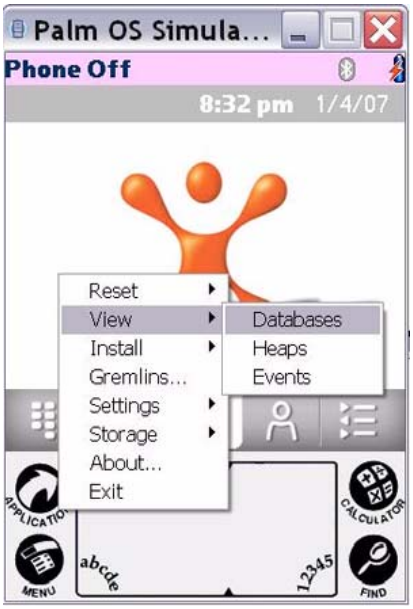
When using the Release Simulator, you may see slight differences in behavior and error messages, but even with these differences the simulator is still a useful debugging tool, especially because the Release Simulator allows you to load and run a newly generated PRC quickly.

The Release Simulator application name is **PalmSim.exe**. Simulators are available for most Palm products, and can be found on the device's home page at the Palm Developer Network (PDN) at **http://pdn.palm.com**.

13.2.4.1 Using the Release Simulator

To launch an application on the Release Simulator, simply drag and drop the .prc file of your application onto the Simulator.

The simulator includes several views that make it useful for debugging, such as the Databases, Heaps, and Events views.



To select a view, use the following steps:

1. Launch the simulator.
2. Right-click the simulator.
3. Select **View**.
4. Choose **Events, Heaps** or **Databases**.

NOTE: To dump a heap, double click it in the Heaps view.

The Gremlins tool is useful for debugging applications that have been created from scratch, testing random events, and basic functionality testing. To use the Gremlins tool:

1. Right-click the simulator.
2. Select **Gremlins**

For more information on the Release Simulator for Palm devices, see the ACCESS Developer Network at **http://www.access-company.com/developers/**.

13.2.4.2 Keystroke equivalents

Simulators do not use device skins. For this reason, it is useful to know how to simulate the possible keystrokes of a device.

The following table decodes the keystrokes that can be simulated in the Palm smartphone version of the Palm OS simulator:

Right-Shift	Option	
Left-Shift	Shift	
Right-Ctrl	Menu	
Left-Ctrl	Various virtual characters:	
	Left-Ctrl-A	vchrMenu
	Left-Ctrl-B	vchrLowBattery
	Left-Ctrl-C	vchrCommand
	Left-Ctrl-D	vchrConfirm
	Left-Ctrl-E	vchrLaunch
	Left-Ctrl-F	vchrKeyboard
	Left-Ctrl-I	vchrFind
	Left-Ctrl-K	vchrCalc
	Left-Ctrl-N	vchrNextField
	Left-Ctrl-P	vchrPrevField
	Left-Ctrl-L	chrCarriageReturn
Esc	vchrHardPower (radio power)	
F1	vchrHard1 (Phone)	
F2	vchrHard2 (Calendar)	
F3	vchrHard3 (Messaging)	
F4	vchrHard4 (Screen power)	
End	hsChrSymbol (Alt)	
Home	vchrRockerCenter	(application key)
Clear	vchrRockerCenter	
Left-Arrow	vchrRockerLeft	

Right-Arrow	vchrRockerRight
Up-Arrow	vchrRockerUp/vchrPageUp (depending on focus mode)
Down-Arrow	vchrRockerDown/vchrPageDown (depending on focus mode)
Page-Up	hsChrVolumeUp
Page-Down	hsChrVolumeDown
F5	Toggle coordinate display in title bar
F7	Sticky-shift (toggles press/release of shift key)
F8	Sticky-Option (toggles press/release of option key)
F9	Plug-in/Unplug a simulated charger
F11	Increase simulated battery charge by 1%
F12	Decrease simulated battery charge by 1%
Left-Ctrl-R	Soft Reset
Shift-Left-Ctrl-R	Hard Reset
Left-Ctrl-S	Power off (simulates down/up of Esc key)

13.2.4.3 Simulator debugging tips

- Ignore the error, "First-chance exception in PalmSim.exe (SYSTEM.DLL):0xC0000005: Access Violation"
- If your interface will not update, drag and drop the .prc file onto the simulator application again.
- If the simulator displays the message, "File already in use," go to the Task Manager and end all PalmSim.exe processes.

For more information on the Release Simulator for Palm devices, see the ACCESS Developer Network at **http://www.access-company.com/developers/**.

13.2.5 Debug Simulator

The Debug simulator is a tool for developers that displays all the otherwise hidden fatal and non-fatal alerts and errors that a production device and the Release simulator would skip.

The Debug Simulator is available for most Palm products, and can be found on the device's home page at the Palm Developer Network (PDN) at **http://pdn.palm.com**.

13.3 DebugPrefs

As discussed in **Section 13.2.5 on page 242**, on production devices, errors are hidden from the user for a better user experience. However, for developers, the DebugPrefs application will expose those errors for accurate debugging information.

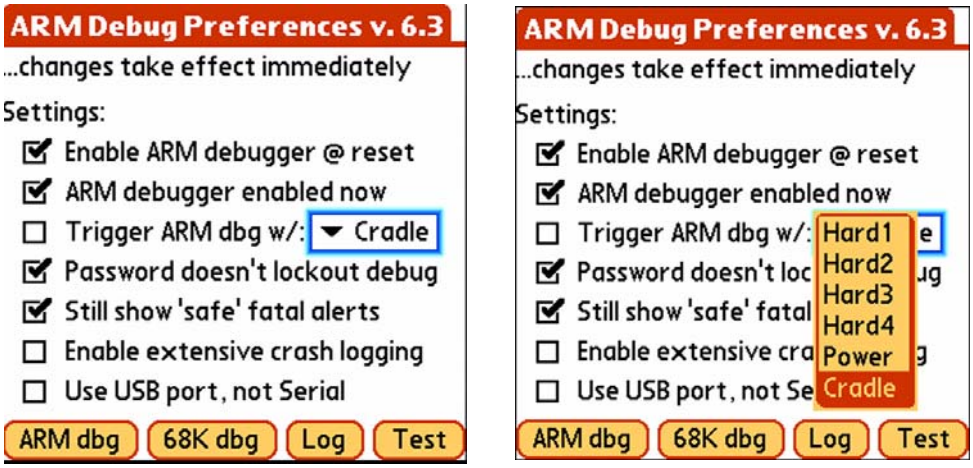
The DebugPrefs application is the file **DebugPref.prc**, which is available in the *ToolsAndUtilities* folder of the Palm OS SDK at the Palm Developer Network at **http://pdn.palm.com**.

DebugPrefs is an application that runs on the device, which allows you to configure how you want a device to behave when an error occurs. DebugPrefs has been updated to version 6.3. The debugging features are discussed in the following sections.

The DebugPrefs application is useful for:

- Post-crash debugging
- Disabling silent resets

13.3.1 Checkbox settings



The DebugPrefs application includes the following checkbox settings:

- **Enable ARM debugger @ Reset -** This option puts the smartphone in debugger mode every time the system resets or crashes. This checkbox is required to be selected for post-crash debugging. When you see a silent-reset, a crash has occurred. This behavior indicates that something in your application caused the device to crash, and you should investigate your code.

  One of the following actions will take place when a fatal error occurs:

  – The ARM Debugger nub will start using the serial port

  – The 68K PACE Debugger Nub will start using the serial or USB port

- **ARM debugger enabled now -** This option is used to enable the ARM debugger. This box is NOT checked by default, but should be selected to do post-crash debugging. (For more information on ARM development, refer to the ACCESS Developer Network at **http://www.access-company.com/developers/**.)
- **Trigger ARM dbg w/ -** This option allows the user to choose a "trigger" that will drop the user in the debugger. The user no longer needs to switch to DebugPrefs or press and hold the HotSync® button. This trigger requires the event loop to be running. The options for the trigger are:
  - Cradle (HotSync®)
  - Power
  - Hard1
  - Hard2
  - Hard3
  - Hard4
- **Password doesn't lockout debug -** Check this option to enable the usual debug triggers even if you have set a password for a smartphone. This checkbox is required to be selected for post-crash debugging.

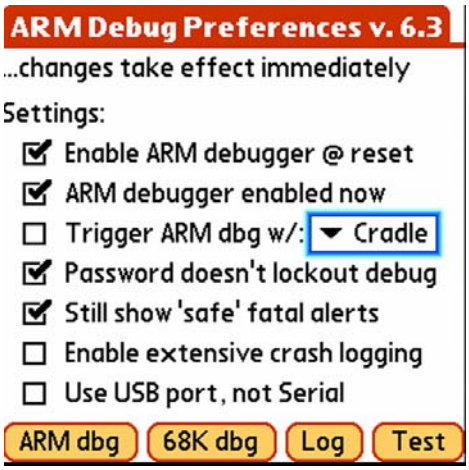
  NOTE: This option doesn't make a smartphone any less secure, because it is always possible to programmatically launch the debugger if you have infrared (beaming) access to a smartphone.

- **Still show "safe" fatal errors -** If the system determines that an error is not a major error, then it displays the error. If it determines it is a major error, the system either automatically resets the smartphone or launches the smartphone in debugger mode, depending on how the **Enable ARM debugger @ Reset** setting is set. This checkbox is required to be selected for post-crash debugging.
- **Enable extensive crash logging -** This option enables a higher level of log functionality.
- **Use USB port, not Serial -** This option allows you use the USB port for debugging instead of the Serial port. This setting must be consistent with the setting of the debugger you are using on your system.

To do post-crash debugging, make sure that the following basic DebugPrefs features are selected:

- Enable ARM debugger @ reset
- ARM debugger enabled now
- Password doesn't lockout debug
- Still show 'safe' fatal alerts

13.3.2 Button options

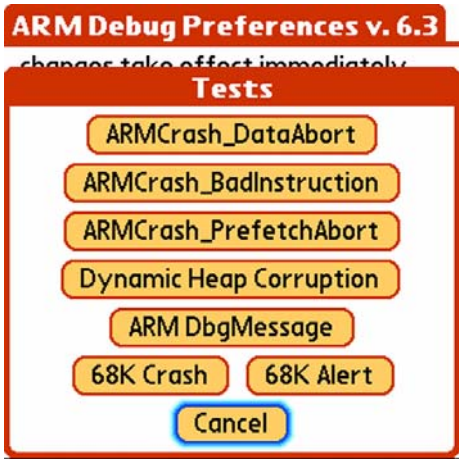


The following options are available by tapping the buttons at the bottom of the screen:

- **ARM dbg -** This button immediately launches the 68K PACE debugger nub.
- **68K dbg -** This button immediately launches the ARM debugger nub.
- **Log -** This button shows when the device crashed and the application that caused it to crash.

  NOTE: This feature is particularly useful because, as mentioned earlier, a Palm smartphone does not display fatal errors when they occur; it simply performs a soft reset. This log will display the last logged time the error happened.

- **Test -** This button simulates one of the following errors:



  - ARMCrash_DataAbort
  - ARMCrash_BadInstruction
  - ARMCrash_PrefetchAbort
  - Dynamic Heap Corruption
  - ARM DbgMessage
  - 68K Crash
  - 68K Alert

13.4 Metrowerks CodeWarrior

Metrowerks CodeWarrior for Palm OS v9 allows you to do source debugging on a Palm OS simulator or on-device debugging on a Palm smartphone. It is also useful for pre-crash debugging. You can connect CodeWarrior with the device using both serial and USB connectors.

CodeWarrior, available for Windows and Mac OS, is a complete programming tool for the Palm OS platform. CodeWarrior is available at the following URL:

http://www.freescale.com/webapp/sps/site/homepage.jsp?nodeId=012726

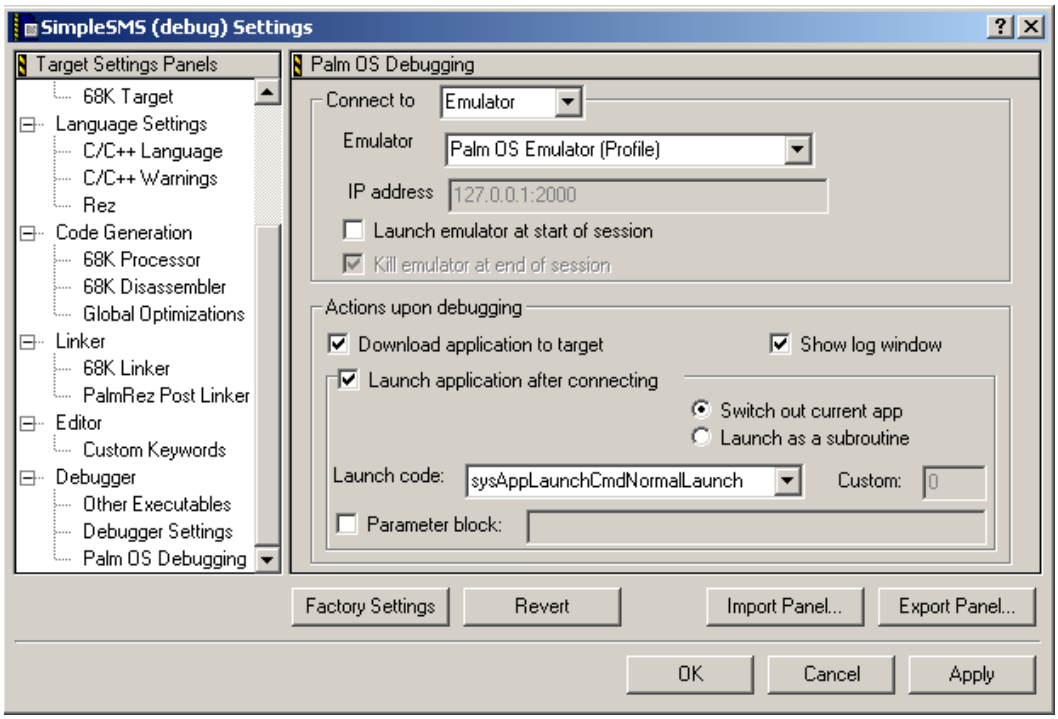
CodeWarrior is useful for:

- Setting breakpoints
- Stepping through or into code
- Viewing memory
- Changing variable values while debugging
- Changing PC values (right-click your application and select "**Change Program Center**"
- Launching your application with different launch codes, such as:
  - `sysAppLaunchCmdNormalLaunch`
  - `sysAppLaunchCmdFind`
  - `sysAppLaunchCmdSystemReset`
- Visual UI design

NOTE: For more information on launch codes, see the ACCESS Developer Network at the following URL: **http://www.access-company.com/developers/documents/docs/palmos/PalmOSReference/AppLaunchCodes.html#1012458**.

CodeWarrior cannot be used for post-crash debugging, and it cannot be used to check CPU registers.

13.4.1 Simulator debugging using CodeWarrior



To debug using CodeWarrior, use the following steps:

1. Select the project settings to build your project object code with symbolic information:

   Settings > 68K Linker > Debugger Info > Generate SYM File

2. Configure the project settings as follows:

   Settings > Palm OS Debugging > Connect to > Emulator

3. Launch the Palm OS simulator

4. Load your application and its symbols and run it by selecting **Project > Run** (or press **Ctrl + F5**). CodeWarrior displays a dialog box notifying you that the application is being loaded to the target.

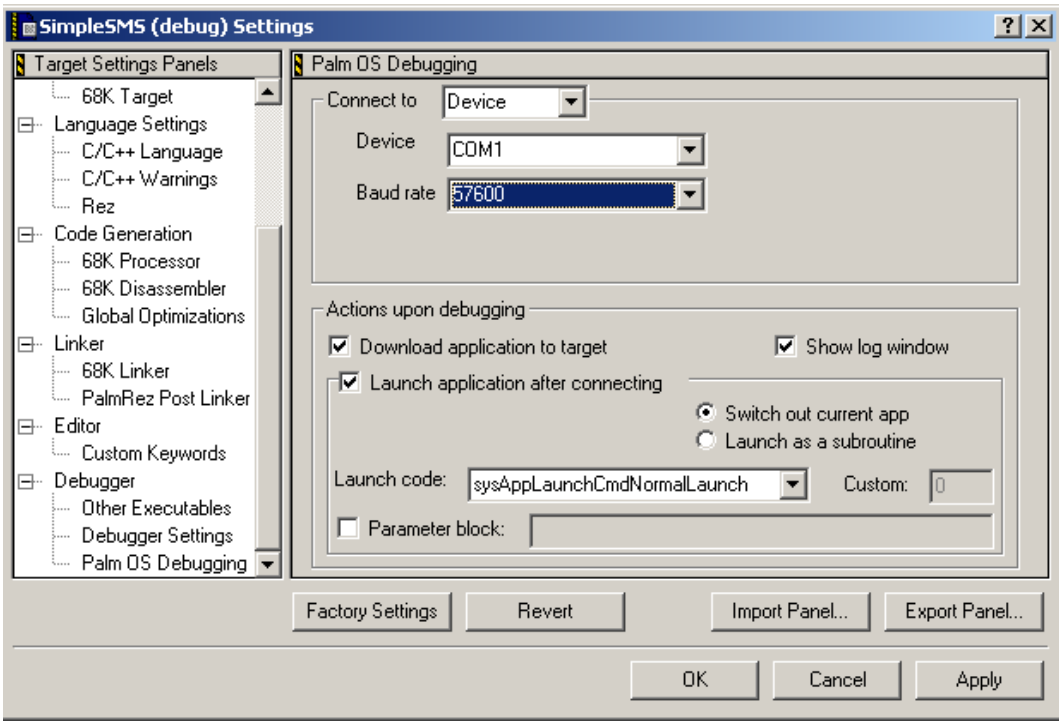
   – **Loading the binary -** The .prc is loaded onto the simulator when the debugging session is initiated.

   – **Loading the symbol -** The debugger can only be initiated from the project build environment (.mcp).

5. Set any breakpoints you need, and begin your debugging session.

6. To debug within your code, select **Project > Debug** (or press **F5**).

13.4.2 On-device debugging using CodeWarrior



To debug with CodeWarrior on the device, use the following steps:

1. Select the project settings to build your project object code with symbolic information:

   Settings > 68K Linker > Debugger Info > Generate SYM File

2. Connect via USB or Serial port (USB is faster)

   Settings > Palm OS Debugging > Connect to > Device

   Device > COM1 or USB (for serial or USB connection, respectively)

NOTE: The Treo 600 smartphone only supports serial debugging

3. Configure the project settings and place the Palm smartphone into Console mode by pressing **Shift + HotSync**, or use the following method to choose between Console and Debug mode:

   – Press **Option + Shift + Find** to open the Find dialog box.

   – Press **S**, and then press **Alt**.

   – The shortcut character appears as an option, usually at the bottom of the screen.

   – Select the shortcut character, enter a period (**.**), and then press **Option-1** or **Option-2** to switch to Debug or Console mode, respectively.

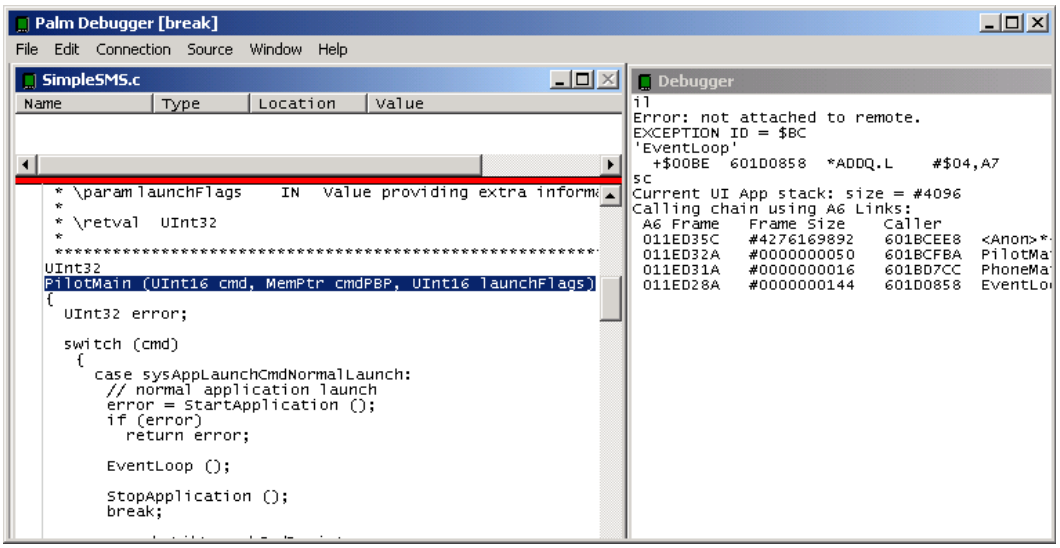
4. Load your application and its symbols and run it by selecting **Project > Run** (or press **Ctrl + F5**). CodeWarrior displays a dialog box notifying you that the application is being loaded to the target.
   - **Loading the binary -** The .prc is loaded onto the simulator when the debugging session is initiated.
   - **Loading the symbol -** The debugger can only be initiated from the project build environment (.mcp).
5. Set any breakpoints you need, and begin your debugging session.

13.4.3 CodeWarrior debugging troubleshooting tips

- If you are having trouble using the USB cable, make sure you have moved usbport.dll from *c:\WINDOWS\system32* to *c:\program files\palmOne* (or *Palm*). Do NOT just copy the file.
- If you are having trouble connecting to Code Warrior, check the following items:
  - When you are using the serial cable, make sure to select the correct COM or USB port and baud rate (**57600**).
  - Make sure that the COM or USB port is not currently being used by other applications
  - Make sure that the serial cable is connected to the right COM port.
  - Be aware that if the application is large, the debugger will be slow.

  If these steps do not work, try the following:
  - Shut down CodeWarrior and restart it.
  - Reset the device
- If you see an error saying that a database is already running on the device, make sure that the application is not running, and delete the existing .prc file if necessary.
- To prevent the device from going to sleep, use the following method:
  - Press **Option+Shift+Find** to open the Find dialog box.
  - Press **S**, and then press **Alt**.
  - The shortcut character appears as an option, usually at the bottom of the screen.
  - Select the shortcut character, enter a period (**.**), and then press **Option-3**.
- CodeWarrior cannot be used for post-crash debugging, and it cannot be used to check CPU registers.

13.5 PalmDebugger



PalmDebugger is a modified version of the **PalmDebugger.exe** application. It allows debugging for GCC-complied 68K code in Palm OS 5 and includes some post-crash debugging ability. The latest version is available as part of the Palm OS SDK in the folder *ToolsAndUtilities*. Download it from PDN at **http://pdn.palm.com**.

To support debugging on Palm smartphones, the PalmDebugger has been updated to communicate with the Palm OS® PACE and the system core.

PalmDebugger is useful for:

- Source level debugging
- Importing .prc and .pdb files
- Post-crash debugging (using the serial cable only)
- Viewing memory
- Listing the database
- Exporting database files
- Heap dumps
- Check CPU registers

13.5.1 Source level debugging

To use PalmDebugger for source level debugging, follow these steps:

1. Build your project object code to generate symbolic information (filename.sym). Compile your code with a command resembling the following example:

   ```
   m68k-palmos-gcc -O2 -g -Wall SimpleSMS.o -o Obj/SimpleSMS.sym
   ```

   For more examples, download the Sample Code from the SDK, available at PDN at **http://pdn.palm.com**.

2. Connect your device to the serial or USB cable, then use one of the following steps:
   - To connect to the Palm OS simulator, launch PalmDebugger, select Connection as Emulator, and then launch the Palm OS simulator.
   - To connect to a Palm smartphone, launch PalmDebugger and select Connection as Serial, and then put the Palm smartphone into Debug or Console mode using the method described in **Section 13.4 on page 246**.

NOTE: USB debugging is not supported for Treo 600 smartphones.

3. Install the database or databases that represent your application, and load its symbols from the Source menu or by using the F8 hot key.
4. Open or load the appropriate .prc, .sym, and .c files. (Ignore the dialog that is displayed asking to locate the gdbstub.c file by selecting Cancel.)
5. Set any breakpoints you need, and begin your debugging session.

13.5.2 Post-crash debugging

To use PalmDebugger for post-crash debugging and display 68K traces, use the following steps:

1. Configure PalmDebugger to use the serial port.
2. Make sure your serial port isn't being used by something else. Check HotSync and Palm OS Debugger.
3. Using the Treo 600 serial cable, connect the Treo 650 serial adapter to the COM port.
4. In the "Debugger" window, enter the command `"att"`.

NOTE: You may need to retry a few times by shutting and restarting PalmDebugger. PalmDebugger will return the `<pc>` message when it is successfully attached.

For more information on post-crash debugging, refer to the Application Note, “Using PalmDebugger” on PDN.

For more information, refer to documentation on the ACCESS website at:

http://www.access-company.com/developers/documents/docs/devguide/UsingDebugger.html#997418.

13.5.3 Common PalmDebugger commands

The following table defines commonly used PalmDebugger commands:

Command	Definition
att	Attaches a device to the debugger
dm <addr> <numBytes>	Reads arbitrary memory addresses
hd 0	Dumps the dynamic heap. (This command is slow.)
hd 1	Checks the DBcache state. ■ Displays feature pointers and loaded resources or records. It works very slowly via serial (takes approximately 45 minutes) ■ Displays labeled chunks that belong to particular applications: – D (dirty) vs. d (not dirty) – M (moveable) vs. m (not moveable) ■ Displays sizes in hexadecimal values.
Help	A console command list
il pc-20 40 \bytes	Disassembles around the current PC
opened	Shows opened databases
Ping	Determines if the connection in the console window is still valid
reg	Emulates CPU registers
sc a6 20	Attempts a 68K stack crawl
wh \a pc	Identifies the database containing the PC. (This command is slow).

13.5.4 Using PalmDebugger to import and export files

PalmDebugger may also be used as a tool to import and export files between your device and your PC.

13.5.4.1 Import <cardNo> <filename>

Run the `Import <cardNo> <filename>` command in the Console window to copy a Palm OS database from the PalmDebugger's Device folder to the device. Use this command when you have built a new version of an application and want to load the application on to the device for testing.

The basic form of the import command is:

```
import <cardNo> <filename>
```

Where `<filename>` is the name of a file on your local system. By default, PalmDebugger looks for the named file in the *Device* sub-directory within the directory that contains the PalmDebugger application itself. The filename can also be specified using a relative or absolute path if it's not in the Device directory.

The `<cardNo>` is always 0, which means, "import the database into the built-in RAM on the device." For example:

```
import 0 HTMLLibTest.prc
```

13.5.4.2 Export <cardNo> <filename>

The export command does the opposite of the import command: it copies a database from the device to the desktop. This command is also run in the Console window.

The basic form of the export command is:

```
export <cardNo> <filename>
```

Where `<filename>` is the name of the PalmOS database.

The database will be copied to your local system in the standard PRC or PDB file format. These two file formats are actually identical. PRC is used to indicate resource databases, and PDB is used to represent record databases.

When it is copied, the database will be named `<filename>.prc` or `<filename>.pdb`. All exported files are placed into the Device sub-directory of PalmDebugger.

Unlike the import command, you may specify a `<cardNo>` other than 0.

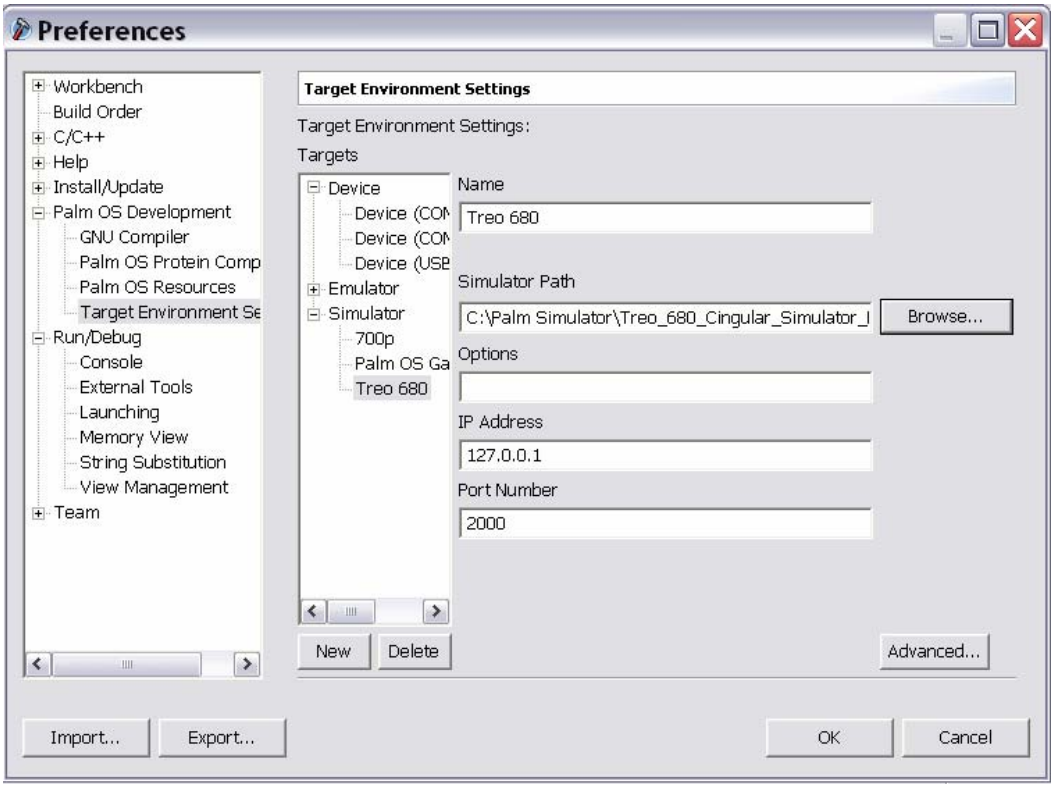
13.5.5 PalmDebugger tips

- If you cannot use breakpoints, close the PalmDebugger application and open it again, or reset your device. Your device will not respond to the keyboard or touch screen until you run your application again.
- PalmDebugger keyboard shortcuts are the same as those used with CodeWarrior.
- PalmDebugger heap dumps are useful for:
  - Finding memory leaks. A heap dump will display all allocated memory chunks.
  - Analyzing memory corruption. If the heap structure if corrupted, a heap dump will usually report it.
- Common causes of heap corruption include:
  - Buffer overruns and underruns. If you write to a memory location either before or after the memory chunk, you will corrupt the structure of the heap.
  - Memory chunks are sequential, so a buffer overrun corrupts the header for the next chunk. A buffer underrun corrupts the header of the current memory chunk.
  - Uninitiated pointers that write to a freed chunk of memory or that write to an unlocked, movable chunk will both cause heap corruption.
- PalmDebugger CANNOT be used for the following things:
  - PalmDebugger cannot be used with CodeWarrior builds while debugging.
  - PalmDebugger cannot be used for modifying variables.
  - PalmDebugger cannot step out to the next function call
  - The CallStack check is not enabled in 68k source-level debugging

For more information on PalmDebugger, see the ACCESS Developer Network at the following URL:

http://www.access-company.com/developers/documents/docs/devguide/UsingDebugger.html#995835

13.6 Garnet OS Developer Suite



The Garnet OS Developer Suite (also called the Palm OS Developer Suite, or PODS) by ACCESS is an integrated development environment that supports Simulator and on-device debugging. Download it from the ACCESS Developer Network at **http://www.access-company.com/developers/**.

The Garnet OS Developer Suite is useful for:

- 68k source code debugging.
- Visual UI design
- Checking variables, memory, registers and databases

To connect to the Palm Simulator, navigate to **Window > Preferences > Palm OS Development > Target Environment Setting > NEW**.

For more information on debugging with the Garnet OS Developer Suite, see the ACCESS Developer Network at **http://www.access-company.com/developers/**.

13.7 Resets

While developing and testing applications, there are often times when devices display strange behavior, or simply require a refresh. For this reason, it is important to understand the different kinds of resets that you can perform on Palm devices running Palm OS.

There are four types of resets: soft resets, warm resets, hard resets, and factory resets. Different situations call for different resets. This section will detail the difference between these four types of resets, explain when to use each one, and provide instructions on how to perform them.

13.7.1 Soft Reset

Performing a soft reset on a Palm device is similar to restarting a PC. It is essentially "rebooting" the system. A soft reset wipes the RAM (the dynamic heap and the DB Cache) clean.

A soft reset is useful in the following situations:

- If the device is not being responsive to button presses or screen taps.
- If you were using the debugger and are now finished.
- If you're seeing strange behavior that you think might be due to previous tests you've run on the device.

Basically, any situation where restarting your desktop computer might help can also be applied to your Palm device with a soft reset.

To perform a soft reset, remove the cover from the back of the device, remove the battery for a few seconds, then replace the battery. Certain Palm devices also have a reset pin, which you can press instead of removing and reinserting the battery with the same result.

13.7.2 Warm Reset

A warm reset, also known as a system reset, tells the system to restart without loading any system extensions.

A warm reset can be helpful in these situations:

- If you've already tried a soft reset and that did not resolve the issue. For example, if your device is stuck in a reset loop.
- If you're seeing a warning of low memory conditions, which suggests that you should delete some items and make some space.

To perform a warm reset, press and hold the Up button on the 5-way navigator on the device, then press the reset pin, or remove and reinsert the battery.

13.7.3 Hard Reset

The purpose of a hard reset is to restore the device to its default state prior to a customer's use. This means that any applications, data, settings, or preferences will be erased when a hard reset is performed. (Make sure to back up all data before hard resetting your device.)

A hard reset not only wipes the RAM clean like the soft reset, but it also erases the storage heap that is in the NAND flash. (For more information on NAND memory, refer to the NVFS API section of the document Palm Developer Guide, Palm OS Platform, which is part of the SDK download.)

For a developer, this reset can be a useful testing tool, as it allows you to test the behavior of your application on a "clean slate" by undoing any changes that development caused to the original state of the device.

The steps to performing a hard reset are different on different devices. For instructions, refer to the Palm Support Knowledge Library at the following URL:

http://kb.palm.com/SRVS/CGI-BIN/WEBCGI.EXE/,/?St=49,E=000000000924981633,K=2784,Sxi=8,Case=obj(42094)

13.7.4 Factory Reset

A factory reset reformats the device and is designed to remove all data so that it cannot be retrieved. This type of reset provides increased privacy of data in comparison to a hard reset, so if your device is changing hands, you should definitely perform a factory reset.

For instructions, see the following Palm Support Knowledge Library article:

http://kb.palm.com/SRVS/CGI-BIN/WEBCGI.EXE?New,Kb=PalmSupportKB,CASE=15574

This page intentionally blank

PART IV

Style Guide

This part of the guide provides guidelines for the "look and feel" of applications that use software components in the Palm OS® SDK.

This page intentionally blank

CHAPTER 14

14. Style Guide

This chapter outlines the style guidelines to follow when developing applications that work with the Blazer web browser.

14.1 Designing pages for the Blazer® web browser

Available on:

- LifeDrive™ mobile manager
- Centro™ and Treo™ smartphones
- Tungsten™ T5 and Tungsten™ E2 handhelds

Although the Blazer web browser's table unrolling technology does a good job of adapting websites for mobile devices, advance planning can reduce the translation and download time and ensure that results are as expected. This section covers the various factors that website designers and programmers should take into account when designing web pages.

14.1.1 General rules for web page design

Here are some general rules that apply when designing websites for mobile devices:

- Make content accessible within one or two links.

  Because the user is typically using a slow, costly connection, it is important that the information the user is trying to access be easily available within one or two navigational moves.

- Keep relevant content and links within the viewable area.
- Personalize and prefill forms whenever possible.
- Use as much screen space as possible without cluttering the screen.
- Keep graphics use to a minimum. When you have to use graphics, keep them simple and small.
- Keep the page size small, preferably under 4KB to 6KB.
- Keep the page simple. Limit the use of JavaScript, and don't use frames or plug-ins. While JavaScript is supported, most JavaScript runs slower on the Blazer web browser than on other handheld browsers due to memory and CPU limitations. For simple JavaScript this is not noticeable, but the use of more elaborate JavaScript may create noticeable delays.

Due to the various restrictions on designing web pages for mobile devices, a website should contain a parallel "mobile" version. The full website can take advantage of all the web technologies, while the mobile site is a slimmed- down version designed to support mobile devices. For example, the Palm website contains a page that is designed for mobile devices: **http://mobile.palm.com**.

Web developers can have their web server automatically load the appropriate page based on the user agent of the browser. For information about the user agent for the Blazer web browser, see **Section 14.1.5.2 on page 273**.

14.1.2 Screen resolution

Mobile devices have a screen size that is much smaller than a desktop screen's resolution. Specifications for Palm devices are as follows:

- On low resolution Treo 600 smartphones, the full-screen resolution is 160 x 160 pixels, and the available space in the browser, due to the toolbar and scroll bar dimensions, is 155 x 145 pixels.
- On high-resolution Treo 650 smartphones and later, the full screen resolution is 320 x 320 with 16-bit color that displays over 65,000 colors. The screen is 186 physical dots per inch (DPI) and has a horizontal and vertical pitch of 136.5 micrometers.
- On high-resolution Tungsten™ handhelds the resolution is 320 x 320, or 320 x 480, depending on the current state of the Graffiti area, and also includes with 16-bit color that displays over 65,000 colors.

Blazer 4.0 web browser also has a URL bar that users can choose to display or hide. When it is displayed, it takes space away from the current web page display area. Designers should take these dimensions into account when creating images, tables, lists, and other components of web pages. Developers who want to alter their page based on the current screen resolution should note that the current screen resolution is included in the User Agent. See **Section 14.1.5.2 on page 273** for more information.

14.1.3 Connection speed

Users can connect to the Internet at several speeds. The speeds available depend on the type of network the user is accessing and the device they are using to connect (CSD, 1xRTT, GPRS, EDGE, and EvDO). Field-tested speeds can range from 10kbps (CSD dial-up) to over 150kbps (EDGE). EvDO connection speed ranges from 400kbps to 600kbps (based on 1MB FTP file downloads). Website designers should keep in mind that different users can connect using a wide range of speeds and should optimize their web pages to load appropriately.

For information on detecting EvDO vs. 1xRTT connections, see **Section 8.4.1 on page 112**.

14.1.4 Content

In Optimized mode, the Blazer web browser typically reformats web pages into a small, screen-friendly format. Sites that have a simple layout with minimal tables and graphics have a good chance of being displayed properly after the reformatting. The sections details the factors to keep in mind when designing your web pages.

14.1.4.1 Page titles

The Blazer web browser does not display web page titles in web page view. To view the title of a web page, users must open the Page Properties dialog box, as shown in the following figure. The page title is used to prepopulate the bookmark description or title field when adding a bookmark.

14.1.4.2 Content optimized for the Blazer® web browser

You can also insert the following tag into the HEAD section of a web page to minimize the amount of reformatting that the Blazer web browser does:

```
<META name="HandheldFriendly" content="True">
```

This tag tells the Blazer web browser to render tables without any special reformatting. If this tag is not present, tables may be unrolled or reformatted. Also, if this tag is present, Blazer 4.0 displays the content in one pass instead of the usual two-pass rendering method.

14.1.4.3 Embedded audio playback

The Blazer 3.0 and 4.0 web browser supports embedded audio playback using the `<OBJECT>` tag. The `<BGSOUND>` and `<EMBED>` tags are not supported. For instance:

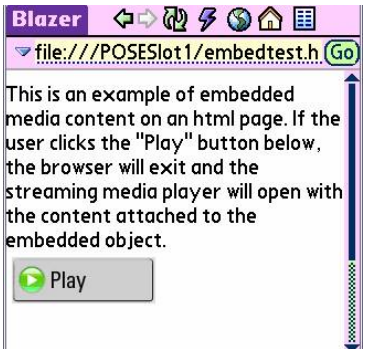
```
"<object data="oscar.mid" type="audio/midi" width="0"  height="0" ></object>"
```

loads and plays the MIDI file oscar.mid while the current page is displayed.

All devices with Blazer 3.0 or Blazer 4.0 loaded in ROM support embedded MIDI. Some devices may also support embedded AMR and/or embedded QCELP.

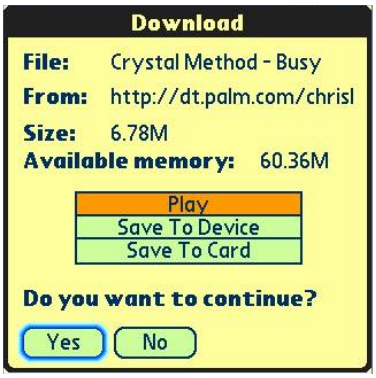
14.1.4.4 Streaming embedded content in Blazer® 4.5

The Blazer 4.5 web browser supports embedded content playback using the <EMBED> and <OBJECT> tags. To do so, Blazer 4.5 searches web pages for embedded objects. If it finds an <EMBED> or <OBJECT> tag on the page with a MIME-type that an application is registered for, then the browser will display a Play button, as shown:

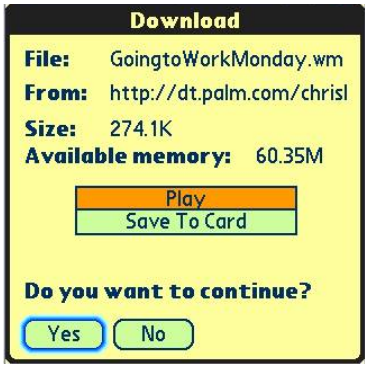


Links are then passed to the streaming application via the Exchange Manager when a user clicks a link.

Streaming content may also be presented as a direct link. If an application registered for streaming the type of content, the user will be presented with a dialog box that allows the user to choose whether to play the content, save it to the device, or save it to the card, as shown:



Once content is downloaded to the device, if there is no application registered to display the content, the user will see the following:



The following code is an example of an <EMBED> tag that the Blazer 4.5 browser will support:

```
<object id="MediaPlayer"

classid="CLSID:22d6f312-b0f6-11d0-94ab-0080c74c7e95"

codebase="http://activex.microsoft.com/activex/controls/mplayer/en/
nsmp2inf.cab#Version=6,4,5,715"

align="baseline" border="0"

type="application/x-oleobject">

<param name="FileName" value="http://mediaframe.yahoo.com/
buildlist.asp?p=movies&f=1808624572&=Sony%20Pictures%20Classics&type=t&id=1349284-
145632&m=wmv&r=700&l=SAV&b=e18qn9911a2ga428e61cf">



                  <!-- Embedded Microsoft Media Player Object for Netscape Navigator. -->

<embed src="http://mediaframe.yahoo.com/
buildlist.asp?p=movies&f=1808624572&=Sony%20Pictures%20Classics&type=t&id=1349284-
145632&m=wmv&r=700&l=SAV&b=e18qn9911a2ga428e61cf"

                        align="baseline" border="0"

                        width="320" height="298"

                        type="application/x-mplayer2"

            pluginspage="http://www.microsoft.com/isapi/
redir.dll?prd=windows&amp;sbp=mediaplayer&amp;ar=media&amp;sba=plugin&amp;"

                        name="MediaPlayer">

                         </embed>

</object>
```

14.1.4.5 Link protocol types

You can use RTSP or HTTP protocol links in an application to link to streaming content:

- **RTSP -** Using RTSP links, an application will redirect the user immediately to the registered streaming content. The user will not be presented with a dialog or a play button.
- **HTTP -** Using HTTP links, if an application is registered for streaming the type of content to be downloaded, an application will present a dialog box that allows the user to play the content, save it to the device, or save it to the card. If no streaming application is registered to handle the MIME-type or file extension, an application will present only the save option.

Important notes on Blazer® 4.5

- The Blazer 4.5 browser does not support a file location being sent via JavaScript post rendering.
- The Blazer 4.5 browser will not display any customized playback buttons. It will use its own Play button, shown previously.

14.1.4.6 File upload

Blazer 4.0 and later has limited file upload support. Blazer only supports uploading files from an expansion card; it does not currently support any uploading from Palm OS main memory. When Blazer is rendering a page with a form containing the file upload tag, Blazer displays a single-line text field. To upload a file, the user must type in the fully-qualified path to the file on the expansion card to be uploaded. (In the future, a Browse button may be added that allows the user to browse for the file to be uploaded.)

On the LifeDrive™ mobile manager, a user can upload files from Palm OS main memory. On devices with the file browser program and Blazer 4.1 and later, the single text line is replaced with a browser button that allows a user to utilize the file browser program to select a file to upload from anywhere the file browser program can explore. File uploads have succeeded up to about 1MB; in general, a 500K upload will succeed.

14.1.4.7 mailto command

The Blazer web browser does support the `mailto` command to send email.

Typically on desktop systems, the web browser launches an email application to send mail when a `mailto` command is encountered. On devices, it is not guaranteed that a mail application is installed. If an email application is installed, a `mailto` command launches the default email application as defined in the Defaults panel in Prefs.

For dialing telephone numbers, see the `tel:` and `phoneto:` tag discussions in the **Section 14.1.5.1 on page 273** section.

14.1.4.8 Multipass rendering

Unless the `HandheldFriendly` tag is present, Blazer 4.0 and later renders each page in two passes. In the first pass, all the text on the page is displayed. In the second pass, images are downloaded and JavaScript is downloaded and executed. By making the text immediately available to the user, the site appears more responsive and the user can read content and select links before the entire page has downloaded. There are several things a content developer can do to ensure that a page is more usable during the first pass:

- Minimize the use of images, especially images with text.

  Because images are not displayed until the second pass, if a site's content is image-based the user has to wait longer before interacting with the page.

- When images are used, include descriptive ALT tags.

  During the first pass the text of the ALT tag is displayed in place of the image.

- Do not use JavaScript in links.

  Usually, when a user selects a link during the first pass, the link is immediately activated. If the link includes JavaScript (and JavaScript is enabled), the browser cannot follow the link until the second pass because the JavaScript engine is not available until the second pass is complete. Thus, if the user clicks on a link with JavaScript during the first pass, an error message is displayed and the link is not activated.

14.1.4.9 Forms

The Blazer web browser supports standard HTML and WML forms, including text boxes, radio buttons, check boxes, text inputs, select lists, multiple selects, and drop-down menus. However, there are some guidelines to follow when you design forms for mobile devices:

- Make sure that the form's open and close tags (`<form>` and `</form>`) are not contained inside a table. Form input may safely be placed inside table cells.
- Text input is supported, but the maximum length of the text is the length of the text input dialog box.

14.1.4.10 Tables

In Optimized mode, the Blazer web browser supports tables by reformatting or unrolling the table so that it fits on the small screen of a mobile device. The table-rendering engine is optimized for displaying simple text-based tables. Web page designers should avoid using one-pixel spacer images for precise content control, because the images and subsequent reformatting may not look ideal on a mobile device.

Table width attributes are not supported. However, table cell width attributes are supported. To widen tables, change the width attribute of the table cells. To preserve the width on the client display, use the `HandheldFriendly` tag. For more information, see **Section 14.1.4.2 on page 263**.

For example, the following HTML code displays a table with a table width of 140:

```
<table border="1">
<tr>
    <td width="140" colspan=5>abcdef</td>
</tr>
<tr>
    <td>1</td>
    <td>2</td>
    <td>3</td>
    <td>4</td>
    <td>5</td>
</tr>
</table>
```

The table appears on a desktop web browser as follows:

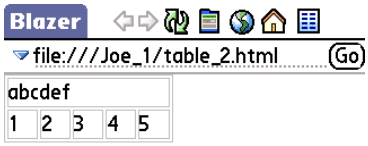
a b c d e f				
1	2	3	4	5

This same HTML code is displayed in the Blazer web browser without the extra spacing if the `HandheldFriendly` tag is not included:

To have the Blazer web browser display the table as intended, include the `HandheldFriendly` tag within the HEAD section of the web page, as follows:

```
<META name="HandheldFriendly" content="True">
```

This causes the device to display the table with the width tag enabled, as shown in the following figure:



14.1.4.11 Images

Using images on a website can enhance the presentation of content and the user experience. At the same time, the use of images slows down the web browsing experience. Therefore, it's important to carefully consider which images to display on mobile devices.

All images pass through an imaging processor in the Blazer web browser client that scales down any image that exceeds the current screen size. The bit-depth of the image is also adjusted to match the device's image display settings.

When designing images for mobile devices, keep the following in mind:

- Image size should be very small (under 4KB).
- Use only a few graphics per page to reduce the load time.
- Use images that are high contrast for easy readability.
- Use ALT tags to display text content while images download.

  Here is what is displayed while the image is loading:

  And here is what appears after the image has finished loading:

14.1.4.11.1 Images with text

Avoid using images in place of text. Unnecessary images add to the total download time for each page. If the Blazer web browser needs to scale the image, the text may become unreadable. The following images show how a text image can look fine on a desktop browser but can become unreadable when scaled and displayed in the Blazer web browser.

14.1.4.11.2 Horizontal header images

Many websites use a horizontal header image for navigation purposes. Typically, this image consists of many smaller images that are formatted in a table with zero- length borders.

On a desktop browser, such an image is displayed appropriately. The Blazer web browser, however, typically reformats such a page so that the images are stacked vertically. The following images demonstrate this behavior.

Header images on a desktop browser:

The same images on the Blazer web browser:

You can clean up the web page by removing spacer images and inserting the `HandheldFriendly` tag at the beginning of the page. The next figure shows the updated page without spacer images in the table. The second figure shows the page in the Blazer web browser without the `HandheldFriendly` tag. Because the spacer images are not present, the browser displays the image in a more readable format. The third figure shows the same page with the `HandheldFriendly` tag present. This tag instructs the Blazer web browser not to reformat the tables. As a result, the table is displayed as intended. The Blazer web browser provides horizontal scroll bars to allow the user to view the entire image.

14.1.4.11.3 Supported image formats

The Blazer 3.0 web browser supports the following image formats:

- GIF
- JPEG
- PNG
- Animated GIF
- WBMP
- BMP (Blazer 4.0 web browser)
- PJPEG

14.1.4.12 Unsupported content

While the Blazer web browser supports most of the HTML 3.2 specification, there are certain unsupported elements. Some of the additional web technologies that are not supported in the Blazer web browser include:

- Java applets
- WMLScript
- Animations (Macromedia Flash)
- Audio, although the Blazer web browser can download an audio file to be played by an application that can handle the audio format
- Browser Plug-Ins

When the Blazer web browser encounters an unsupported element, it ignores the associated code when displaying the web page. In most cases, the user experiences the web page with reduced functionality. If the website requires the unsupported technologies to view the information, the Blazer web browser users will not be able to use the website.

14.1.5 Working with the Blazer® web browser

This section discusses topics that are specific to the Blazer web browser and Palm OS® software. You should make note of these issues when implementing web pages that you have optimized for use with the Blazer web browser.

14.1.5.1 Palm OS™ software integration tags

In general, all HTML files must begin with the `<html>` tag or the Blazer web browser may not recognize them as HTML.

Blazer Web Browser 3.0 and 4.0 do not support date picker and time picker. Support may be added in a later release of the Blazer web browser.

14.1.5.2 Browser identification with user agent string

As part of communication to a web server, web browsers send out a string indicating what type of browser is accessing the server. This is referred to as the user agent string.

The Blazer 3.0 web browser user agent string is:

```
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows 95; PalmSource;
Blazer 3.0) 16;160x160
```

The Blazer 4.0 web browser user agent string is:

```
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows 98; PalmSource/
hspr-H102; Blazer/4.0) 16;320x320
```

The Blazer 4.5 web browser user agent string is:

```
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows 98; PalmSource/
Palm-D052; Blazer/4.5) 16;320x320
```

In the user agent strings above, `hspr-H102` for Blazer 4.0 and `Palm-D052` for Blazer 4.5 respectively indicates which Palm device the browser is running on; different devices return different strings.

Also, the current screen size is included, so if the user changes the state of the Graffiti area, the user agent string is updated to match the screen size. For these reasons, you should not search for an exact match when checking for the presence of the Blazer 4.0 web browser. Instead, simply check to see if the user agent string contains the substring `Blazer`.

Web developers can use this header to send the version of a web page optimized for display by the Blazer web browser.

14.1.5.3 Cookies

The Blazer web browser contains support for cookies. Typically, website developers use cookies to store a small amount of information about a user's ID or profile, website personalization, and so forth. As with any web browser, users can have the Blazer web browser not accept cookies or clear out the cookie store. Cookies are stored on the local Palm OS device.

The Blazer web browser contains full support for the `SET-COOKIE` header. The only restriction is that the value attribute must not exceed 2000 bytes. Cookies with a value attribute larger than 2000 bytes may be rejected. Other types of cookie headers are not supported.

If a cookie has an expiration date, then it is a persistent cookie. If there is no expiration date, it is a session cookie. Session cookies are kept post-session so that a user can get back into their previous session as long as it is within the time limit described below.

14.1.5.4 Session handling

On a desktop browser, a session cookie persists until the user quits the web browser. On a Palm OS device, only one application can be active at any given time. To take into account that a user may temporarily switch to another application while using the Blazer web browser, the system ends a session 20 minutes after the Blazer web browser exits. (This time may vary based on the wireless service provider.)

14.1.5.5 Security

The Blazer web browser uses the standard Palm OS security libraries—the same libraries that are shipped with every version of Palm OS 5.2.1 and later. The Blazer web browser incorporates 128-bit SSL 3.0 encryption technology to ensure that visiting any site with a device is as secure as browsing from a desktop. The 128-bit encryption uses the RC4 algorithm and RSA-based key exchange. It also incorporates RSA-based digital signature verification, MD5 and SHA-1 secure hash algorithms, support for X.509 certificates, and an intuitive indication of a secure connection by a lock icon in the toolbar or URL bar.

14.1.5.6 Caching

The Blazer web browser includes a web page cache to improve performance. If a page requires a regular refresh, or should not be cached, the web server must send the appropriate standard HTTP cache control headers in the HTTP response.

14.1.5.7 Downloads

The Blazer 4.0 web browser limits downloads to 2MB. On the LifeDrive™ mobile manager with the Blazer web browser 4.1 and later, much larger downloads are possible. If an item is 2MB or less and there is sufficient DBCache, a standard download occurs. If an item is larger than 2MB or there is insufficient DBCache, the user sees a dialog stating that Blazer is writing to the internal drive. The Blazer web browser writes a VFS file directly to disk rather than using the Exchange Manager.

14.1.6 Testing your website

Designing your web pages will most likely be an iterative process. Test your websites early and frequently to ensure that you are able to achieve the intended design and layout. The following sections contain a few pointers to help you successfully test your website's content.

14.1.6.1 Multiple devices

If you intend your content to be accessible through a variety of devices, be sure to test your website with as many of those devices as possible during the development process. Areas you should look at in particular are device memory and screen size.

14.1.6.2 Refreshing content

Because the Blazer web browser uses cached pages on a device to provide quick access to frequently viewed content, constant updates to a web page during development may not be visible on the device. Set the cache size to zero during testing. During development, you should use the Clear button in Preferences to clear out the cache and use the Page -> Refresh menu command to make sure the current page is loaded. Images are typically refreshed when the page is refreshed. If images are not being updated in the Blazer web browser, the cache should be cleared out or the image file name should be changed.

14.1.7 International support

The Blazer web browser client supports the Palm OS character set. This character set is similar to the ISO-8859-1 character set. The Palm OS character set allows support for most Western European characters. The Blazer web browser application has also been localized into the following European languages:

- English
- French
- Italian
- German
- Spanish
- Brazilian Portuguese

14.1.7.1 HTML encoding

The Blazer web browser supports web pages that are encoded using the ISO-8859-1 and UTF-8 character formats. This allows Western European languages to be displayed properly on the Blazer web browser client. All HTML files must begin with the `<html>` tag or they may not be recognized as HTML.

Web pages that are encoded in UTF-8 must indicate so by including one of the following tags:

Content-type HTTP Header:

```
Content-Type: text/html; charset=utf-8
```

HTTP-equiv META Tag:

```
<META http-equiv="content-type" content="text/html; charset=utf-8"\
```

14.1.7.2 WML encoding

WML content is typically encoded in UTF-8 format. WML pages that are encoded in the ISO-8859-1 format must indicate so by including one of the following tags:

Content-type HTTP Header:

```
Content-Type: text/vnd.wap.wml; charset=iso-8859-1
```

HTTP-equiv META Tag:

```
<META http-equiv="content-type" content="text/vnd.wap.wml; charset=iso-
8859-1"\
```

XML Tag:

```
<?xml version="1.0" encoding="iso-8859-1"\
```

14.1.7.3 Accept headers

The Blazer web browser client supports Accept HTTP headers. The client sends this header to the web server based on the languages the client supports. You can use this header to determine what type of content to deliver to the device.

The Blazer Web Browser 3.0 Accept headers for an English ROM appear as follows:

```
Accept: text/html, application/vnd.wap.xhtml+xml, application/xhtml+xml;
profile="http://www.wapforum.org/xhtml", image/gif, image/jpeg, image/
pjpeg, */*
Accept-Language: en, *;q=0.8
```

For Blazer 4.0 and later, the Accept header is similar to the Blazer 3.0 header, except that it also includes every type currently registered with the Palm OS Exchange Manager.

(The Accept-Language header varies based on the current language selected upon setup of the device.)

```
Accept-Encoding: deflate, gzip
```

14.1.8 List of acronyms

The following table lists web browser and Internet acronyms.

Acronym	Definition
cHTML	Compact HTML. A subset of HTML for mobile devices. Primarily used in i-Mode devices.
ECMA	European Computer Manufacturers Association.
GIF	Graphics Interchange Format.
HDML	Handheld Device Markup Language. An old format used for web-enabled phones. This is no longer used.
HTML	Hypertext Markup Language.
HTTP	Hypertext Transfer Protocol.
ISO	International Organization for Standards. The name is derived from the Greek word *iso*, meaning “equal.”

Acronym	Definition
JPEG	Joint Photographic Experts Group.
PNG	Portable Network Graphics.
TCP/IP	Transmission Control Protocol/Internet Protocol.
UCS	Universal Character Set.
UTF	UCS Transformation Format.
WAP	Wireless Application Protocol.
WBMP	Wireless Bitmap. A graphic format optimized for wireless devices.
WML	Wireless Markup Language.
XHTML	Extensible HTML. A cross between HTML and XML.
XML	Extensible Markup Language.

14.1.9 Palm OS® integration tags

In general, all HTML files must begin with the `<html>` tag or the Blazer web browser may not recognize them as HTML.

The Blazer web browser supports several tag attributes that extend HTML support for Palm OS devices, as shown in the following table.

Keyword	Description
HandheldFriendly	This attribute for the META tag tells the proxy server that the web page has been specifically designed for small screens. The proxy server tries to render the tables as close to specification as possible. Example: `<META name="HandheldFriendly" content="True">`
Palm	This keyword is used in the HREF attribute to launch a specific application on the device. Blazer web browser exits and the specified application becomes the active application. Example: `<A HREF="palm:memo.appl">Memo Pad</A>`
Palmcall	This keyword is similar to the Palm keyword except that Blazer web browser sublaunches the specified application. Once the application exits, the user returns to Blazer web browser. For example: `<A HREF="palmcall:flip.appl">Flipper</A>` Parameters can also be passed into the application using the `sysAppLaunchCmdURLParams` launch code. For more information on this and the `palm` and `palmcall` tags, refer to the following tutorial on the Palm OS developer website: **www.palmos.com/dev/tech/webclipping/tutorials/tutorial_palm.html**

Keyword	Description
phoneto: and tel:	These keywords are equivalent and are used in the HREF attribute to dial a specific phone number. Blazer web browser exits and the phone application becomes the active application when these keywords are used. Examples: `<A HREF="tel:555-1212">Jenny</A> and <A HREF="phoneto:588-2300">Empire</A>`

file:	This keyword is used in the `HREF` attribute, in the URL bar, and in the Open URL dialog box to access browser content stored locally on an expansion card. A network connection is not required when accessing content using `file:///`.
	Note that the syntax always includes three slashes after `file:`
	Blazer 3.0 web browser supports the following syntax:
	`file://dir1/dir2/file.txt`
	In this example, Blazer 3.0 web browser would attempt to open the file file.txt located on the expansion card in directory /dir1/dir2.
	In addition to the Blazer 3.0 syntax, Blazer 4.0 web browser includes support for an additional syntax that handles devices with multiple expansion cards better, such as the Tungsten T5 handheld.
	An example of the additional syntax is:
	`file:///volname/dir1/dir2/file.txt`
	In this example, Blazer 4.0 would locate the expansion card named volname, and then open file file.txt in directory /dir1/dir2 on that expansion card.

14.2 Gadgets

In the Phone application on Palm smartphones, there is a fixed position for the system gadgets. For third-party applications to be consistent with the Phone application, they should position the system gadgets in the same location. We have included instructions for placement of the gadgets based on the Phone application.

14.2.1 Required headers and libraries

The `PmSysGadgetLibrary` is required to use the gadgets. This library contains the system Battery, Signal, and Bluetooth® wireless technology gadget implementation. It includes the following header files:

- `68K/Libraries/PmSysGadgetLib/PmSysGadgetLib.h`
- `Common/Libraries/PmSysGadgetLib/PmSysGadgetLibCommon.h`

The header file `PmSysgadgetlibCommon.h` has changed for Centro smartphones. The enumerated type `PmSysGadgetBtStatusTypeEnum` includes the following item, which indicates Bluetooth radio status:

```
kIndicatorBtConnectStereo       /**< Connected stereo **/
```

14.2.2 Overlapping gadgets in Treo™ 680

In Treo 680, support has been added to `PmSysGadgetLibrary` for overlapping gadgets. If a gadget is defined with the same boundaries as an existing gadget, it will be given a lower priority than the previously defined gadget, and will only be displayed if the other gadget is not visible.

14.2.3 How to include the Battery gadget

To include the Palm smartphone Battery gadget in an application form:

1. Add a gadget to the application form at position (304, 0) for 320 x 320 resolution for a Treo 650, Treo 680, Treo 700p, Treo 755p, or Centro smartphones. Use half these dimensions for a Treo 600 smartphone.

   Because the system determines the height and width of the gadget, the width and height specified in the rcp/rsrc file will not matter.

2. In the application form event handler for `frmOpenEvent`, add the following code:

```
PmSysGadgetStatusGadgetTypeSet ( gPmSysGadgetLibRefNum
                                 frmP,
                                 <YOUR_BATTERY_GADGET_ID>,
                                 pmSysGadgetStatusGadgetBattery);
```

Updates and events associated with the Battery gadget are handled automatically.

14.2.4 How to include the Signal gadget

To include the Signal gadget in an application form:

1. Add a gadget to the application form at position (278, 0) for 320 x 320 resolution for a Treo 650, Treo 680, Treo 700p,Treo 755p, or Centro smartphones. Use half these dimensions for a Treo 600 smartphone. Because the system determines the height and width of the gadget, the width and height specified in the rcp/rsrc file will not matter.
2. In the application form event handler for `frmOpenEvent`, add the following code:

```
PmSysGadgetStatusGadgetTypeSet ( gPmSysGadgetLibRefNum
                                   frmP,
                                   <YOUR_SIGNAL_GADGET_ID>,
                                   pmSysGadgetStatusGadgetSignal);
```

Updates and events associated with the Signal gadget are handled automatically.

14.2.5 How to include the Bluetooth® wireless technology gadget

NOTE: The Treo 600 smartphone does not include Bluetooth wireless technology.

To include the Bluetooth wireless technology gadget in an application form:

1. Add a gadget to the application form at position (260, 0) for 320 x 320 resolution for a Treo 650, Treo 680, Treo 700p, or Treo 755p smartphone. Use half these dimensions for a Treo 600 smartphone. Because the system determines the height and width of the gadget, the width and height specified in the rcp/rsrc file will not matter.
2. In the application form event handler for `frmOpenEvent`, add the following code:

```
PmSysGadgetStatusGadgetTypeSet ( gPmSysGadgetLibRefNum
                                   frmP,
                                   <YOUR_BLUETOOTH_GADGET_ID>,
                                   pmSysGadgetStatusGadgetBt);
```

Updates and events associated with the Bluetooth gadget are handled automatically.

PART V

Hardware Developers Kit

This part of the guide provides details on how to develop hardware peripherals for Treo™ 650 smartphones and later, as well as for Tungsten™ T5 and Tungsten™ E2 handhelds.

This page intentionally blank

CHAPTER 15

15. Multi-connector Specifications

This chapter defines the interfaces and interactions of the Palm® expansion Multi-connector, also known as the Athena connector, and its surrounding circuits and controlling software.

15.1 Overview

Available on:

- Centro™ and Treo™ smartphones
- Tungsten™ T5 and Tungsten™ E2 handhelds and LifeDrive™ mobile managers

This chapter specifies the electrical and software interface characteristics of Palm's Multi-connector. These characteristics include sets of various charging, cradle, and cable configurations, but from a handheld or smartphone perspective the electrical specification and systems software requirements are consistent across multiple devices. Specific devices may not, however, implement all of the features of the Multi-connector.

The Multi-connector supports the following features:

- Charging power from an external adapter with adapter detection ability
- Universal Serial Bus (USB)
- Serial communications (no flow control, logic levels)
- Dedicated HotSync® technology interrupt
- Power out
- Stereo headphone-level output
- Peripheral detection

The Multi-connector interface supports interaction with the following devices:

- USB HotSync® cables and cradles
- Other serial devices

  Due to serial peripheral detachment detection requirements, a device cannot be created that employs automatic serial peripheral detachment detection *and* utilizes the USB VBUS line for USB charging, signaling, or connections. Any device requiring both serial peripheral detection and USB functionality cannot automatically detect detachment of the peripheral through the normal serial peripheral detachment mechanism. Detachment for this situation must be handled through a nonstandard mechanism, such as serial error handling. For more information on the peripheral detection mechanism, see **Section 15.4 on page 292**.
- Pass-through peripherals

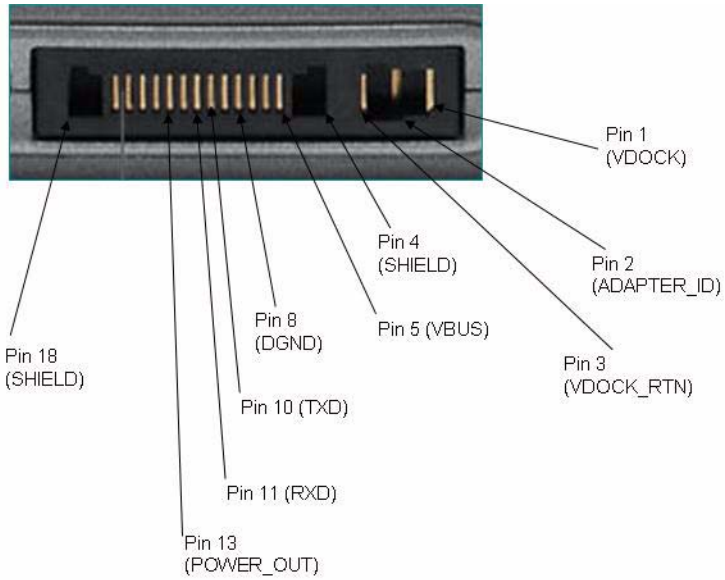
  These include peripherals that connect to a handheld or smartphone and have another connector to allow the peripheral to be connected to a USB HotSync cable or cradle.

15.2 Pinout of the Multi-connector

The pinout, with or without mounts, is described in the following table. All pin references that follow in this chapter refer to the device connector pin-numbering scheme shown in this table.

Pin # on device/ Multi-connector	Pin # on charger/ adapter connector	Pin# on data/ cable connector	Name	Direction with respect to the device	Default state with no attachment	Function
1	1		VDOCK	Power	CHRG_IN	DC charging voltage, 5V
2	2		ADAPTER_ID	Input	VCC, weak pull-up	Adapter identification
3	3		VDOCK_RTN	Power	GND	DC charging return
4	-	1	SHIELD	Shield	GND	Cable shield
5	-	2	VBUS	Power	VBUS_IN	USB charging voltage, 5V typical, 500 mA max
6	-	3	USB_DP	Input/output	Floating	USB Data +
7	-	4	USB_DN	Input/output	Floating	USB Data -
8	-	5	DGND	Power	GND	Digital ground, and VBUS return
9	-	6	Reserved	NA	NA	Do not connect

Pin # on device/ Multi-connector	Pin # on charger/ adapter connector	Pin# on data/ cable connector	Name	Direction with respect to the device	Default state with no attachment	Function
10	-	7	TXD	Input/output	VCC, weak pull-up	Transmit data, 3.3V logic level
11	-	8	RXD	Input	VCC, weak pull-up	Receive data, 3.3V logic level
12	-	9	HOTSYNC	Input	VCC, weak pull-up	HotSync input, active low, pulled up on device
13	-	10	POWER_OUT	Output	High impedance	Power output to external devices
14	-	11	SPKR_L	Analog output	AC coupled	Speaker output left
15	-	12	SPKR_R	Analog output	AC coupled	Speaker output right
16	-	13	AGND	Power	GND	Analog ground
17	-	14	MIC_IN	Analog input	DC coupled	Microphone input
18	-	15	SHIELD	Shield	GND	Cable shield



15.2.0.1 Shielding

On all peripheral devices, the shield pins 4 and 18 should be grounded with any available shielding system ground. For example, on a USB cable these pins should be connected to the USB cable outer shield, which in turn connects to the shield on the USB connector at the other end of the cable.

Where no external shielded ground is available peripherals should connect pins 4 and 18 to the peripheral's system ground.

15.2.1 USB

Pins 5, 6, 7, and 8 constitute the USB VBUS, D+, D-, and GND pins respectively.

The handheld is designed to accept the following parameters on pins 5, 6, 7, and 8:

Name	Description	Minimum	Average	Maximum	Units
(V)USB_VBUS_CHG	Input charging voltage	4.375	5.0	5.25	V
(V)USB_VBUS_CHG	Input serial peripheral detection voltage	2.97	3.3	3.63	V
(I)USB_VBUS_L	Input charging current, no negotiation, sunk from VBUS	-	-	100	mA
(I)IUSB_VBUS_H	Input charging current, with negotiation, sunk from VBUS	-	-	500	mA

15.2.2 Serial interface hardware

Pins 10 and 11 provide 3.3V logic-level serial connections with no dedicated hardware flow control pins. The direction of these pins with respect to the device is as follows:

- Pin 10 transmits from the handheld
- Pin 11 receives into the handheld.

The serial port connected to pins 10 and 11 supports the following bit rates and configuration options:

- 1,200 baud
- 2,400 baud
- 4,800 baud
- 9,600 baud
- 14,400 baud
- 19,200 baud
- 28,800 baud
- 38,400 baud
- 57,600 baud
- 115,200 baud
- 7 data bits

- 8 data bits
- No stop bits
- 1 stop bit
- Parity bit
- No parity bit

Both pins 10 and 11 are pulled high within the device by weak pull-ups. For more information on how these weakly pulled, high input characteristics are used in the peripheral detection mechanism, see **Section 15.4 on page 292**.

Pins 10 and 11 operate at 3.3V nominal voltage levels.

Treo 650 smartphones, Tungsten T5 handhelds and E2 handhelds, and LifeDrive mobile managers are designed to accept the following parameters on pins 10 and 11:

Name	Description	Minimum	Average	Maximum	Units
(V)RXTX_INL	Input logic low voltage*	0	-	0.594	V
(V)RXTX_INH	Input logic high voltage*	2.904	-	3.63	V
(V)TX_OUTL	Output logic low voltage*	0	-	0.3	V
(V)TX_OUTH	Output logic high voltage*	2.67	-	3.63	V
(V)TX_OC	Open circuit TX line voltage*	-	3.3	-	V
(V)RX_OC	Open circuit RX line voltage*	-	3.3	-	V

*With respect to digital ground, pin 8

15.2.3 Serial interface software

The Multi-connector serial pins interface with the Palm OS® software as a virtual serial port.

15.2.4 HotSync® interrupt hardware

Pin 12 provides a HotSync Interrupt pin. The HotSync interrupt is weakly pulled high inside the device.

A HotSync interrupt is initiated when pin 12 is pulled to GND.

The HotSync interrupt is not used in the peripheral detection mechanism; it only initiates a HotSync operation.

The presence of the pullup on the HotSync pin can be used by a peripheral as a method to detect when a device is attached to the peripheral.

Treo 650 smartphones, Tungsten T5 handhelds and E2 handhelds, and LifeDrive mobile managers are designed to accept the following parameters on pin 12:

Name	Description	Minimum	Average	Maximum	Units
(V)HS_INL	Input logic low-voltage* triggering interrupt	0	-	0.594	V
(V)HS_OC	Open circuit line voltage*	-	3.3	-	V

*With respect to digital ground, pin 8

15.2.5 HotSync® interrupt software

The HotSync interrupt is always set as an input. The default interrupt detection occurs on a falling edge.

15.2.6 Power output

Pin 13 provides a power output to power an external peripheral. This power output is limited to low-current capability only. The power output is normally driven LOW or floated as a high impedance signal to minimize the chances of a short to GND damaging the device.

Treo 650 smartphones, Tungsten T5 handhelds and E2 handhelds, and LifeDrive mobile managers are designed to accept the following parameters on pin 13:

Name	Description	Minimum	Average	Maximum	Units
(V)POUT_L	Output inactive voltage*	0	-	0.363	V
(V)POUT_H	Output active voltage*	2.97	3.3	-	V
(I)POUT	Output current supplied	30	-	-	mA
(R)POUT	Minimum load series resistance to GND*	89	-	-	Ohms
(C)POUT	Maximum load capacitance	-	-	4.7	µF

*With respect to digital ground, pin 8

15.2.7 Audio detection

Tungsten T5 handhelds and E2 handhelds, and LifeDrive mobile managers automatically detect and switch audio if they detect an attached audio peripheral and the audio peripheral indicates that a headset is inserted into the handheld.

Tungsten T5 handhelds and E2 handhelds, and LifeDrive mobile managers should not, however, have a headset inserted into the headset jack and be attached to an audio peripheral at the same time. The audio signal would be shared between the two, resulting in a loss of volume on the headset and the Multi-connector audio output channels.

15.2.8 Audio output

Pins 14 and 15 provide headphone-level stereo audio output. This output is intended to drive an external audio output device (such as a “boom box”), or to interface with a car kit.

Devices typically connect pins 14 and 15 directly to their headset stereo output signals.

15.3 Peripheral requirements

For information on the peripheral detection mechanism that drives many of the requirements listed in this section, see **Section 15.4 on page 292**.

Peripherals are required to conform to the following specifications to ensure successful operation with Treo 650 smartphones, Tungsten T5 handhelds and E2 handhelds, and LifeDrive mobile managers:

Name	Description	Minimum	Average	Maximum	Units
(C)PHL_LOAD	Capacitive load on POWER OUT pin	-	-	4.7	µF
(R)PHL_LOAD	Minimum load series resistance on POWER OUT pin	121	-	-	Ohms
(V)PHL_RXTX_INL	Serial line input logic low voltage*	0	-	0.594	V
(V)PHL_RXTX_INH	Serial line input logic high voltage*	2.904	-	3.63	V
(V)PHL_RX_OUTL	Serial handheld receive line output logic low voltage*	0	-	0.3	V
(V)PHL_RX_OUTH	Serial handheld receive line output logic high voltage*	2.67	-	3.63	V
(R)PHL_TX_DOWN	Pull-down to GND on TX*	1K	-	10K	Ohms
(R)PHL_RX_DOWN	Pull-down to GND on RX*	1K	-	10K	Ohms

*Pull-downs required for peripheral detection mechanism. May not be required on either or both of the RX and TX lines.

15.3.1 Audio peripherals

Audio peripherals should conform to the following requirements:

Name	Description	Minimum	Average	Maximum	Units
(R)PHL_A_DET	Maximum series resistance to ground on TX line for Audio Peripheral detection	-	-	10K	Ohms
(R)PHL_A_DET_HS	Maximum series resistance to ground on RX line for Audio Peripheral Headset Jack Insertion detection	-	-	10K	Ohms
(R)PHL_A_DET_NHS	Minimum series resistance to ground on RX line for Audio Peripheral Headset Jack Insertion Absence detection	10M	-	No maximum	Ohms
(R)AOUT_MIN	Minimum series resistance to ground on pins 14 and 15	8	-	-	Ohms

15.3.2 General serial peripherals

Serial peripherals should conform to the following requirements.

Name	Description	Minimum	Average	Maximum	Units
(T)PHL_SER_DLY == (T)DET_RXTX_DLY	Delay from POWER OUT applied to peripheral driving into Rx line	450	-	-	mS
RPHL_SER_DET	Series resistance to ground on TX and RX line before POWER_OUT applied for Serial Peripheral detection	6.8K	-	-	Ohms
VPHL_SER_DET	Minimum voltage on TX and RX line after POWER_OUT applied for Serial Peripheral detection	2.904	-	POWER_OUT	V
RPHL_SER_NDET	Maximum series resistance between VBUS and POWER_OUT for Serial Peripheral Detachment detection	-	-	10	Ohms

15.4 Peripheral detection

The Multi-connector interface provides class-level peripheral detection. The class-level detection mechanism is performed using hardware detection with no software requirements on the peripheral.

Peripheral detection is initiated by triggering either the serial TX or serial RX lines in their GPIO states as falling edge interrupts. After detection of an interrupt, software in the device debounces the interrupted line for at least (T)DET_DBC milliseconds. If the condition causing the interrupt on the TX or RX lines still exists after (T)DET_DBC milliseconds, class-level detection initiates.

15.4.1 Class-level detection

Class-level detection involves sampling the serial TX and RX lines as GPIO inputs both before and after the application of POWER_OUT. This provides four bits to define the attached peripheral, resulting in 12 possible attached configurations. (The situation in which both RX and TX stay high after attachment of the peripheral is invalid as no attachment interrupt is detected, thus invalidating four possible combinations.)

15.4.1.1 Peripheral attachment

Before attachment of any peripheral, the TX and RX lines are configured as GPIO inputs and have weak pull-ups attached to each line.

When a peripheral is attached, at least one of the TX or RX lines must by definition be low. After debouncing the signal, the device samples the RX and TX lines and applies power to the peripheral through the POWER_OUT signal. The device then waits (T)DET_PWR_DLY milliseconds to allow the peripheral to power up, and then samples the TX and RX lines again.

If you want a peripheral to identify itself, it must have strong pull-downs on the appropriate serial communications lines. Some serial peripherals may not require such strong pull-downs because the impedance to ground of the unpowered serial input pins may provide small enough resistance to GND to allow detection to operate effectively. Also, the peripheral must not power the RX line until (T)DET_RXTX_DLY milliseconds after power is applied.

When a Tungsten T5 or Tungsten E2 handheld or LifeDrive mobile manager detects that an audio peripheral is attached, the device automatically switches audio from the internal device speaker to the audio of the peripheral. As noted earlier, Tungsten T5 or Tungsten E2 handhelds and LifeDrive mobile managers should not have a headset inserted into the headset jack and be attached to an audio cradle at the same time. The audio signal would be shared between the two, resulting in a loss of volume on the headset and the Multi-connector audio output channels.

The truth table for class-level detection is as follows:

Before attachment		Peripheral attached, no POWER_OUT		Peripheral attached, POWER_OUT applied		Class
TX	**RX**	**TX**	**RX**	**TX**	**RX**	
1	1	0	0	0	0	Audio peripheral detected, headset not inserted
1	1	0	0	0	1	Reserved for future use
1	1	0	0	1	0	Reserved for future use
1	1	0	0	1	1	Serial peripheral detected
1	1	0	1	0	0	Reserved for future use
1	1	0	1	0	1	Audio peripheral detected, headset inserted
1	1	0	1	1	0	Reserved for future use
1	1	0	1	1	1	Reserved for future use
1	1	1	0	0	0	Reserved for future use
1	1	1	0	0	1	Reserved for future use
1	1	1	0	1	0	Reserved for future use
1	1	1	0	1	1	Reserved for future use

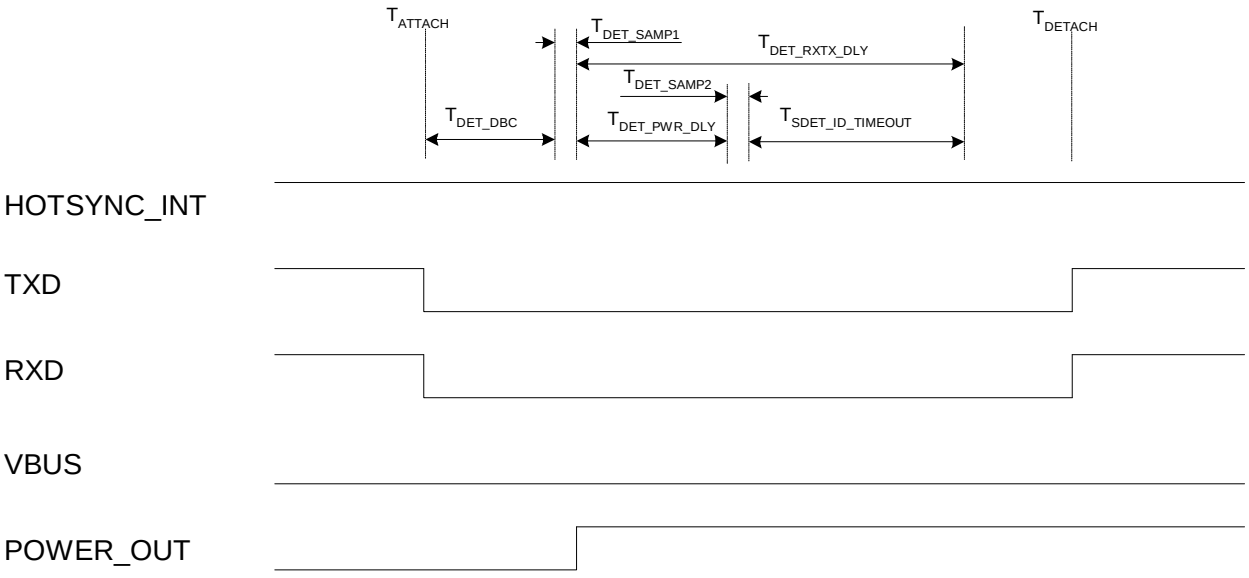
15.4.1.2 Peripheral removal

For audio class peripherals, detachment is detected by the TX and RX lines going high. The device detects these changing GPIO signals by detecting the rising edge, and initiates peripheral removal activities.

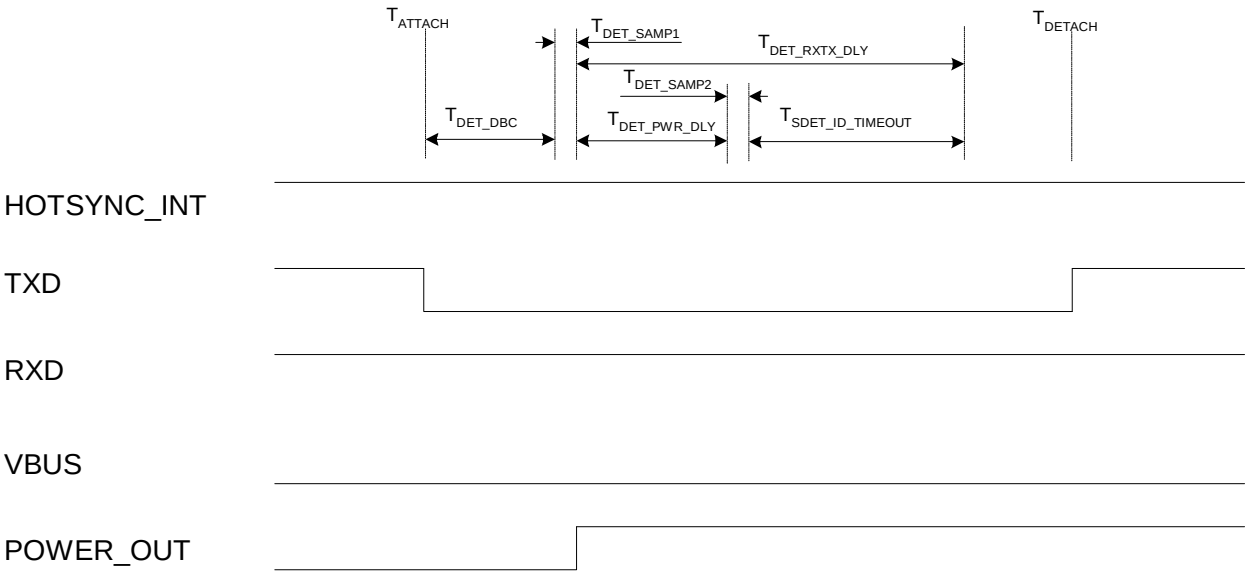
For serial class peripherals, detachment is detected by the absence of 3.3V on the USB VBUS line. The device detects this changing GPIO signal by detecting the falling edge and initiates peripheral removal activities.

15.4.2 Audio peripheral detection timing diagrams

The following figure shows the timing diagram upon attachment of an audio peripheral with a headset not inserted.

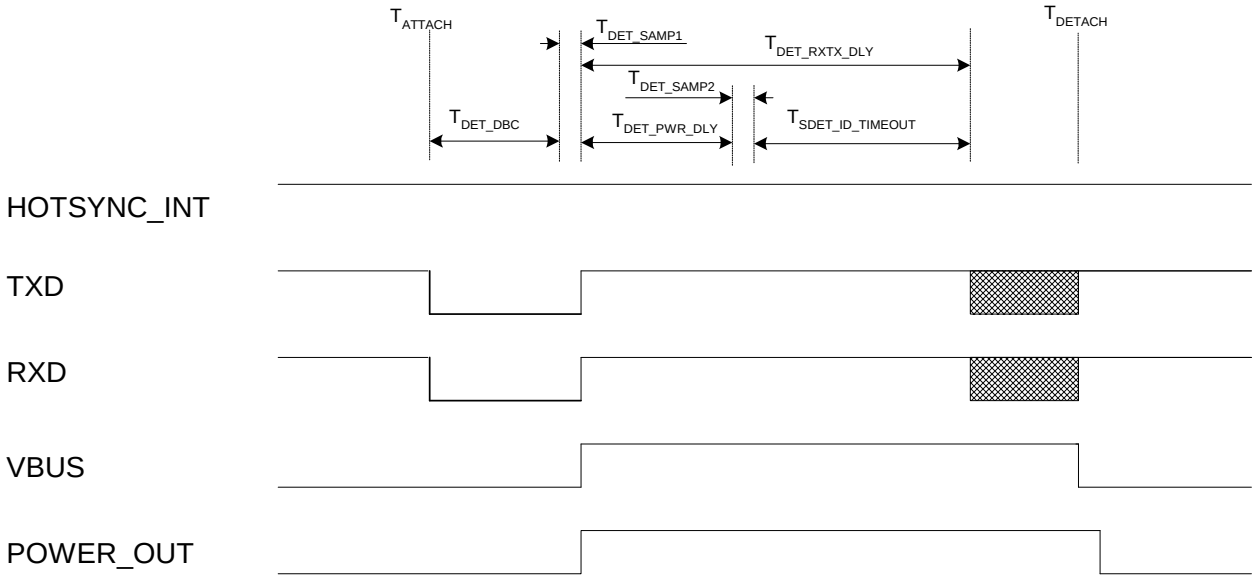


The following figure shows the timing diagram upon attachment of an audio peripheral with a headset inserted.



15.4.3 Serial peripheral detection timing diagram

The following figure shows the timing diagram upon attachment of a serial peripheral.



15.4.4 Peripheral detection timing specifications

The following are the timing requirements for the peripheral detection mechanism:

Name	Description	Minimum	Average	Maximum	Units
(T)DET_DBC	Attachment interrupt debounce duration	150	-	-	mS
(T)DET_SAMP1	Time to read the state of the RX and TX GPIO before POWER OUT applied	-	-	50	mS
(T)DET_PWR_DLY	Delay from POWER OUT applied to reading the state of the RX and TX GPIO	200	-	200	mS
(T)DET_SAMP2	Window of time from maximum (T)DET_PWR_DLY to read the state of the Rx and Tx GPIO after POWER OUT applied	-	-	50	mS

Name	Description	Minimum	Average	Maximum	Units
(T)SDET_ID_TIMEOUT	Timeout from (T)DET_SAMP2 window finished until determination that peripheral is Serial Protocol Level detection peripheral	200	-	200	mS
(T)DET_RXTX_DLY	Delay from POWER OUT application to peripheral driving into RX line	450	-	-	mS

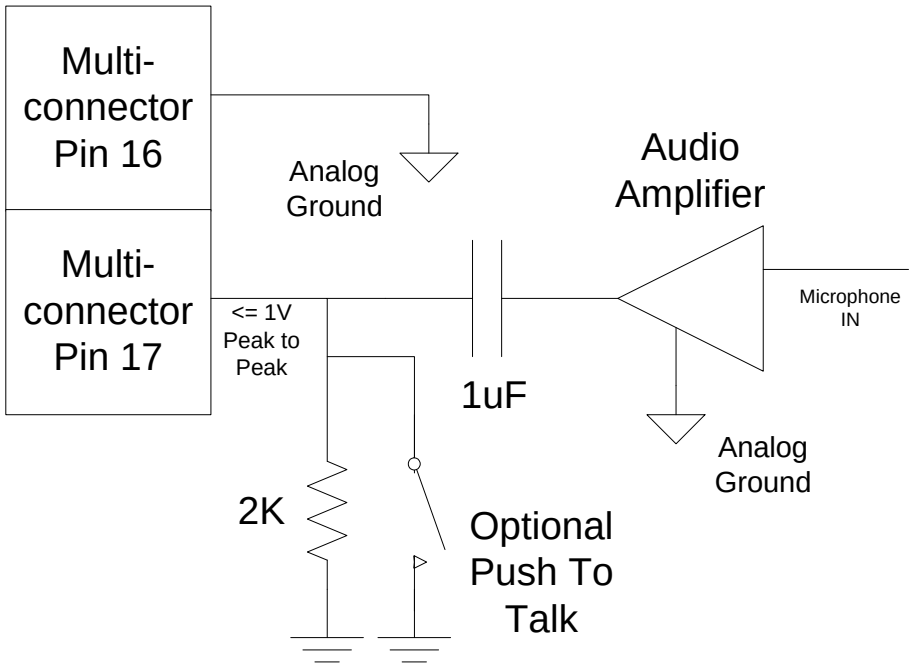
15.5 Interfacing with an audio peripheral

The recommended method for interfacing with an audio peripheral is for the peripheral to be wired as described in the following table:

Name	Pin #	Recommended configuration
VDOCK	1	Charging source if charging device; otherwise no connect.
ADAPTER_ID	2	Connect to pin 3 if charging device with >=1Amp source; otherwise no connect.
VDOCK_RTN	3	Charging source ground if charging device; otherwise system ground.
SHIELD	4	Shield ground or system ground.
VBUS	5	No connect.
USB_DP	6	No connect.
USB_DN	7	No connect.
DGND	8	System digital ground.
USB_ID	9	No connect.
TXD	10	10K Ohm pull-down to DGND.
RXD	11	Insertion detection switch on headset jack, if any. Signal should connect to GND when no headset is inserted. Signal should be high impedance when headset is inserted.
HOTSYNC	12	No connect.
POWER_OUT	13	Connect to system power if required by peripheral.
SPKR_L	14	Audio left signal.

Name	Pin #	Recommended configuration
SPKR_R	15	Audio right signal.
AGND	16	System audio ground.
MIC_IN	17	Microphone input.
SHIELD	18	Shield ground or system ground.

Circuit Needed for Proper Coupling and Bias of Microphone Input



15.6 Interfacing with a serial peripheral

The recommended method for interfacing with a serial peripheral is for the peripheral to be wired as described in the following table:

Name	Pin #	Recommended configuration
VDOCK	1	Charging source if charging device; otherwise no connect.
ADAPTER_ID	2	Connect to pin 3 if charging device with >=1Amp source; otherwise no connect.
VDOCK_RTN	3	Charging source ground if charging device; otherwise system ground.
SHIELD	4	Shield ground or system ground.
VBUS	5	Connect to POWER_OUT.
USB_DP	6	No connect.
USB_DN	7	No connect.
DGND	8	System digital ground.
USB_ID	9	No connect.
TXD	10	680 Ohm pull-up to POWER_OUT. 6.8K Ohm pull-down to DGND.
RXD	11	680 Ohm pull-up to POWER_OUT. 6.8K Ohm pull-down to DGND.
HOTSYNC	12	The presence of pull-up on this pin can be used by a peripheral as a method to detect when a device is attached to the peripheral.
POWER_OUT	13	Connect to system power if required by peripheral.
SPKR_L	14	No connect.
SPKR_R	15	No connect.
AGND	16	No connect.
MIC_IN	17	No connect.
SHIELD	18	Shield ground or system ground.

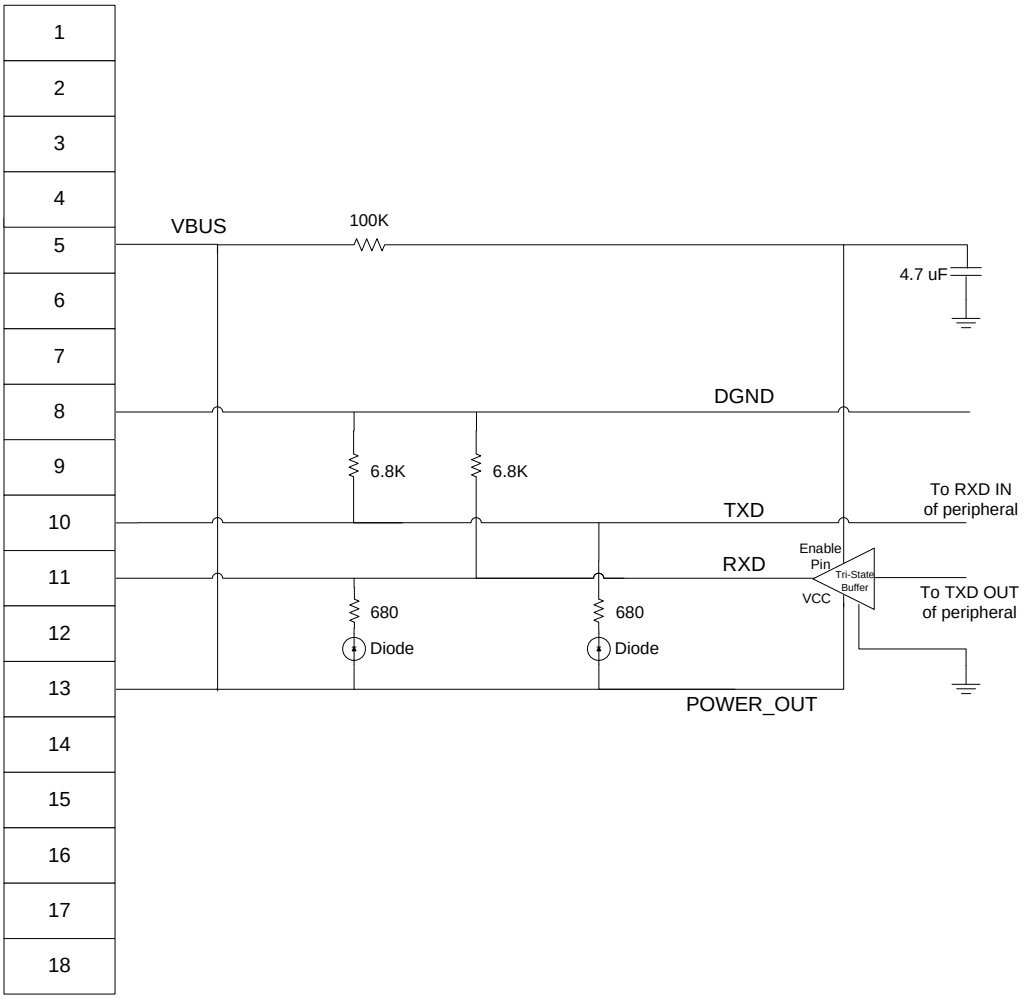
15.6.1 Electrical diagram of a serial peripheral

A peripheral that conforms to the following electrical diagram will be detected as a serial peripheral on Multi-connector devices.

To ensure that the peripheral works reliably with Treo 650 smartphones and other Palm devices, the serial peripheral's 1K resistors were changed to 680 ohm, and the 10K resistors were changed to 6.8K.

If the peripheral uses the POWER_OUT pin to detect that a device detached, it should include the diode shown in the circuit. This will ensure that there is no loop back from the Tx line (peripheral side) to the POWER_OUT pin. If a diode is included, then make sure to adjust for the voltage drop across the diode.

The Tx line of the peripheral always drives high according to the specification, and if the diode is not present, the POWER_OUT pin will drive high at all times. In this scenario, even if the device is removed, the POWER_OUT pin will stay high and prevent the peripheral from using the POWER_OUT pin to reliably detect that the device is detached.



15.6.2 Serial peripheral design guidelines

- (Optional) PWR_OUT tied to VBUS through a user selectable switch.
- 6.8K pull-down resistors on the TXD and RXD pins to ground.
- 680 Ohm pull-up resistors from TXD and RXD to PWR_OUT pins.
- Tie Multi-connector pins 4, 18 to pin 8 (digital ground).
- If the serial peripheral is going to be connected to a PC, a transceiver is needed to convert the logic level to RS-232. Make sure all flow control lines are handled at the DB9 side connected to PC. Connect pin 7 to pin 8 (RTS to CTS) and pin 1 and 6 to pin 4 (DSR to DTR and CD).
- Transceiver tied to TXD and RXD pins must be high impedance while PWR_OUT is low, and must remain high impedance 500mS after PWR_OUT goes high. This could be done with a charge up capacitor on the ENA or PWR pin of the transceiver or with a PIC microcontroller device. In other words, TXD and RXD must not be active while the peripheral detection is in progress. There should be no data on TXD and RXD while the detection is in progress.
- PWR_OUT will only source 30mA total.
- Do not drive any pins higher than 3.3V.
- Do not drive VBUS over 3.3V.
- HotSync going low is an indication of detachment detection.

15.7 Serial Peripheral Usage

15.7.1 Serial support on the Multi-connector interface

The Multi-connector interface supports serial connections with no hardware flow control. It provides 3.3V logic-level serial connections. It DOES NOT support RS-232 level serial connections.

The Multi-connector's Rx, Tx, and Gnd pins cannot be connected directly to a PC's Tx, Rx, and Gnd pins, respectively. To connect the Multi-connector to a PC, you will need a transceiver to convert the logic level to RS-232.

15.7.2 How to use the serial port

For devices that have the Multi-connector, applications must use the Serial Manager to control the serial port. To control the serial port, use the following functions:

- To open the serial port use the `SrmOpen` function.
- To close the serial port use the `SrmClose` function.
- Use the `serPortCradleRS232Port` function to define the port.

The following example shows how to open and close the serial port programmatically:

```
// To Open the Serial Port
err = SrmOpen( serPortCradleRS232Port, baudRate, &portId );

// To Close the Serial Port
err = SrmClose( portId );
```

Devices that use the Multi-connector DO NOT support port auto-detection. Also, the serPortCradlePort function DOES NOT work for serial connection.

15.7.3 Multi-connector peripheral attach and detach notifications

- `sysExternalConnectorAttachEvent` is sent when a peripheral is attached
- `sysExternalConnectorDetachEvent` is sent when a peripheral is removed
- `notifyDetailsP` determines the type of the peripheral
  - `DockStatusAttached` is sent when some type of peripheral is attached. The value is `0x0002`.
  - `DockStatusExternalPower` is sent when a peripheral is attached and using some type of external power source. The value is `0x0004`.
  - `DockStatusCharging` is sent when the internal power cells are recharging. The value is `0x0008`.
  - `DockStatusUSBCradle` is sent when a USB cable or cradle that is plugged into a PC (VBUS is present) is attached. The value is `0x0010`.

NOTE: This bit is not set for Treo 650 smartphones.

  - `DockStatusSerialPeripheral` is sent when a Multi-connector serial peripheral is attached. The value is `0x0040`.

NOTE: This bit is not set for Treo 650 smartphones.

15.7.3.1 Multi-connector serial peripheral notification

The Multi-connector uses the following serial peripheral notifications:

- `sysExternalConnectorAttachEvent` is sent when a serial peripheral is attached with `notifyDetailsP` equal to `0x0040` (`DockStatusSerialPeripheral`).
- `sysExternalConnectorDetachEvent` is sent when a serial peripheral is removed with notifyDetailsP equal to `0x0040` (`DockStatusSerialPeripheral`).

15.7.3.2 Notification support on Treo™ 650

Previously, a bug prevented notifications from being sent on Treo 650 smartphones.

Currently, this bug is fixed in all maintenance releases for Treo 650 smartphones, and notifications are now properly sent to applications.

NOTE: On Treo 650 smartphones, `DockStatusUSBCradle` and `DockStatusSerialPeripheral` bits are never set. Perform class detection using the `PmConnectorLib` library.

15.7.3.3 Notification support on Tungsten™ T5

Definitions for notifications sent on Tungsten T5 devices include:

- `DockStatusCharging` is sent when the Power Adapter is attached.
- `DockStatusUSBCradle` is sent when the USB cable or cradle is attached.

NOTE: A bug in Tungsten T5 also causes the DockStatusUSBCradle notification to be sent when a serial peripheral is attached.

15.7.4 Known Issues

A bug in Tungsten T5 causes the `DockStatusSerialPheripheral` notification to be sent when an audio peripheral is attached.

The same bug causes the `DockStatusUSBCradle` notification to be sent when a serial peripheral is attached.

15.7.5 Coding example

The following example illustrates how to use peripheral attachment and detachment notifications:

```
UInt32 PilotMain( const UInt16    cmd,
                  const MemPtr    cmdPBP,
                  const UInt16    launchFlags )
{
       Err                         error               = errNone;
       UInt16                      cardNo              = 0;
       LocalID                     dbId                = 0;
       SysNotifyParamType         *notifyP             = (SysNotifyParamType *)NULL;
       UInt16                      connectorType       = 0;


       switch (cmd)
       {
           case sysAppLaunchCmdNormalLaunch:
                  error = SysCurAppDatabase( &cardNo, &dbId );
                SysNotifyRegister( cardNo,
                       dbId,
                       sysExternalConnectorAttachEvent,
                       0,
                       sysNotifyNormalPriority,
                       0 );

              SysNotifyRegister( cardNo,
                       dbId,
                       sysExternalConnectorDetachEvent,
                       0,
                       sysNotifyNormalPriority,
                       0 );
                      ...
                      break;

              case sysAppLaunchCmdNotify:
                     notifyP       = (SysNotifyParamType *)( cmdPBP );
                     switch ( notifyP->notifyType )
                     {
                           case sysExternalConnectorDetachEvent:
                                  connectorType = ((UInt16)(notifyP->notifyDetailsP));
                                 // Connector Type is 0x0040 for the serial peripheral.
                                  break;
                        case sysExternalConnectorAttachEvent:
                           connectorType = ((UInt16)(notifyP->notifyDetailsP));
                                       // Connector Type is 0x0040 for the serial peripheral.
                                       break;
                     }
                     break;

              default:
                     break;
       }
}
```

15.7.6 Serial peripheral detection

To enable detection of a peripheral as a serial peripheral on a Multi-connector device, make sure that serial peripheral attachments and detachments are detected, and that appropriate notifications are sent, use the following steps:

1. Connect the 6.8K pull-downs on both Tx and Rx pins to the GND pin.
2. Connect the 680 ohm pull-ups on both Tx and Rx pins to the POWER_OUT pin.
3. Connect the VBUS signal to the POWER_OUT signal.

Once the serial peripheral is detected, the POWER_OUT pin will stay up as long as the peripheral is attached.

You may hear a beep when attaching or detaching the serial peripheral. This sound is related to detecting a serial peripheral attachment or detachment. But there is no guarantee that the beep will sound.

15.7.6.1 Serial peripheral detection on Treo™ 650

Previously, a bug prevented serial peripheral detection notifications from being sent on Treo 650 smartphones.

Currently, this bug is fixed in all maintenance releases for Treo 650 smartphones, and serial detection notifications are now sent properly to applications.

NOTE: Because the details bit from the notification is not set on Treo 650 devices, you may use the PmConnectorLib API to determine whether the peripheral uses a USB or serial connection.

15.7.6.2 Serial peripheral detection on Tungsten™ T5

A bug in the serial peripheral detection on Tungsten T5 devices causes the peripheral with the configuration detailed in Section 5 to be detected as a USB.

To detect a serial peripheral on Tungsten T5 correctly, use the following steps:

1. Attach the serial peripheral. A `sysExternalConnectorAttachEvent` notification with `notifyDetailsP` equal to `DockStatusUSBCradle` will be sent.
2. When the `sysExternalConnectorAttachEvent` (`DockStatusUSBCradle`) notification is received, turn off the Power_Out pin using the APIs provided in **Section 15.7.7.3 on page 306**.
3. The device will send a `sysExternalConnectorDetachEvent` notification with `notifyDetailsP` equal to `DockStatusUSBCradle`.
4. Next, the device will send a `sysExternalConnectorAttachEvent` notification with `notifyDetailsP` equal to `DockStatusSerialPeripheral`. This tells you that a serial peripheral is attached to the Tungsten T5.
5. Next, the device will send a `sysExternalConnectorDetachEvent` notification with `notifyDetailsP` equal to `DockStatusSerialPeripheral`.
6. Next, the device will send a `sysExternalConnectorAttachEvent` notification with `notifyDetailsP` equal to `DockStatusUSBCradle`. This tells you that attachment detection is complete.
7. When you remove the serial peripheral, the device will send a `sysExternalConnectorDetachEvent` notification with `notifyDetailsP` equal to `DockStatusUSBCradle`.

15.7.7 Connector library (PmConnectorLib)

Use the PmConnectorLib library to turn on and off the Power_Out pin programmatically.

15.7.7.1 PmConnectorLib on Treo™ 650

Previously, the PmConnectorLib library was not available on Treo 650 smartphones.

Currently, all recent maintenance releases of the Treo 650 smartphone and later include this library, and applications can utilize the APIs provided by the library.

15.7.7.2 PmConnectorLib on Tungsten™ T5

On Tungsten T5 devices, the `PmConnectorLib` library must use slightly different definitions than those available as part of the Palm OS SDK. Use the following definitions when using the `PmConnectorLib` on Tungsten T5:

```
#if (CPU_TYPE != CPU_68K) || (defined BUILDING_PMCONNECTOR_LIB)
   #define PMCONNECTOR_LIB_TRAP(trapNum)
#else
   #include <LibTraps.h>
   #define PMCONNECTOR_LIB_TRAP(trapNum) SYS_TRAP(trapNum)
#endif

#define kPmConnectorLibType  sysFileTLibrary
#define kPmConnectorLibCreator      'PmAt'
#define kPmConnectorLibName  "PmConnector"


#define kPmConnectorLibTrapOpensysLibTrapOpen
#define kPmConnectorLibTrapClosesysLibTrapClose

#define kPmConnectorLibCtrlPowerOn0x01
#define kPmConnectorLibCtrlPowerOff0x02

Err
PmConnectorLibOpen (UInt16 refNum)
      PMCONNECTOR_LIB_TRAP (kPmConnectorLibTrapOpen);

Err
PmConnectorLibClose (UInt16 refNum)
      PMCONNECTOR_LIB_TRAP (kPmConnectorLibTrapClose);

Err
PmConnectorLibControl (UInt16 refNum, UInt16 cmdId, void *parmP)
      PMCONNECTOR_LIB_TRAP (sysLibTrapCustom + 1);
```

15.7.7.3 Power_Out API

Once the serial peripheral is detected, the POWER_OUT pin will stay up as long as the peripheral is attached. However, if you would like to programmatically turn on or off the POWER_OUT pin, use the APIs in the following examples, which are available as part of `PmConnectorLib`:

NOTE: This API should be used only with `PmConnectorLib` version 1.3 and later.

To turn on the Power_Out pin:

```
// Send Power to the Power_out pin
{
   // Try to find the library
   err = SysLibFind(kPmConnectorLibName, &refNum);
   if (err != errNone)
       err = SysLibLoad(kPmConnectorLibType,
                 kPmConnectorLibCreator, &refNum);

   if(err) {
      FrmCustomAlert(InformationAlert,
                "Unable to load connector library", NULL,
                NULL);
   }
   else
   {
      PmConnectorLibOpen( refNum );
      PmConnectorLibControl(refNum,
                   kPmConnectorLibCtrlPowerOn,             (void *)NULL);
      PmConnectorLibClose( refNum );
   }
}

To turn off the Power_Out pin:

// Remove Power from the Power_out pin
{
   // Try to find the library
   err = SysLibFind(kPmConnectorLibName, &refNum);
   if (err != errNone)
       err = SysLibLoad(kPmConnectorLibType,
                 kPmConnectorLibCreator, &refNum);

   if(err) {
      FrmCustomAlert(InformationAlert,
                "Unable to load connector library", NULL,
                NULL);
   }
   else
   {
      PmConnectorLibOpen( refNum );
      PmConnectorLibControl( refNum,
                      kPmConnectorLibCtrlPowerOff,
                      (void *)NULL);
      PmConnectorLibClose( refNum );
   }
}
```

15.7.8 Serial HotSync

Palm, Inc. does not officially support Serial HotSync on devices with the Multi-connector.

15.7.8.1 On Treo™ 650

Previously, Serial HotSync worked on Treo 650 by disconnecting VBUS from the POWER_OUT pin.

Currently, because the class detection mechanism was implemented in maintenance releases for Treo 650 ROMs, you should no longer disconnect VBUS from POWER_OUT to force serial detection. To ensure proper detection, connect VBUS to POWER_OUT.

15.7.8.2 On Tungsten™ T5

Serial HotSync works on Tungsten T5. To use Serial HotSync, make sure that the VBUS is not connected to the POWER_OUT pin. Also, be aware that Serial HotSync only works at lower baud rates on Tungsten T5 devices.

15.7.9 Known Issues

15.7.9.1 Data transfer via the Network Preferences Panel

Using the cradle or cable in Network Preferences to connect to the network over the serial connection may not work on Tungsten devices. There are two reasons for this problem:

1. The serial peripheral as described in this guide (with pull up resistors connecting POWER_OUT to Rx and Tx, pull down resistors connecting Ground to Rx and Tx, and the POWER_OUT pin connected to VBUS) is detected by the device as a USB.

   Network Preferences uses this detection to determine which port, Serial or USB, to open for the network connection.

   To work around this issue, make sure VBUS is NOT connected to the POWER_OUT pin. This will ensure that the serial port is open for the network connection. (Tungsten T5 only.)

2. Network Preferences enables flow control.

   The serial driver on Tungsten devices, which do not have serial flow control pins, does not handle disable flow control. The serial driver enables flow control and waits for the CTS to go high before transmitting data. With flow control enabled, the CTS will never go high. Therefore, Network Preferences cannot transmit data over the serial.

   To work around this issue, disable flow control on the serial port. For instructions, see the Tungsten T5 Flow Control Sample Code included in the Palm OS SDK.

15.7.9.2 Wake/Sleep loop on Palm® T|X

When a Palm T|X handheld is attached to a serial peripheral and goes to sleep, the device will automatically wake back up. If the device is set to auto-off, this bug will cause a wake/sleep loop.

By default, the device turns the POWER_OUT pin low when it goes to sleep. This causes the VBUS pin to go low. This change of state triggers an interrupt that wakes up the device.

15.8 Interfacing with Smart serial peripherals

Smart serial peripheral detection is a detection scheme available only on Palm T|X handhelds and Treo 700p, Treo 755p, and Centro smartphones.

A smart serial peripheral is detected in two steps:

1. Smart serials use the class 110111. Do not allow smart serial peripheral detection to interfere with the class detection. In other words, smart serial peripheral detection MUST NOT drive the Tx or Rx pins during the class detection cycle. To prevent driving the Tx or Rx pins at this time, use tri-state buffers or set the GPIO pins on the Smart serial accessory to an “Input” state during class detection.

   The class detection cycle that detects a smart serial accessory is defined as the period when the POWER_OUT pin is low, then transitions to high. The period ends when the device begins to broadcast `'#''NAK'` (`0x23, 0x15`). (See Step 2 for details.)

   For details on the state of pins before and after the POWER_OUT cycle, refer to the Smart serial timing diagram that follows.

2. After class detection is completed, the Palm device will attempt handshakes with the Smart Serial accessory to identify the ID of the Smart Serial peripheral attached.

15.8.1 Smart serial peripheral handshaking process

The Smart serial peripheral interface uses the following handshaking procedure to communicate to the Palm device. Currently, only the Palm T|X and Treo 700p and Treo 755p smartphones uses the Smart serial peripheral interface.

15.8.1.1 Initialization

1. The device detects that a Smart Serial accessory is attached (Class code `0x07`).
2. The device configures its Tx pin as Serial Output and Rx pin as Serial Input.
3. The device opens the serial port at 9600 Baud (`8, N, 1`).

15.8.1.2 First Handshake

1. The device sends a `'#''NAK'` (`0x23, 0x15`) to the Smart Serial accessory.
2. The device waits to receive the Peripheral ID packet (8 bytes of data) from the Smart serial accessory:
   - If a Peripheral ID packet (8 bytes of data) is not received in 500mS, then the process enters the `SECOND_HANDSHAKE`.
   - If a Peripheral ID packet (8 bytes of data) is received within 500mS, then the packet is checked for validity:
     - The device checks that the first byte of data is `'P'` (`0x50`) and the second byte of data is `'1'` (`0x31`).
     - The device calculates the checksum by adding the first 7 bytes of data, and keeps the least significant byte of that data.
     - The device compares the calculated checksum with the last byte of data.
     - If the checksum does not match the last byte of data, then the process enters the `SECOND_HANDSHAKE`.
3. The device sends a `'#''ACK'` (`0x23, 0x06`) to the Smart serial accessory.
4. The device waits to receive `'#''ACK''O''K'` (`0x23, 0x06, 0x4F, 0x4B`).
   - If a `'#''ACK''O''K'` (`0x23, 0x06, 0x4F, 0x4B`) is not received in 500mS, then the process enters the `SECOND_HANDSHAKE`.
   - If a `'#''ACK''O''K'` (`0x23, 0x06, 0x4F, 0x4B`) is received in 500mS, then the process enters `FINALIZATION`.

15.8.1.3 Second Handshake

1. The device sends a `'#''NAK'` (`0x23, 0x15`) to the Smart serial accessory.
2. The device waits to receive the Peripheral ID packet (8 bytes of data) from the Smart serial accessory.
   - If a Peripheral ID packet (8 bytes of data) is not received in 500mS, then the process enters the `THIRD_HANDSHAKE`.
   - If a Peripheral ID packet (8 bytes of data) is received within 500mS, then the packet is checked for validity:
     - The device checks that the first byte of data is `'P'` (`0x50`) and the second byte of data is `'1'` (`0x31`)
     - The device calculates the checksum by adding the first 7 bytes of data, and keeps the least significant byte of that data.
     - The device compares the calculated checksum with the last byte of data.
     - If the checksum does not match the last byte of data, then the process enters the `THIRD_HANDSHAKE`.
3. The device sends a `'#''ACK'` (`0x23, 0x06`) to the Smart serial accessory.
4. The device waits to receive `'#''ACK''O''K'` (`0x23, 0x06, 0x4F, 0x4B`).
   - If a `'#''ACK''O''K'` (`0x23, 0x06, 0x4F, 0x4B`) is not received in 500mS, then the process enters the `THIRD_HANDSHAKE`.
   - If a `'#''ACK''O''K'` (`0x23, 0x06, 0x4F, 0x4B`) is received in 500mS, then the process enters `FINALIZATION`.

15.8.1.4 Third Handshake

1. The device sends a `'#''NAK'` (`0x23, 0x15`) to the Smart serial accessory.
2. The device waits to receive the Peripheral ID packet (8 bytes of data) from the Smart Serial accessory.
   - If a Peripheral ID packet (8 bytes of data) is not received in 500mS, then the process enters `FINALIZATION`.
   - If a Peripheral ID packet (8 bytes of data) is received within 500mS, then the packet is checked for validity.
     - The device checks that the first byte of data is `'P'` (`0x50`) and the second byte of data is `'1'` (`0x31`).
     - The device calculates the checksum by adding the first 7 bytes of data, and keeps the least significant byte of that data.
     - The device compares the calculated checksum with the last byte of data.
     - If the checksum does not match the last byte of data, then the process enters `FINALIZATION`.
3. The device sends a `'#''ACK'` (`0x23, 0x06`) to the Smart serial accessory.
4. The device waits to receive `'#''ACK''O''K'` (`0x23, 0x06, 0x4F, 0x4B`).
   - If a `'#''ACK''O''K'` (`0x23, 0x06, 0x4F, 0x4B`) is not received in 500mS, then the process enters `FINALIZATION`.
   - If a `'#''ACK''O''K'` (`0x23, 0x06, 0x4F, 0x4B`) is received in 500mS, then the process enters `FINALIZATION`.

15.8.1.5 Finalization

1. If a `'#''ACK''O''"K'` (`0x23, 0x06, 0x4F, 0x4B`) is received, then the Class Code and Peripheral ID received in the Peripheral ID packet is broadcast.
2. If a `'#''ACK''O''"K'` (`0x23, 0x06, 0x4F, 0x4B`) is not received, then the device will “soft” detach and re-attempt to detect the class type. (POWER_OUT goes low without mechanical detachment, then high after a time period).

See the following table for an example of the correct handshake sequence:

Step	Palm device	Smart Serial accessory
1	Class Detection	NO ACTION
2	'#''NAK' (hex 23, hex15)	Send Peripheral ID CODE
3	Validate Peripheral ID code, Checksum	NO ACTION
4	'#''ACK' (hex 23, hex 06)	'#''ACK'O''K' (hex 23, hex 6, hex 4F, hex 4B)

15.8.1.6 Handshake rule exception

If the Smart Serial accessory does not receive '#''NAK' from the device for 0.6 seconds, it will broadcast the Peripheral ID code unconditionally, and wait for '#''ACK' or '#''NAK' from the device for a maximum of 1.5 seconds. If the device and Smart serial accessory do not achieve a complete handshake sequence, then both will reset to an undocked condition and go through the process again.

15.8.1.7 Sample use case

When a valid class detection is complete, Treo 650 will begin to broadcast "#''NAK' (hex23, hex 15) repeatedly until either the peripheral returns a valid response, or the Treo 650 device times out (after 1 second without a valid reply from the peripheral).

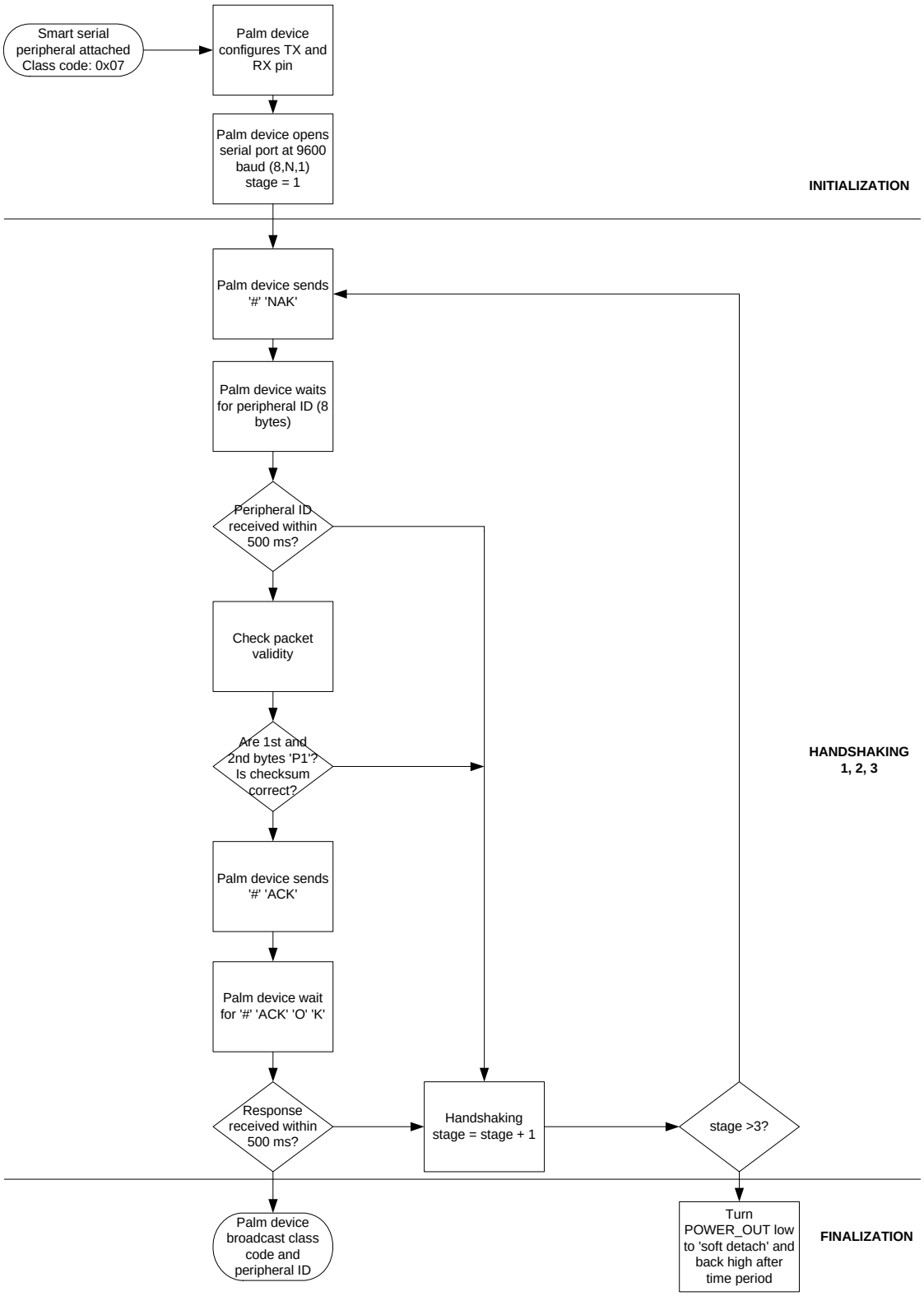
If the peripheral broadcasts the correct Creator ID code, then the device will broadcast '#''ACK' (hex 23, hex 6). The device will continue to broadcast '#''ACK' until the peripheral returns a valid reply.

Upon seeing the valid '#''ACK' from the device, the peripheral will reply '#''ACK''O''K' (Hex 23, Hex 6, Hex 4F, Hex4B).

If the peripheral never sees a '#''NAK' from the device after .5 seconds, it will broadcast the creator ID code unconditionally, then wait for '#''ACK' from Treo 650 for up to 1 second.

If neither the device nor the peripheral manages to synchronize after 2 seconds, both will reset to an undocked condition and go through the process again.

The tables on the following pages are examples of the correct handshake sequence.


Smart serial peripheral attached Class code: 0x07
Palm device configures TX and RX pin
Palm device opens serial port at 9600 baud (8,N,1) stage = 1
INITIALIZATION
Palm device sends '#' 'NAK'
Palm device waits for peripheral ID (8 bytes)
Peripheral ID received within 500 ms?
Check packet validity
Are 1st and 2nd bytes 'P1'? Is checksum correct?
HANDSHAKING 1, 2, 3
Palm device sends '#' 'ACK'
Palm device wait for '#' 'ACK' 'O' 'K'
Response received within 500 ms?
Handshaking stage = stage + 1
stage >3?
Palm device broadcast class code and peripheral ID
Turn POWER_OUT low to 'soft detach' and back high after time period
FINALIZATION

	Byte Order	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Palm Device		#	NAK	#	NAK									#	ACK				
	Data	0x23	0x15	0x23	0x15									0x23	0x06				
Smart Serial Accessory						P	1	0	P	A	I	m	CKSUM			#	ACK	O	K
	Data					0x50	0x31	0x00	0x50	0x61	0x6C	0x6D	0x0B			0x23	0x06	0x4F	0x4B

This page intentionally blank

CHAPTER 16

16. Headset Jack Specifications

This chapter defines the interface and interactions of the headset jack and its surrounding circuits and controlling software.

16.1 Overview

Available on:

- Centro™ and Treo™ smartphones

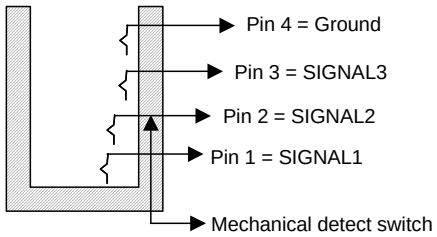
Audio accessories are used for the following general categories of functions:

- Cell phone audio—Mono-aural sound and mic input
- Stereo audio—MP3 playback, game, multimedia content
- Voice input—Voice dialing, voice recording

Typically, mono-aural sound accessories for cell phones have 2.5mm plugs with three or four contacts, while audio accessories that provide stereo audio (music) often have 3.5mm plugs.

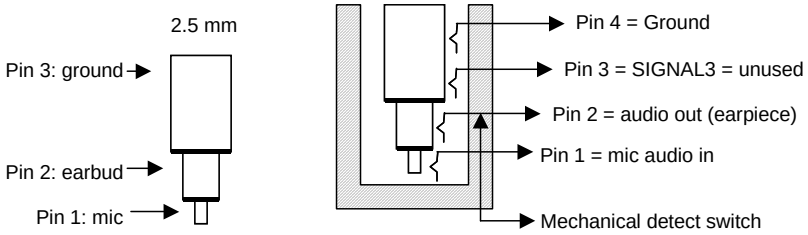
Palm smartphones use a 2.5mm plug. A 3.5mm headphone accessory may be used in a 2.5mm headset jack through a 3.5mm-to-2.5mm adapter/converter (available at most electronics retailers). This converter adapts the physical connector size while retaining the electrical signal connectivity. This chapter assumes that the 3.5mm accessories use a 3.5mm-to-2.5mm adapter in order to work with Palm smartphones.

Most stereo accessories (3.5mm with adapter) have three contacts on the plug. 2.5mm cell phone headsets may have three or four contacts on the plug (three-pole or four-pole plugs). The functional description of this headset jack is illustrated in the following diagram:

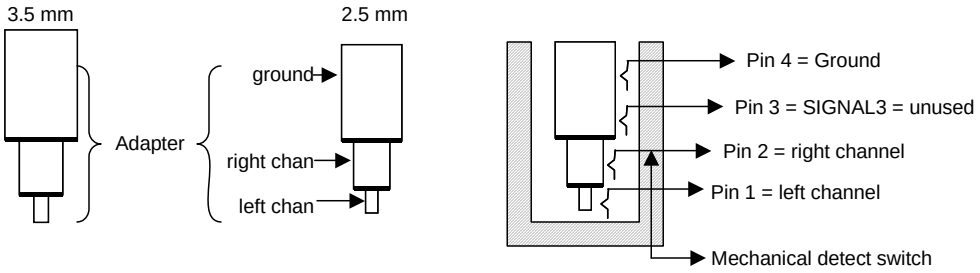


A mechanical detect switch is used to detect the presence of an external accessory. Electrically, SIGNAL1, SIGNAL2, and SIGNAL3 can each be biased separately on the motherboard to detect the impedance of the inserted audio accessory. These signals are measured by the applications processor to distinguish among possible accessories. The possible connection configurations are shown in the following figures.

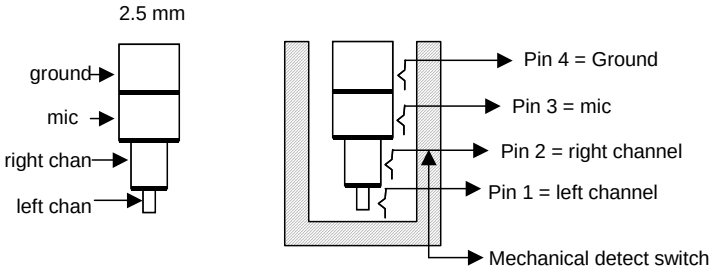
16.1.1 Standard 2.5mm cell phone headset (3-pin)



16.1.2 Stereo headphones (3-pin, 2.5mm or 3.5mm via adapter)



16.1.3 (Custom) Combination headphone/headset (4-pin, 2.5mm)



Each accessory is distinguished by the amount of impedance (electrical resistance) it presents to each signal pin (SIGNAL1, SIGNAL2, and SIGNAL3). Each signal may be biased via a resistor network. When the accessory is plugged into the jack, the voltage present at each pin is measured by the application processor and a decision algorithm is followed.

16.2 Stereo audio accessories

Stereo accessories that use passive speakers may have impedances of 8Ω, 16Ω, 32Ω, or 150Ω. Powered or amplified speakers may have similar low impedance, or they may have 2kΩ, 10kΩ, or higher impedances.

Stereo audio accessories are identified by the *relative* impedance of the left and right channels. The distinguishing characteristic of stereo accessories is that the right and left channels present the same level of impedance to the audio circuitry. Consequently, the application processor needs to measure the impedance on SIGNAL1 and SIGNAL2.

- If SIGNAL1 and SIGNAL2 are both high impedance, the software assumes that an active stereo circuit accessory has been inserted, and it routes the right and left audio channels appropriately.
- If SIGNAL1 is high impedance and SIGNAL2 is low impedance, the software assumes that a mono-aural cell phone headset has been inserted, and it routes the mic signal to SIGNAL1.
- If SIGNAL1 and SIGNAL2 are both low impedance, the software assumes that an active stereo circuit accessory has been inserted, and it routes the right and left audio channels appropriately.

The following table illustrates the results:

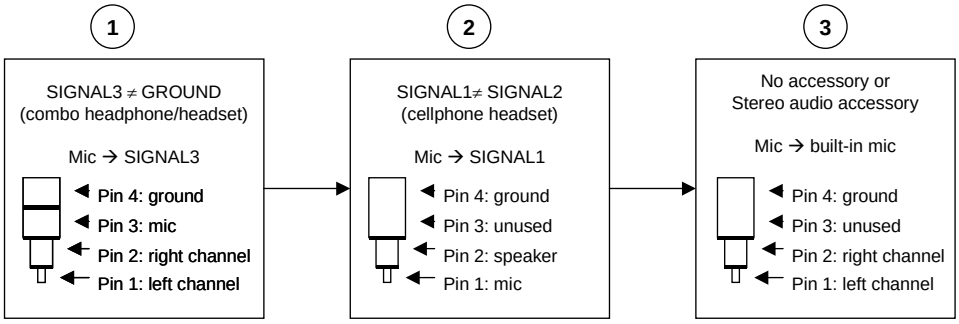
Processor measures SIGNAL1 and SIGNAL2	Processor routes audio signals	Result
SIGNAL1 = high impedance SIGNAL2 = high impedance	SIGNAL1 = left channel SIGNAL2 = right channel	Stereo
SIGNAL1 = high impedance SIGNAL2 = low impedance	SIGNAL1 = microphone SIGNAL2 = passive speaker	Mono
SIGNAL1 = low impedance SIGNAL2 = low impedance	SIGNAL1 = left channel SIGNAL2 = right channel	Stereo

16.3 Microphones

There are three different microphone sources available in Treo 650 smartphones and later: the built-in internal mic, the custom combination headphone/headset accessory, and the standard cell phone headset accessory.

The three potential microphone sources can be detected sequentially in the following priority:

1. Measure SIGNAL3. If SIGNAL3 is high impedance (not equal to GROUND), then the software assumes that a combination (hybrid) stereo headset/headphone has been detected. The mic path is routed to SIGNAL3.
2. Measure SIGNAL1 and SIGNAL2. If SIGNAL1 is high impedance and SIGNAL2 is low impedance, then a mic has been detected on SIGNAL1 (mono headset accessory). The mic path is routed to SIGNAL1.
3. If no accessory is inserted, then the software routes the signal to the built-in mic.



16.4 Speaker Architecture

Treo 650 smartphones and later have a two-speaker audio architecture. One speaker, called the receiver, is mostly dedicated to telephony sound and is tuned to voice frequency. The other, the external speaker, is mostly dedicated to system sounds and is tuned to polyphonic sounds. It is also used for the speakerphone mode. Palm smartphones also enable the user to play stereo sound through the headset jack.

Palm smartphones have improved sound support, and their audio subsystem consists of three major categories:

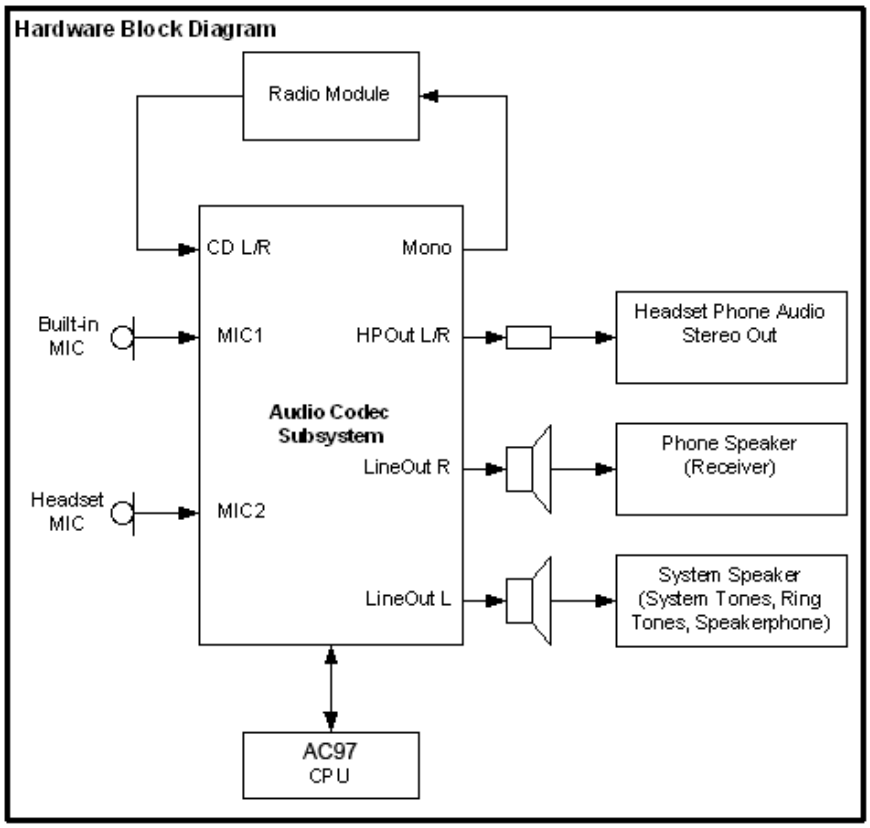
1. **Radio audio control -** Enables audio for the cellular radio, playback for received streams, and encoding and sending for recorded voice.
2. **System audio control -** Controls the speaker output for polyphonic sound playback and the microphone input for voice recording.

   NOTE: The Treo 600 smartphone does not capture the microphone input through an A/D converter.

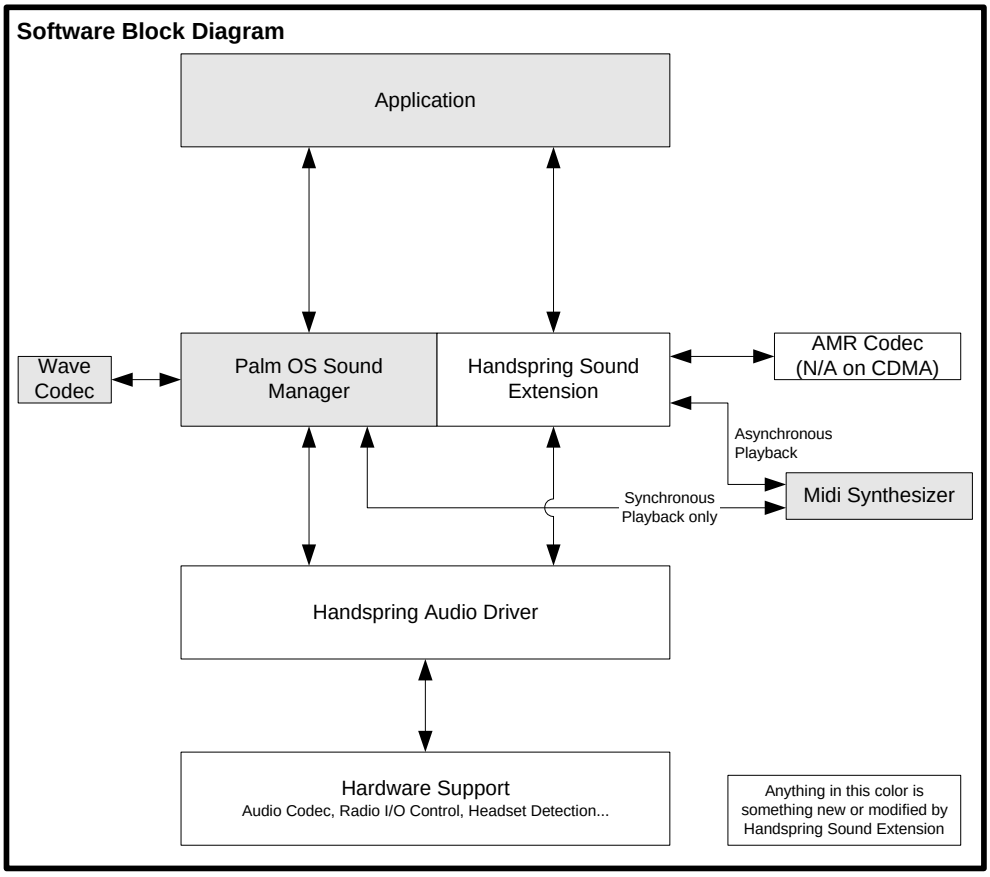
3. **Ring Tone Manager -** Controls ring tone playback when a call is received.

It's also possible to redirect the radio module audio output to the SDIO connector so that an external card could make use of it. For example, the audio output could be sent to a headset using Bluetooth® wireless technology.

The following figure shows the Treo smartphone audio subsystem hardware.



The following figure shows the Treo smartphone audio subsystem software.



The usual software interface for playing sound is the standard Palm OS Sound Manager. Palm has added special controls that automatically direct the sound playback to the right output, as described in the following tables.

The only other APIs you need are found in the *API Guide*. These enable control of the audio stream when it is needed. As mentioned earlier, Palm® extension to the Palm OS should automatically take care of switching the right input/output, depending on usage.

16.5 Usage scenarios

16.5.1 Treo™ 600 smartphones

The usage scenarios in the following tables show how the Treo 600 smartphone system interacts with the audio subsystem when playing or recording audio stream(s) (that is, what types of sounds are routed to which inputs and outputs). Some of the supported scenarios are not currently used by Palm, but could be created by a third party (6, 7, 10–24).

	Usage scenario	Voice sound in	Voice sound out	If phone sounds*	If system sounds**
1	Voice call using smartphone only	Built-in mic	Built-in receiver	Mixed-in, built-in speaker	Built-in speaker
2	Voice call using headset	Headset mic	Headset speaker	Headset speaker	Built-in speaker
3	Headset plugged in, not on a call	N/A	N/A	Headset speaker and built-in speaker	Built-in speaker
4	Stereo headphone plugged in, not on a call	N/A	N/A	Headset speaker and built-in speaker	Built-in speaker
5	Voice call using speakerphone	Built-in mic	Built-in speaker	Mixed-in, built-in speaker	Mixed-in, built-in speaker
6	Voice call using other type of headset	OT headset mic	OT headset speaker	OT headset speaker	Mixed in OT headset speaker
7	Other type of headset in use, not on a call	N/A	N/A	OT headset speaker & built-in speaker	Built-in speaker
8	Car kit on a call	Car kit mic	Car kit speaker	Mixed-in car kit speaker	Built-in speaker
9	Car kit off a call	N/A	N/A	Car kit speaker	Built-in speaker
10	PTT voice call using headset	Headset mic	Headset speaker	Headset speaker and built-in speaker	Built-in speaker
11	PTT voice call using speakerphone	Built-in mic	Built-in speaker	Mixed-in, built-in speaker	Mixed-in, built-in speaker
12	Voice record memo	Built-in mic	N/A	Built-in speaker—priority	Built-in speaker
13	Voice command using smartphone only	Built-in mic	N/A	Built-in speaker	Built-in speaker
14	Voice command using headset	Headset mic	N/A	Headset speaker and built-in speaker	Built-in speaker
15	Voice command using car kit	Car kit mic	N/A	OT headset speaker	Built-in speaker

	Usage scenario	Voice sound in	Voice sound out	If phone sounds*	If system sounds**
16	MP3 music playback—smartphone only	N/A	N/A	Mixed-in, built-in speaker	Mixed-in, built-in speaker
17	MP3 music playback—headset	N/A	N/A	Mixed-in headset and built-in speaker	Mixed-in headset and built-in speaker
18	MP3 music playback—stereo headphone	N/A	N/A	Mixed-in headphone and built-in speaker	Mixed-in headphone and built-in speaker
19	Playing games—smartphone only	N/A	N/A	Mixed-in, built-in speaker	Mixed-in, built-in speaker
20	Playing games—headset	N/A	N/A	Mixed-in headset and built-in speaker	Mixed-in headset and built-in speaker
21	Playing games—stereo headphones	N/A	N/A	Mixed-in headphone and built-in speaker	Mixed-in headphone and built-in speaker
22	Voice memo playback—smartphone only	N/A	N/A	Mixed-in, built-in speaker	Mixed-in, built-in speaker
23	Voice memo playback—headset	N/A	N/A	Mixed-in headset and built-in speaker	Mixed-in headset and built-in speaker
24	Voice memo playback—stereo headphones	N/A	N/A	Mixed-in headphone and built-in speaker	Mixed-in headphone and built-in speaker

Notes for these usage scenarios include:

- Input
  - **Built-in mic -** The microphone contained in the handset for use when the headset is held to one's ear.
  - **Headset mic -** The microphone contained in the headset.
  - **Other headset or car kit mic -** The microphone contained in the appropriate peripheral kit.
- Output
  - **Built-in receiver -** The speaker on the front of the handset that is used when the handset is held next to the ear. Generally used for listening to voice calls.
  - **Built-in speaker -** The louder speaker located on the back of the handset. Generally used for system sounds and speakerphone mode.
  - **Headset, OT, or car kit speaker -** The speaker built in to the appropriate peripheral.
- *"If phone sounds": This refers to how the audio is routed if the audio from a class of sounds dedicated to phone usage interrupts the current usage scenario. This includes the following types of sounds:
  - Ring tones.
  - Call progress tones.

- DTMF (dual tone multi-frequency).
- Low battery. This is technically a system sound, because it is generated by the PDA. It is treated as an exception because a low battery warning is critical phone-related information that must be heard when on an active call, regardless of what the system sound settings are.

**"If system sounds": All system sounds will be played through both the headset/headphone and built-in speaker except the system sounds for alarms, SMS alerts, and Mail alerts.

16.5.2 Treo™ 650 smartphones and later

The usage scenarios in the following table show how Treo 650 smartphones and later interact with the audio subsystem when playing or recording audio stream(s).

	Usage scenario	Ringer switch sound on/off	On active call?	Telephony audio	Alerts (Attention Manager)	MP3 (application audio)	High-priority system sounds
1	Base mic/speaker	OFF	YES	Receiver	No sound[a]	No sound	No sound
2	Base mic/speaker	OFF	NO	No sound	No sound	No sound	No sound
3	Base mic/speaker	ON	YES	Receiver	Receiver[a]	Mute/hold	Receiver[a]
4	Base mic/speaker	ON	NO	No sound	Speaker	Speaker	Speaker
5	Speakerphone mode	OFF	YES	No sound	No sound[a]	No sound	No sound
6	Speakerphone mode	OFF	NO	N/A	N/A	N/A	N/A
7	Speakerphone mode	ON	YES	Speaker	Speaker[a]	Mute/hold	Speaker[a]
8	Speakerphone mode	ON	NO	N/A	N/A	N/A	N/A
9	Headset inserted	OFF	YES	Headset	Headset[a]	Mute/hold	Headset
10	Headset inserted	OFF	NO	No sound	Headset	Headset	Headset
11	Headset inserted	ON	YES	Headset	Headset[a]	Mute/hold	Headset[a]
12	Headset inserted	ON	NO	No sound	Speaker and mute MP3	Headset	Speaker and mute MP3
13	Bluetooth headset in use	OFF	YES	Bluetooth	No sound	Mute/hold	Bluetooth
14	Bluetooth headset in use	OFF	NO	No sound	No sound	No sound	No sound
15	Bluetooth headset in use	ON	YES	Bluetooth	Speaker	Mute/hold	Bluetooth
16	Bluetooth headset in use	ON	NO	No sound	Speaker	Speaker	Speaker

a - Alerts should be sound mixed-in at a lower volume than the active call in progress.

CHAPTER 17

17. External Hardware Drawings

This chapter explains where to find external hardware drawings for Palm devices and peripherals.

17.1 Palm™ devices

Available for:

- Centro™ and Treo™ smartphones
- Tungsten E2 and Tungsten T5
- LifeDrive
- Palm T|X
- Palm Z22

External Hardware Drawings are provided in IGES or STP format for download at the Palm Developer Network (PDN) website. To find the external surface drawing for your device, go to the PDN Knowledge Base and search for "External Hardware Drawing." Links to all available device drawings will be returned in the search results.

17.2 Expansion Parts Store

The Expansion Parts Store, hosted by Northstar Systems, offers PDN software and hardware developers development kits (including hardware drawings) for the creation of peripherals for Palm products. Supported devices currently include Treo 650 smartphones, Tungsten T5, and many other handheld and mobile manager devices. The Northstar site includes information on connectors, cables, modem housings, handheld plastics, as well as SDIO and Bluetooth development tools.

To find the link to the Expansion Parts Store, log into PDN, then use the side navigation menu to go to **develop > get devices > parts**.

17.3 Cable and connector drawings

ATL Technologies manufactures cable assemblies and connector systems used with Palm devices. At their website, ATL provides external hardware drawings in PDF format for peripherals including the USB sync cable, power cable, Multi-connector, and a charging cradle.

To find the link to ATL Technologies, log into PDN, then use the side navigation menu to go to **develop > get devices > parts**.

This page intentionally blank

PART VI

Appendix

This part of the guide provides supplemental reference information.

This page intentionally blank

APPENDIX A

Sample Code

The Sample Code provided in the Palm OS SDK serve as examples to illustrate programming concepts and API usage. The following table includes a title and description for each Sample Code included in the Palm OS SDK.

For complete details on each Sample Code, refer to the ReadMe files included in the Palm OS SDK.

Sample Code Title	Description
_CDMAStatus	This sample code shows how to retrieve radio connection status, operator name, and voicemail number on CDMA phones. It also shows how to get the base station info on Centro CDMA devices
_GSMStatus	Shows how to retrieve operator status, name, and voicemail count on GSM phones.
_HelperReceiver	Shows how to use Palm's Helper APIs. It shows how to register, enumerate, validate and execute the system's helper notifications for SMS, Email, and WEB.
_HelperSender	Shows how to use Palm's Helper APIs to broadcast helper notifications for SMS, Email, and Web to registered applications.
_HTTPLibTest	Demonstrates how to use the built-in shared HTTP Library.
_Keyguard	Shows how to activate the Keyguard programmatically, and also disable it every time there is an incoming SMS alert.
_LaunchCommands	Shows how to use various Phone App launch commands to dial or show keypad. It also shows how to launch the Web browser, including in minimal mode.
_MMSCounter	Shows how to register an application for incoming message notifications and retrieve the number of unread MMS messages.

Sample Code Title	Description
_MMSReceiver	Shows how to use Palm's Helper APIs to register, enumerate, validate and execute the system's helper notifications for MMS.
_PalmPhotoCapturePreview	Shows how to use the Palm Photo Library to use the Camera interface to capture a photo, preview the photo and save it to the device's photo album.
_PhoneEventReceiver	Shows how to register for Phone Events PhnEvt***** to receive phone event notifications from the phone library.
_PhoneInfo	Shows how to use the Palm Telephony Library to display useful information.
_RadioSample	Shows how to detect if the radio is off or on, how to turn the radio on/off, how to query the SIM card for the phone number of the device and how to check if the device has service available using the signal level.
_REMTimer	Shows how to programmatically register and put the device into normal sleep and REM sleep.
_RingTone	Shows how to get the list of tones from the system database, play the tones, and stop the tone play.
_SimpleSMS	Demonstrates how to use the built-in shared SMS Library.
_STETest	Shows how to use the Palm Smart Text Engine.
AddEmailtoVersaMail	Shows the different methods used to add a newly composed email to VersaMail (via exchange manager, sub launch [not supported in VM3.0], UI).
BluetoothMgr	This sample demonstrates how to use the Bluetooth Library including the Bluetooth Manager APIs and Socket and Service Discovery APIs. It establishes an ACL connection between two Bluetooth enabled devices and allows querying for the name of the connected device. It also allows sending a string from one device to the other.
BtClientServer	Uses the Serial Manager to implement a Bluetooth application that can be run in either client or server mode. The application sends files from the client to the server over Bluetooth.

Sample Code Title	Description
CameraV2	Shows how to use the Camera Manager library. **NOTE:** This library may not be available on older devices.
CameraV3	Shows how to add basic camera/imaging functionality by using the Camera Manager library.
Conduit	Demonstrates how to synchronize between records on the handheld and records on the desktop. The conduit includes three applications: 1. MLConduit 2. MLDesktopApp 3. MLDeviceApp
CustomFav	Shows how to use the Favorites DB Library to create custom favorites in the Phone App view (blue grid).
Extended Sound	Shows how to use the extended sound manager API, and features recording, playback, etc.
FileBrowserSample	Shows how to use the FileBrowser API to browse through files on the device and look for text files. This sample code should be run on Life Drive and Tungsten T5 devices.
FiveWayV2	Shows how to detect key presses/releases using the navigation APIs which support 5-way navigation (top, down, left, right, center).
FotoFinish	Shows how to use the photo library APIs and provides simple photo editing functionality.
FotoFinish2	Similar functionality to the FotoFinish sample application, except that FotoFinish2 also uses version 2 photo library APIs.
FullScreenWriting	Shows how to use GoLCD APIs that allow the user to write Graffiti® on and provide input directly from the screen.
HtmlLibTest68K	Shows how to use the built-in HTML library to render HTML data and display HTML pages on a Palm OS form.
IRCommunication	Shows how to use the Serial Manager to transmit and receive Raw IR Data and IR COMM data. This sample code also shows how to wrap and unwrap the Raw Data in an IrDA packet.

Sample Code Title	Description
MemoryInfo	Shows how to query the amount of free memory in the dynamic heap and in the DBCache.
MiniADPCM	Shows how to encode and decode ADPCM samples using the Codec Plug-in Manager.
MiniBuzzer	Shows how to use the SoundFileStream library to record and play various audio formats.
MiniGIF	Shows how to encode and decode GIF images using the Codec Plug-in Manager.
MiniImage	Shows how to use the image codecs to encode and decode between various file formats.
MiniMP3	Shows how to use the Codec Plug-in Manager to play MP3 files.
MiniMPEG4	Shows how to encode YCbCr420Planar and YCbCr422Planar to MPEG4 video format and decode MPEG4 video format to YCbCr420Planar format.
NavExample	Shows how to create a 5-way enabled app using the fnav resources. Refer to the rcp file for usage.
NetSelector	Shows how to use the NetPref library to iterate through existing network profiles in the Network panel and use the NetMaster library to initiate data connection using a non-default profile.
NVFSTest	Shows how to commit database changes to the NAND drive while application is running and database is open.
Pushdemo	Shows how to set up a connection between a device (the client) and a PC (the server), and then use that connection to push packets of data from the PC to the device.
PmConnectorTest	Demonstrates the PmConnector Library APIs. It shows how to disable automatic detection of serial devices and how to enable/disable the power_out pin that supplies power to peripherals that are not Multi-connector compliant.
SerialCommunication	Shows how to use the Serial Manager to transmit and receive serial data. It also shows how to detect a serial peripheral and how to Power On/Off the Multi-connector POWER_OUT pin. It additionally demonstrates how to work around the flow control problem on Tungsten T3.

Sample Code Title	Description
StatusBar	Shows how to detect the Slider status on Tungsten T3. It also shows how to detect the screen rotation on Tungsten T3 and Tungsten T5.
TickStock	TickStock is a sample application that demonstrates: ■ How to detect and handle incoming call interruption ■ How to detect if device is using EvDO connection ■ Downloading data while keeping UI responsive ■ Implementing [background] HTTP request as a state machine This application connects to Yahoo! Finance website and downloads the HTTP content until it finds the latest price of the queried stock.
Treo650PowerOut	Shows how to power on/off the Multi-connector POWER_OUT pin on Treo 650.
TungstenT5FlowControl	Shows how to enable/disable flow control on the Tungsten T5 device.
TungstenT5SwitchAudio	Shows how to switch audio from speaker to Multi-connector and from Multi-connector to speaker on Tungsten T5.
TungstenT5SwitchAudioAlt	Shows how to switch audio from speaker to multi-connector and from multi-connector to speaker on Tungsten T5.
VersaMailAccConfig	Shows how to configure email accounts using the __MMDevice database.
VersaMailAttchmntPlugin	Versamail attachment plug-in for text plug-ins.
VersaMailFontApp	Demonstrates the VersaMail Font Manager APIs. It shows how to change the size, style or font of text.

This page intentionally blank

Index

Numerics

16-bit color 35, 36, 37, 63, 262
1xRTT network parameters 75
1xRTT vs. EvDO connections 112
5-way navigator 34, 36, 37
  5-way Navigator and Keyboard API 157–161
  default navigation 159
  hardware matrix 34, 36, 37
  keystroke equivalents 241
  Option and Shift key APIs 140
  overview 158
  rocker (on Tungsten T5) 165
68K PACE debugger nub 237

A

Accept headers 203, 276
Accept-Language string 276
ACCESS (PalmSource)
  Developer Program 22
  documentation 19
access points (handhelds) 93
accessing
  data 31, 144
  file volumes for Tungsten 153
  internal file volumes 152
  tips 181
  tutorials 181
  web pages 200, 201, 211, 275
Acrobat files 203
acronyms for use with Blazer 276
action buttons 157, 161
ADAPTER_ID parameter 296, 298
adding
  email accounts in VersaMail 213
  header images in web pages 271
  images to training tools 182, 187, 194
  images to web pages 267, 269
  insertion points 180
  network profiles 93
  plug-ins 220
  ring tone management applications 69
  ring tones 69, 71
  sound databases 69
  system gadgets 279, 280
  trusted certificates 88
  web pages 261
addrAppNotificationCreateNewRecord command 111
address information in the Message database 122
Adobe Acrobat files 203
AGND parameter 285, 297, 298
alarm management tool 69
alerts
  alert tones 69
  Attention Manager 324
  Debug Prefs button options 245
  Debug Prefs settings 243
  displayed by the Debug Simulator 239, 242
  for extreme low memory condition 154
  in SMS messages 115
  SMS alert 329
ALT tag 267, 269
always-on data session 112
AMR 68
AOUT_MIN parameter 291
apn key (VersaMail) 215
AppCmdLaunch.h header file 218
application focus mode 157, 159
applications
  assigning secondary button features 159
  camera slider notifications and 64
  changing databases from 145, 151
  compatibility specifications by device 41, 43
  creating ring tone management 69
  customizing navigation 162
  displaying certificates from 88
  displaying formatted help for 181
  enabling tethered mode for 125
  for Treo smartphones 29
  full-screen writing and 196
  HTTP requirements and third-party 85
  icons 127, 128
  incoming message events and 117
  key presses and 140
  launching Network Preference dialog from 79

linking to web pages 211, 212
NetLib library interface and 86
opening bitmap files and 127
outgoing email and 216
registering file types for 127
registering with Exchange Manager 220
running on multiple devices 166, 176
sleep-request notifications and 135, 136
SMS messages and 115, 120, 121
stub 193
switching volumes and 128
TCP sockets and 85
updating GSM indicator from 102
updating Operator indicator from 102
updating Voicemail indicator from 102
ARM debugger nub 243, 244, 245
AT command 125
Athena connector *See* Multi-connector
attachments
email 213, 218, 220
serial peripheral 304
Attention dialog box 175, 176
Attention Manager 139, 324
audio 66–72, 322
API 66
applications 141
codecs 205
detecting 289
detecting peripherals 289
devices
detecting 292
detection timing specifications for 294
interfacing with 296, 298
requirements for 291
embedded audio playback 263
files 141
output 289, 319, 320
pause and resume 67
peripheral devices 289
playback 67, 263
recording 67
speaker architecture 319
streaming audio and video 204, 264
Tungsten handhelds 289
detecting audio peripherals for 292, 294
authentication
HTTP library 85, 87, 88
SMS 116
web server for Blazer 202
auto-keyguard mode 140
automatic message reassembly 120
auto-off
timeout value 135, 139
timer (REM sleep) 136
timer reset 139
wake/sleep loop on Palm TX 308

B

backward compatibility
DIA 197
FrmNav API calls 166
Palm OS network preferences 79
batteries 34, 36, 37
battery
conservation with REM sleep 135
DBCache issues 151
external dialogs 61
gadget 279
keystroke equivalents 241, 242
low battery alert 323
NVFS 155
power management 227
soft reset 237
beaming
bookmarks 199
digital rights management 207
incoming 127
launch debugger 244
BGSOUND tag 263
binary
loading the binary in CodeWarrior 247
messages 117, 120
segmentation 120
bitmap
codec 56
data returned by Camera Manager 63
family 127
File Browser API 127
files 127, 128
NVFS 156
STE rendering engine 134
wireless bitmap graphic format 277
Blazer web browser 199–212, 261–278
designing for 261
displaying application tips and 193
displaying web pages and 275
encoding formats for 275
extending HTML support for 277
international support for 275–276
overview 200, 273
saving images and 206
streaming audio and video 204
Treo smartphone support for 29
unsupport HTML elements for 272
Bluetooth

1.2, resolves car kit issue 30
Blazer Palm OS Network Library 200
BluetoothMgr Sample Code 330
by device 35, 36, 38
development tools 19, 325
gadget 279, 280
headset in use 324
headsets 319
on-device debugging 236
option (email) 216
radio status for Centro 279
Ring Tone Manager 319
VersaMail 216
bookmarks
beaming 199
Blazer 201, 263
boot code 144
breakpoints
CodeWarrior 246, 247, 249
PalmDebugger 254
broadcasting sleep-request notifications 135
bulleted lists 189
button
5-way navigator 34, 36, 37, 158
action 157, 161
Center 176
Debug Prefs options 245
Down 177
events 135, 159
keystroke equivalents 241
New Folder button in File Browser API 131
Option and Shift 140
Play, browser 264, 266
power 135, 139
radio 267
secondary features 159
Up 177
button mapping 165–180

C

cable configurations 283
cache 147
database issues 151
DBCache 32, 145, 154
difference between NOR and NAND flash memory 147
size 152, 275
web pages 274, 275
Call Log view 105, 107
callouts 187
camera
by device 35, 37, 38
Camera Manager 62–64
Camera Manager known issue 73
CameraV2 Sample Code 331
CameraV3 Sample Code 331
CamLibControl command 62
color-depth mode 63
configuration dialog box 63
Photo Library 59
slider 64
slider notifications 62, 64
CameraLib 64
Caps Lock key 140
capturing images 59, 63
cascading style sheets 201, 202
case sensitivity 190, 215
CCSM
database 75
table support 77
CDMA
_CDMAStatus sample code 329
by device 34, 35
CDMA Library 101
CDMA/1xRTT radio 29
Centro smartphones xiii, 27
media codec formats 56
NetPref Library 75
Phone library description 99
smartphone 117, 120
Telephony APIs 99, 101
Treo smartphones 29
Cell ID 103
cell width attributes 267
Center button 175, 176
Centro smartphones 27–28
CDMA xiii, 27, 103
changes to PmSysgadgetlibCommon.h 279
features not supported 28
GSM xiii, 27, 34, 103
hardware described 34
hardware features 34
HsPhoneGSM.h for Centro GSM smartphones 103
location based services 103
SDIO support 39
software library compatibility 40
What's New xiii
Centro smartphones. *See also* smartphones
certificates 88, 202, 274
changing
databases 151
changing databases 151
character
encoding 117

Graffiti 196
in a URL 133
ISO-8859-1 character set 275
limits for messages 120
limits for SMS messages 116
message text 122
Palm OS character set 275
sets 275
shortcut 237, 248
substitution strings 119
Universal Character Set 277
UTF-8 character formats 275
values, 5-way button 158
virtual 140, 168, 169, 170, 171, 241
check boxes 161, 267
cHTML web pages 199
chunked encoding 86
CIF formats 63
cipher suites (SSL) 87, 88
class-level detection mechanism 292, 293
code compiling 251
Codec Plug-in Manager 50–58, 332
CodecMgrEncodeDecode 72
codecs
Codec Plug-in Manager 50–58, 332
codec wrapping 52
formats by device 56
linked lists 51
media formats 50
MiniGIF sample code 332
MiniImages sample code 332
streaming content 205
Tones library 68
CodeWarrior 246–249
CodeWarrior Development Studio 23
debugging troubleshooting tips 249
loading the binary 247
on-device debugging 248
setting breakpoints 246
simulator debugging 247
color-depth mode 63
communications 201, 273
data communications libraries by device 40, 42
SMS library 116
compatibility logo program 23
compiling code 251
Compose form (email messages) 216
compression
GZIP and Zlib in HTTP library 86
JPEG 56
MPEG 4 56
profile 205
SMS messages 117
connections
data ports and 125
keep-alive 86
loading web pages and 262
networks and 79
phone calls and 102
serial 286
speed, Internet 262
web sites and 85
connectionType key (VersaMail) 215
Console mode, debugging 237, 248
Contacts application 111
content handler 203
Content-Type string 275, 276
cookies 85, 274
Blazer 274
persistent 274
session 274
CPM. *See* Cryptography Provider Manager
cradle 244, 283, 292, 301, 302, 307, 325
createTipsRsc utility 190, 192
creating
databases 151
focus rings 164, 179
folders 131
network profiles 93
plug-ins 220
ring tone management applications 69
sound databases 69
topic titles 182
web pages 261
cryptography 85
Cryptography Provider Manager 87, 88
CSS. *See* cascading style sheets
CtlDrawControl function 164
custom event handlers 163
Customer Service and Support, Palm 19
customizing navigation 162

D

data
accessing 31, 144
always-on session 112
applications 29
caching 147, 154
encoding and decoding 86, 119, 275
encryption 87, 93, 117, 274
ports 125
store 152
streaming 91
data communications. *See* communications
databases

archived ring tones and 69
cache 151
changing 145, 151
committing changes to 151
creating 151
DBCache 154
MMDevice 214
navigation resources and 162
NetPref library enhancements and 75
opening 145, 154
resource 152
SMS messaging and 115
VersaMail applications and 213, 219
DataMgr 151
dataSesstionStatus 112
date picker 273
DBCache 32, 145, 147, 154
Debug mode 237, 248
debugger nubs 236
debugging 235–255
HTML library 97
on-device debugging 235, 248
post-crash debugging 244
Simulator 242
Simulator vs. on-device debugging 235
source code 246
tests 245
tools 23
DebugPrefs application 243–245
default directory 128, 130
default navigation 159, 163, 178
deferSleep 135, 136
deleting
focus rings 179
NBS messages 115, 118
network profiles 93
ring tone databases 70
ring tones 69
descriptor files 206
Designed for Palm Products (DFPP) 20, 23
designing
for mobile devices 262, 267, 269
web pages 261–274
desktop environments 31, 128, 266
DET_DBC parameter 295
DET_PWR_DLY parameter 295
DET_RXTX_DLY parameter 291, 296
DET_SAMP1 parameter 295
DET_SAMP2 parameter 295
detaching peripheral devices 293
detecting
audio 289
audio peripherals 292, 294
device hardware 165
EvDO vs. 1xRTT connections 112
key presses 140
REM sleep mode 137
Developer community forums 23
Developer Guide 17
device discount program 20
Device Loaner Program 23
device loaner program 20
DGND parameter 296, 298
diagnostics mode, Transparency API 125
Dial Pad 105, 108
dialing phone numbers 109, 278
dialog boxes 129, 159, 161
digital certificates 88, 274
digital rights management 207
dimensions
battery gadget 279
Bluetooth gadget 280
icon 127
signal gadget 280
toolbar and scroll bar 262
directional rocker keys. *See* 5-way navigator
directory paths 18, 128, 129
disabling insertion point 180
display engine 133, 134
displaying
application tips 193
certificates 88
email attachments 220
errors 244
formatted help 181
images 269, 271
Keyguard dialog box 140
Open dialog boxes 129
tutorials 193
web pages 212, 267, 272
displays
See also screens
compatibility for Tungsten 31
scrolling multipage 132
specifications for 35, 36, 37
turning on 139
waking up 139
DmCloseDatabase 151, 155
DmNewRecord 151, 155
DmResizeRecord 151, 155
DmSyncDatabase 151, 155
documentation
ACCESS Developer Program 19
feedback 20
Palm Customer Service and Support 19
Palm Developer Network 23

Palm OS SDK 24
dots per inch (DPI) 35, 262
Down button 177
download manager 203, 206, 207
download restrictions 206
downloading
ACCESS (PalmSource) SDK 24
DebugPrefs application 243
files 203, 207, 274
Palm OS SDK 24
PalmDebugger 250
PilRC resource compiler 162
PIM SDK 47
ring tones 203
training images 196
web pages 199, 270
zLib libraries 28, 30
drawing focus rings 164, 179
drawing UI objects 164, 179
drives 153
drop-down menus 267

E

editable fields 159, 160
email
adding to VersaMail folders 213
applications 115, 266
attachments 218, 220
Bluetooth 216
launch commands 216
linking addresses to applications 132
mailto command 266
outgoing 216
plug-ins 213
retrieving 115
saving messages 216
sending 141, 266
sending attachments 220
SMS 216
VersaMail 213–221
emailAddress key (VersaMail) 215
EMBED tag 263, 264, 265
Emulator 23, 24, 247, 251
encoding 86, 117, 119, 275
encoding video for streaming 73
encryption 87, 93, 117, 274
enterprise environments 213
errors 151, 239, 242, 243, 244
EvDO 29, 112
event handlers 163
EvtGetEvent events 139
EvtResetAutoOffTimer events 139
Exchange Manager 28, 30, 127, 128, 203, 217, 220
exgRegEditExtensionID constant 127
ExgRegisterDatatype structures 127
expansion cards 203, 266, 278
Expansion Manager 153, 225
expansion Multi-connector 283–313
Expansion Parts Store 19, 325
expCapabilityHidden constant 153
expCapabilityNonRemovable constant 153
ExpCardInfoType structures 153
external audio output device 289
external dialogs 61
External Hardware Drawings 325

F

factory reset 257
fallback services 75
FAT files 147, 152
fatal errors 239, 243
Favorites
application 127
CustomFav sample code 331
icon 127
launching the Phone application in the Favorites view 110
view 105, 106, 110
fax services 28, 30
feature pointers 154, 252
fields 159, 177, 180
file
allocation tables (FAT) 147, 152
downloading 203, 207, 274
name extensions 128, 130, 131
names 128, 130, 190
opening 127, 128, 129
paths 128, 129
saving 130
selecting 130
storage 153
types 203
uploading 266
volumes 147, 152, 153
File Browser API 127–131
file keyword, integration tag 278
FileBrowserLib68K.h header file 128
FileBrowserLibClose function 128
FileBrowserLibCommon.h 127, 129, 130
FileBrowserLibOpen constant 129
FileBrowserLibOpen function 128
FileBrowserLibParseFileURL function 128
FileBrowserLibShowOpenDialog function 129

FileBrowserLibShowSaveAsDialog function 129, 130, 131
fileType parameter 128, 130
First-chance exception, PalmSim.exe 242
flags 130
  5-way navigator 159
  File Browser 130
  NVFS 153
  SMS header information 122
  UI 75
flash memory 144, 145, 147
FldDrawField function 164
FldGrabFocus function 180
FldReleaseFocus function 180
fnav resources 159
focus
  application focus mode 157
  change events 158
  drawing for Treo 164
  initial 161, 179
  moving focus for Tungsten 161
  navigation 179
  object focus mode 157, 158
  rings 164
  rings, deleting 179
  tapping 159
  Tungsten 161
  UI 179
folders 130, 131, 182
font
  colors 132
  database (VersaMail) 219
  emoticons 132
  graphics 132
  information 219
  library (VersaMail) 213, 219
  picker (VersaMail) 213, 219
form tag 267
formatted help 181
formNavRscType function 162
forms 162, 267
  HTML 267
  modal 163
  nonmodal 161, 163
forums 23
Frequently Asked Questions (FAQs) 19, 23
FrmDispatchEvent function 158, 163
FrmGetFocus function 180
FrmHandleEvent function 163, 179
FrmNav function 166
FrmNavDrawFocusRing function 164, 166
FrmNavGetFocusRingInfo function 164, 166
FrmNavObjectTakeFocus function 166
FrmNavRemoveFocusRing function 164, 166
frmObjectFocusLost events 158, 163
frmObjectFocusTake events 158, 163, 179
frmOpenEvent events 279, 280
FrmSetFocus function 163, 179
FrmSetNavEntry function 159, 160, 178
FrmSetNavOrder function 160, 178
FrmSetNavState function 159
FtrGet function 151, 219
FtrSet function 219
Full-Screen Writing API 196

G

gadgets
  adding 279
  battery 279
  Bluetooth 279
  libraries 279
  overlapping 106, 279
  signal 102, 280
  system gadgets 163, 279
  updating 279
Garnet OS Developer Suite 22, 255
gateways 120
generating symbolic information 247, 248, 251
Getting started 21
GoLCD Manager API 196
goLcdGraffitiMode constant 196
GoLCDSetBounds function 196
goToCreator field 127
GPRS/EDGE radio 27, 29
Graffiti 2
  Blazer 273
  by device 34, 36, 37
  Full-Screen Writing API 196
  FullScreenWriting sample code 331
  Graffiti Shift Indicator (GSI) 140, 196
  writing area 197, 262
graphic element design guidelines 194
graphics 261, 262
  See also images
Gremlins 240
grounding devices 286
GSM
  _GSMStatus sample code 329
  alphabet 117, 119
  Centro smartphones xiii, 27, 34
  connected indicator 102
  devices 117
  HsPhoneGSM.h for Centro 103
  messages 120
  network parameters 75
  Phone library description 99

Telephony APIs 101
updating indicator 102
GSM/GPRS network parameters 75

H

HandheldFriendly
keyword 277
tag 263
handhelds 31, 37, 283
See also Tungsten handhelds; Zire handhelds
Handspring extensions 180
hard reset 145, 147, 230, 242
hardware 34, 36, 165
Centro smartphones 34
described 34
detecting device 165
external drawings 19, 325
interfacing with smart peripherals 308
LifeDrive mobile manager 36
Palm T|X handhelds 36
Palm Z22 37
peripheral detection 292
smartphone audio 319
Treo smartphones 34
Tungsten 36
Zire handhelds 37
hash algorithms 88
header files 28, 30, 31
header information, SMS 121, 122
headphones 289
headsets 289, 292, 294, 319
heap 144
corruption 254
DBCache 147
dumps, PalmDebugger 254
heap ID 152
help, formatted 181
Helper API, MMS 141
helperAppID parameter 141
HelperNotifyExecuteType structures 141, 142
HelperServiceMMSDetailsType structures 142
HelperServiceMMSObjectType structures 142
high-resolution devices 219, 262
home page (web) 201
Home/Roaming network configurations 75
horizontal header images, Blazer 271
HotSync
application 79
Interrupt pin 288
operations 75, 219, 288
HOTSYNC parameter 296, 298
HS_INL parameter 288
HS_OC parameter 288
HsAppLaunchCmd.h header file 107, 111
hsAttrDisplayOn events 137
HsAttrGet function 137
hsAttrKeyboardLocked events 139, 140
HsAttrSet function 139
hsChrKeyboardLock events 139
HsCreators.h header file 107
hsFtrIDNavigationSupported events 165, 166
HsNav function 166
HsNavDrawFocusRing function 164, 166, 179
HsNavGetFocusRingInfo function 164, 166
HsNavObjectTakeFocus function 166, 179
HsNavRemoveFocusRing function 164, 166, 179
hsNotifyRemSleepEvent 136
hsNotifyRemSleepRequestEvent 136
hsNotifyRemSleepRequestEvent events 136
hsNotifySetInitialFldStateEvent events 180
HsPhone.h header file 99
HsPhoneGSM.h 103
HTML
cHTML web pages 199
encoding 275
extending 277
files 182
forms 267
HtmlLibTest68K sample code 331
library 95, 97
parser 133
renderer 97
specifications 272
tag 273
unsupported elements for Blazer 272
web pages 199
xHTML web pages 199
HTTP
_HTTPLibTest sample code 329
client profiles 85
headers 274, 276
HTTPLibTest sample code 91
library 85–91
link 73
protocol 85
requirements for third-party 85
stack 201
httpErrorLibraryAlreadyOpen 87
HTTPLibTest project 91
HTTPS connections 85
hypertext links 267
HyperText Transfer Protocol. *See* HTTP

I

icons 127, 128
  dimensions 127
  labeling 127
  lock 274
image
  files 127, 128, 272
  formats 272
  horizontal header for Blazer 271
  resolution 63
ImageLib 65
images
  adding header in Blazer 271
  adding to training tools 182, 187, 194
  adding to web pages 267, 269
  as text 270
  Camera Manager for 62, 63
  capturing 59
  designing web pages and 261, 262
  displaying 269, 271
  downloading training 196
  flagging for download 206
  icons and 127
  JPEG 206
  Mini Image sample code 332
  mobile devices and 269
  previewing 63
  refreshing 275
  saving 206
  scrolling through 271
  sending 141
  spacers 267, 271
imaging
  applications 141
  multimedia features 58–64
  processor 269
incoming messages 116, 117, 121
incoming voice calls for Treo 700p 113
incomingPort key (VersaMail) 215
incomingServer key (VersaMail) 215
indented lists 189
index.html file 182
INet library 28, 30, 31
InitFontUI method 219
initial focus 161, 179
inking 196
input
  audio 322
  File Browser API 129
  Graffiti 2 input areas 196
  input areas on Tungsten 165
  Option and Shift keys 140
  text input, Blazer 267
insertion points for fields 180
interaction mode 157
interface connectors 35, 37, 38
internal drives 153
internal file volumes 153
international support for Blazer 275–276
Internet connection speed, Blazer 262
Internet Service Providers 116
Internet, Blazer 199
Interrupt pin (HotSync) 288
IOTA application 77
ISO-8859-1 character set 275
ISPs 116
IUSB_VBUS_H parameter 286

J

JavaScript 182, 201, 261, 266
JPEG 50, 56, 206, 272, 277
JPEGLib, deprecated 64

K

keep-alive connections 86
key
  apn (VersaMail) 215
  Caps Lock 140
  Center button event 175
  events 135, 139, 158
  exchange 87
  Option key 140
  Option Lock 140
  presses 139, 140, 159, 161
keyboards 34, 36, 37, 165
keyDown events 158, 175
Keyguard 139–140
keyHold events 158
keystroke equivalents 241
keyUp events 158
kFileBrowserLibFlagNoFiles constant 130
kFileBrowserLibFlagNoFolders constant 130
kFileBrowserLibFlagNoNewFolder constant 130
kFileBrowserLibFlagOneVolume constant 130
kFileBrowserLibFlagPromptOverwrite constant 130
kFileBrowserLibFlagRequireExtension constant 130
kFrmNavObjectFlagsSkip constant 159
Knowledge Base 19, 23
Known issue with Camera Manager and streaming on Treo 700p smartphones 73
kPmConnectorClass Notifications 44

kSysAppLaunchCmdOpenFile constant 128

L

labeling icons 127
large icons 127
launch codes (SMS messages) 116
launch commands (VersaMail) 216, 218, 220
launching
  Blazer web browser 193
  Contacts application 111
  email applications 266
  Network Preference dialog from applications 79
  Palm web browser (Blazer) 211, 212
  Phone application 107–110
  VersaMail application 218
LBS 103
LCD on/off notification 137
LCD Overlay 60–62
Legacy profiles 79
lessons
  creating 182, 186
  naming 184
libraries
  compatibility specifications for 40
  data communication libraries by device 40, 42
  downloading 28, 30
  loading 75, 129
  multimedia described 49
  network services and 93
  not supported 28, 30, 31
  security libraries 274
  system gadgets and 279
  telephony 99
  Treo smartphones and 30
  web browsing and 85, 86, 88
library 68
LifeDrive mobile manager
  features 32
  hardware described 36
  performance issues 32
LifeDrive mobile managers
  features not supported 32
Link walking 97
linked list of codecs 51
linked list of format pairs 51
linking examples (NetPref) 75
linking to web pages 212, 267
links
  in Blazer 267
  in streaming content 264
lists 189, 267
loading
  File Browser library 129
  NetPref library 75
  the binary in CodeWarrior 247
  VersaMail Font library 219
  web pages 262
location based services 103
lock icon 274
logins (networks) 77
logo compatibility program 20
logo program (DFPP) 23
losing power 151
low-resolution devices 219
LstDrawList function 164
Lz77Mgr.h header file 28, 30, 31

M

MailAddRecordParamsType structures 218
MailAddRecordsParamsTypePlus structures 218
mailto command 266
main menu, tutorial 194
main views 163
marketing programs 20
markup languages 201
MD5 hash algorithm 88
Media File Formats 205
media formats (codec) 50
MemHeapFreeBytes function 152
memory 144, 203
memory architecture, NVFS 144–150
memory, by device 34, 36, 37
menu page 182
menus 182, 194, 267
message database 115, 118, 119, 121
messages 116
  automatically reassembling 120
  character limits for 120, 121
  composing email 216, 218
  encoding SMS messages 119
  events 117
  incoming 121
  MMS devices and 141
  NBS 115, 121
  outgoing 121
  receiving incoming 117
  segmenting 120
  sending 115
  SMS 115
  storing 121
  updating list of 118
  verifying integrity of 88
  voicemail and 102

Messaging application 141
MMS devices and 141
SMS devices and 115, 120
messaging services 115
Metrowerks CodeWarrior 246–249
microSD xiii, 27
MIDI 68
files 263
ring tone 203
Ring Tone DB 69
sound formats 69
TonesLibAddMidiTone 71
MIME type 128, 130, 206, 220
MMDevice database 213, 214
MMDevice record keys 215
MMPRO_ADD_MESSAGE_WITH_ATTACHMENT command 218
MMPRO_LAUNCH_CODE command 218
MMPRO_PLUGIN_EXTENDED_QUERY_LAUNCHCODE command 221
MMPRO_PLUGIN_GET_INFO_LAUNCHCODE command 220
MMPRO_PLUGIN_QUERY_LAUNCHCODE command 220
MMPRO_PLUGIN_RECEIVE_LAUNCHCODE command 221
MMPRO_PLUGIN_SEND_LAUNCHCODE command 221
MMProLaunchStruct structures 218
MMProLaunchStruct2 structures 218
MMS
_MMSCounter sample code 329
_MMSReceiver sample code 330
helper function APIs 141
Helper service providers 141
Messaging application 141
MMSHelperCommon.h header file 141
MMSReceiver 141
MMSSender 141
send requests 142
service providers 142
mobile devices 29, 115, 262
mobile originated messages 116
mobile terminated messages 116
mobile.palm.com 262
modal forms 159, 161, 163
modems 125
MP3
Codec Plug-in Manager 50, 57
example of creating ringtones 71
headset jack specifications 315, 322, 324
MiniMP3 sample code 332
Ringtone database 70
Sound File Stream library 66
streaming content 205
support for Zire 33
Tones library 68
Multi-connector 283–313
Multi-connector events 44
multiline fields 177
multimedia 49–72
features on handheld devices 42
features on Palm smartphones 40
files 207
libraries 49
video playback 72
Multimedia Messaging Service. *See* MMS messages
multipage displays 132
Multipurpose Internet Mail Extensions. *See* MIME type
multisegmented messages 116, 118, 120
music playback 29, 30, 33, 320

N

naming icons 127
NAND flash memory 144–157
Narrow Band Socket. *See* NBS messages
navigation 160
(default) 159, 163
buttons 241
customizing 162
events 158, 162, 163
focus 179
information 178
macros 177
order 178
order of UI objects 160
resources 162, 178
Navigational API 165
navigator buttons 34, 36, 37, 140, 158
navigator, 5-way. *See* 5-way navigator
NBS messages 115, 118, 121
NetLib
library interface 86
wrapper function callbacks 86
NetMaster library 77
NetPref library 75, 77
NetPrefUtils package 75
NetServices library 93
network 75, 77, 125
connections 102
preference database 75
profiles 77, 79, 93
Network Preference
database 77, 79

panel 77–79
New Contact window 111
New Folder button 131
Newsletter 23
Non Volatile File System. *See* NVFS
nonmodal forms 159, 161, 163
nonremovable volumes 153
NOR flash memory 144, 147
normal sleep deferral 135
Northstar Systems 19, 325
notification loop 103
numbered lists 189
Nutshell installer 207
NVFS
API 144–155
architecture 144–150
checking for 151
guidelines for developing 151

O

object focus mode 157, 158
OBJECT tag (Blazer) 264
On-device debugging using CodeWarrior 248
Open dialog box 128, 129, 130
Open URL dialog box 278
opening
databases 145, 154
files 127, 128, 129
Operator's Name indicator 102
Optimized mode (Blazer) 262
Option keys 140, 159
Option Lock key 140
optionKeyMask events 159
organizer 29
Outbox (VersaMail) 216
outgoing
email 216, 218
SMS messages 121
outgoingPort key (VersaMail) 215
outgoingServer key (VersaMail) 215
output
redirecting audio 319, 320
Treo smartphone audio subsystems and 322
Overlapping gadgets 106, 279
overriding default navigation 163

P

PACE debugger nub 237, 245
page keys 158
Page Properties dialog box 263
page scrolling 177
paging 177
Palm Customer Service and Support 19, 105, 106
Palm keyword 277
Palm libraries 40
Palm Multi-connector. *See* Multi-connector
Palm OS
alphabet 119
character set 275
Developer Suite (PODS) 22, 255
downloading SDK 24
Notification Manager 226
organizer 29
Palm OS Programmer's API Reference 19
Palm OS Programmer's Companion 19
security libraries 274
simulator 251
Software Development Kit (SDK) 17, 21, 224
software for handhelds 42
software for Palm smartphones 40
system Helper API 141
Tone DB 69
Palm OS Developer Suite (PODS) 22, 255
Palm Ring Tone manager 206
Palm Software Development Kit (SDK) 24
Palm T|X handhelds
hardware described 36
software specifications for 42
Palm web browser (Blazer) 85, 199, 201, 212
Palm Z22 organizer
hardware described 37
software specifications for 42
Palmcall keyword 277
PalmCameraSlider.h header file 62, 64
PalmCodecFormat.h header file 50
PalmDebugger 250–254
commands 252
heap dumps 254
keyboard shortcuts 254
PalmGoLCD.h header file 196
PalmNetServices.h header file 93
palmNULLCodecID 50
palmOneCamera.h header file 62
palmOneCameraCommon.h header file 62
palmOneCodecFormat.h header file 50
palmOneCodecPluginMgr.h header file 50
palmOneCreators.h header file 111, 137
palmOneNavigator.h header file 177
palmOnePhoto.h header file 59, 60
palmOneSGHiResFonts.pdb 219
palmOneSGLowResFonts.pdb 219
PalmRC utility 192

PalmSGFontLib value 219
PalmSim.exe 242
PalmSource. *See* ACCESS
PalmVMLaunch.h header file 218
PalmVMPlugin.h header file 220
PalmWiFiCommon.h header file 93
parameters
  media structures and 50
  Multi-connector IO parameters 287
  NetPref library enhancements and 77
parsing engine 133
parsing file URLs 128
partitions 147
pass-through peripherals 284
password key (VersaMail) 215
PDB files 207, 213
pen events 196
performance
  Blazer 274
  NVFS 147
peripheral
  audio 289
  detaching devices 293
  detecting audio 292
  detecting audio for Tungsten handhelds 292, 294
  detection mechanisms 290
  detection timing specifications 295
  devices 288, 291, 293
  hardware detection 292
  interfacing with smart peripherals 308
  pass-through 284
  power devices 288
  powering up 292
  requirements for Palm smartphones 290
  requirements for Tungsten handhelds 290
  SDIO creation 229
  serial detection mechanisms 292
  serial devices 292
  serial peripheral devices 291
  timing specifications 294
peripherals 19, 325
persistent connections 79
persistent cookies 274
PHL_A_DET parameter 291
PHL_A_DET_HS parameter 291
PHL_A_DET_NHS parameter 291
PHL_LOAD parameter 290
PHL_RX_DOWN parameter 290
PHL_RX_OUTH parameter 290
PHL_RX_OUTL parameter 290
PHL_RXTX_INH parameter 290
PHL_RXTX_INL parameter 290
PHL_SER_DLY parameter 291
PHL_TX_DOWN parameter 290
PhnDataServiceType 112
phnEvtDataSessionStatus 112
phnEvtDisconnectConf 112
phnEvtDisconnectInd 112
phnEvtError events 102
phnEvtIndication events 102
phnEvtMessageInd events 118
phnEvtRegistration events 102
phnEvtSegmentInd events 118, 120
phnEvtStartIncomingCall 112
phnEvtVoiceMail events 102
PhnLib notifications 103
PhnLibBoxInformation function 102
PhnLibCurrentOperator function 102
PhnLibGetMMSUAString() 104
PhnLibRegister 103
PhnLibRegister function 117
PhnLibRegistered function 102
PhnMsgBoxDataType 104
phone
  calls 99, 102
  events 102
  libraries 28, 30
Phone application 107–110
phone numbers
  dialing 109, 278
  linking to applications 132
  prefilling for Dial Pad 108
phoneAppLaunchCmdViewHistory command 107
phoneAppLaunchCmdViewSpeed command 110
phoneto keyword 278
Photos API 59
pictures. *See* images
PilRC resource compiler 162
PIM SDK 47
PIM structures 47
pinout specifications (multi-connector) 284
pixels 35, 262
Play button 264, 266
playing
  ring tones 206
  streaming content 264
Plug-in Manager 220
plug-ins 213, 220, 261
PmPalmOSNVFS.h header file 151, 152, 153
PmSndStreamPause 67
PmSndStreamResume 67
PmSysGadgetBtStatusTypeEnum 279
PmSysGadgetLibrary header 279

PmSysGadgetStatusGadgetTypeSet
structures 279, 280
PODS. *See* Palm OS Developer Suite
point-to-point services 116
pop-up
lists 178
triggers 159
windows 61
ports 125, 286
POUT parameters 288
power
button 135, 139
peripherals 288
power loss 151
POWER_OUT parameter 292, 295, 296, 298
poweredOnKeyMask bit 139
powering up peripherals 292
PRC
applications 207
Creator IDs 53
files 53, 192, 207
PRC-Tools 22
previewing images 63
private volume 147, 152, 153
processors 34, 36, 37
product lines xiii, 27
proxies 85
proxy servers 200
punctuation-mode indicator 196

Q

QCELP 68
QCIF formats 63

R

radio 27, 29, 125
APIs 101
Bluetooth radio status 279
buttons 267
CDMA/1xRTT 29
GPRS/EDGE 27, 29
modem 125
RAM 144
RC4 algorithm 274
reboots 147
reformatting
tables (Blazer) 271
web pages 262, 263
refreshing web pages 274, 275
registering
applications 220
camera slider notifications 64
content handlers 203
icon file types 127
SMS service 117
ZIP files 207
Release Simulator 239
REM sleep
API 135
mode 136–139
request notifications 136, 138
removing
focus rings 179
NBS messages 115, 118
network profiles 93
ring tone databases 70
ring tones 69
rendering engine 134
replyTo key (VersaMail) 215
resets 147, 256–257
resetting auto-off timer 139
resolution 35, 63, 219, 262
resource databases 152
resources 159, 162
rich text formats 132
ring tone 68
archived 69
database 69
downloading 203
management 69
management applications 69
MIDI 203
Palm Ring Tone manager 206
removing databases 70
Ring Tone DB 69
tools 70
roaming 75
rootMailbox key (VersaMail) 215
RPHL_SER_DET parameter 291
RPHL_SER_NDET parameter 291
RSA algorithm 87
RSA-based key exchange 274
RTSP 73
RX_OC parameter 287
RXD parameter 296, 298
RXTX_INH parameter 287
RXTX_INL parameter 287

S

Sample Code 329–333
_CDMAStatus 329
_GSMStatus 329
_HelperReceiver 329
_HelperSender 329
_HTTPLibTest 329

_Keyguard 329
_LaunchCommands 329
_MMSCounter 329
_MMSReceiver 330
_PalmPhotoCapturePreview 330
_PhoneEventReceiver 330
_PhoneInfo 330
_RadioSample 330
_REMTimer 330
_RingTone 330
_SimpleSMS 330
_STETest 330
AddEmailtoVersaMail 330
BluetoothMgr 330
BtClientServer 330
CameraV2 331
CameraV3 331
Conduit 331
CustomFav 331
Extended Sound 331
FileBrowserSample 331
FiveWayV2 331
FotoFinish 331
FotoFinish2 331
FullScreenWriting 331
HtmlLibTest68K 331
HTTPLibTest 91
IRCommunication 331
MemoryInfo 332
MiniADPCM 332
MiniBuzzer 332
MiniGIF 332
MiniImage 332
MiniMP3 332
MiniMPEG4 332
NavExample 332
NetSelector 332
NVFSTest 332
PmConnectorTest 332
Pushdemo 332
SerialCommunication 332
StatusBar 333
TickStock 333
Treo650PowerOut 333
TungstenT5FlowControl 333
TungstenT5SwitchAudio 333
TungstenT5SwitchAudioAlt 333
VersaMailAccConfig 333
VersaMailAttchmntPlugin 333
VersaMailFontApp 333
sample plug-in 220
Save As dialog box 128, 130
saving
email messages 216
files 130
images 206
screen resolution 35, 262
screens
See also displays
mobile devices and 262
sleep mode and 139
specifications for 35, 36, 37
web pages and 201, 202
Zire handhelds and 33
scroll bars 132, 271
scrolling images 271
scrolling multipage displays 132
SD card xiii, 39, 59, 71, 97, 128, 223
SD slot volumes 153
SDET_ID_TIMEOUT parameter 296
SDIO
application guidelines 226
applications 223
Auto Run 229
card initialization 230
CSA (Code Storage Area) 231
detecting card insertion and removal 228
interrupt handling 228
peripheral creation 229
slot driver 226, 230
software architecture 224
specifications 223
support by device 39
SDK. *See* Software Development Kit
SDRAM memory devices 144
Secure Sockets Layers. *See* SSL
security features (web sites) 202
security libraries 274
segmentation
binary 120
information 122
schemes (messages) 120
text-based 120
selector triggers 159
Send command 216
send URL send scheme 216
sending
email 141
email attachments 220
email mailto 266
pictures 141
text messages 115, 116
serial connections 286
serial drivers 125
serial peripheral
detection mechanisms 284, 290, 292

detection, Smart 308–313
devices 291, 292
serial pins interface 287
serverType key (VersaMail) 215
service
centers 116
MMS 141
Palm Customer Service and Support 105
providers 116, 200
session cookies 274
SET-COOKIE header 274
SHA1 hash algorithm 88
shared content library 182
SHIELD parameter 296, 297, 298
shield pins (multi-connector) 286
shift
indicator 196
shift key 140, 159
shift-lock indicator 196
states 180
shiftKeyMask events 159
Short Messaging Service. *See* SMS
Signal gadget 102, 280
Simulator 235, 238
debugging tips 242
debugging using CodeWarrior 247
features 239
Release Simulator 239
vs. on-device debugging 235
sleep-deferral notifications 135, 136, 139
small icons 127
Smart Text Engine API 132, 133
See also STE
smartphones
accidentally turning on 139
adding system gadgets for 279, 280
audio hardware for 319
audio software for 320
audio subsystems for 321, 324
battery gadget 279
button mapping 165–180
Center button behavior 175
debugging on 248, 249
drawing focus for 164
EvDO 112
Handspring extensions for 180
launching Contacts application for 111
launching Phone application for 107
MMS helper functions API 141
Multi-connector IO parameters for 287
navigating through 165, 166, 177
NVFS API 144
option and shift key APIs 141
Option plus App key mapping 173
page scrolling 177
Palm OS simulator for 251
peripheral requirements for 290
putting to sleep 135, 137
See also Treo or Centro smartphones
signal gadget 280
smart peripherals 308
SMS library 117
software specifications for 40
streaming audio and video 73
text messaging service for 116
Tips and Tutorials 181
waking up 139, 140
SMS
database 115, 118, 119, 121
email option 216
library 28, 30, 115–121
messages 115, 117
messaging application 115, 120
Phone library description 99
service, registering 117
technology 116
SmsLib.h header file 28, 30
SndFileStream 66, 67
SndStream API 66
socket pool 86
soft reset 147, 230, 242, 245, 256
software compatibility by device 40
Software Development Kit (SDK) 17, 21, 24
sound API 66
Sound Manager 66, 320
sounds 322
See also audio; ring tones
source code
compiling 251
debugging 246
spacer images 267, 271
SPKR_L parameter 296, 298
SPKR_R parameter 297, 298
SSL
cipher suites 87
connections 85
library 85, 87
STE
engines 133
library 132
STETest.zip 132
storage 31, 121, 144, 153
store, data 152
streaming
audio and video 73, 199, 204, 264
content 205

data 91
encoding video 73
links in content 264
MP3 205
playing content 264
stub applications 193
style guide 261–280
style sheets 182, 201, 202
See also cascading style sheets
substitution strings (messages) 119
svcCFACmdQuickEdit parameter 79
symbolic information 247, 248, 251
sysAppLaunchCmdAddRecord command 218
sysAppLaunchCmdExgReceiveData launch command 127, 128
sysAppLaunchCmdGoTo launch command 127
sysAppLaunchCmdPanelCalledFromApp command 79
sysFtrNumFiveWayNavVersion constant 165
sysFtrNumUIHardwareFlags constant 165
sysFtrNumUIHardwareHard5Way constant 165
sysFtrNumUIHardwareHasKbd constant 165
sysFtrNumUIHardwareHasThumbWheel constant 165
sysFtrNumUIHardwareHasThumbWheelBack constant 165
SysHandleEvent command 135, 139
SysLibFind function 219
sysNotifyExternalDialogClosedEvent 61
sysNotifyExternalDialogOpenedEvent 61
system extensions 41, 43, 125–197
system gadgets 163, 279
systemMgr.h header file 218
SysUISwitch function 217

T

tab order 157, 159, 160
table-rendering engine 267
tables
cell width attributes 267
navigating through 163, 177
reformatting 271
web pages and 262, 267
tags 190, 201
taking pictures 62
tapping 159, 196, 206
TCP sockets 85
tel keyword 278
telephony 99–113
CDMA Library 101
functions 100
GSM APIs 101
header files 28, 30, 100
libraries 99, 100
Telephony Manager 28, 30
testing web pages 275
tethered mode 125
text 270
fields 132, 159, 177, 180
images as 270
input 267
input dialog box 267
messages 115, 116
rich text formats 132
text-based segmentation (messages) 120
text-based tables 267
text messages
See also messages
thumb wheels 165
time picker 273
tips
accessing 181
CodeWarrior debugging tips 249
debugging tips 242
displaying application tips 193
files 182, 190
Tips and Tutorials 181
tutorial 182
utilities 192
TipsTutorialsUtil.zip 192
title key (VersaMail) 215
Tones library 68
TonesLib.h header file 68
TonesLibAddMidiTone 71
TonesLibToneClose() 70
TonesLibToneCreate() 70
TonesLibToneWrite() 70
toolbar 262, 274
topic (for Tutorial, defined) 181
touch screens. *See* screens
training tools 181, 182
Transparency API 125
Treo 700p smartphone ROW carrier release 30
Treo smartphones
applications 29
Camera APIs and 62
CDMA 29
features described 29
features not supported 30
hardware described 34
Treo smartphones. *See also* smartphones
trusted root certificates 88
Tungsten handhelds
accessing file volumes for 153

detecting audio 292, 294
drawing focus for 164
enabling network services for 93
features described 31
features not supported 31
flash memory for 144
hardware described 36
moving focus for 161
multi-connector IO parameters for 287
navigating through 158, 165, 177
paging through 177
peripheral requirements for 290
software specifications for 42
TungstenT5FlowControl sample code 333
TungstenT5SwitchAudio sample code 333
TungstenT5SwitchAudioAlt sample code 333
tutorial
accessing tutorials 181
design guidelines 194
displaying 193
files 181
TipsTutorialsUtil.zip 192
utilities 192
TX_OC parameter 287
TX_OUTH parameter 287
TX_OUTL parameter 287
TXD parameter 296, 298
typographical conventions 18

U

UCS 277
UDHs (user data headers) 120
UI flags 75
UI objects
cycling through 161, 162
drawing 164, 179
giving focus to 179
navigation order for 160
unlinking examples (NetPref) 75
unordered message segments 120
unsupported HTML elements 272
Up button 177
updating
GSM Connected indicator 102
list of messages 118
system gadgets 279, 280
web pages 275
uploading files 266
URL 127, 132
bar (Blazer) 262, 274, 278
characters 133
linking to web pages 211
Open URL dialog box 278
parsing file URLs 128
scheme 205
USB cables and connectors 286
USB VBUS devices 284, 286
USB_DN parameter 296, 298
USB_DP parameter 296, 298
USB_ID parameter 296, 298
USB_VBUS_CHG parameter 286
USB_VBUS_L parameter 286
useEncryptedPassword key (VersaMail) 216
useEsmtp key (VersaMail) 216
user agent string 273
user data headers 120
User-Agent string 273
userName key (VersaMail) 216
UTF-8 character formats 275
utilities 192

V

VBUS parameter 296, 298
vchrAutoOff events 135, 139
vchrHardPower events 135
vchrHardRockerCenter events 176
vchrPowerOff events 135
vchrResumeSleep events 135
vchrRockerCenter events 175, 176
VDOCK parameter 296, 298
VDOCK_RTN parameter 296, 298
VersaMail 213
account configurations 213, 214
AddEmailtoVersaMail sample code 330
adding to folders 213
apn key 215
application 216, 220
application API 213
applications and databases 213
Attachment Plug-ins API 220
connection type 215
device APIs 213
emailAddress key 215
Font library 213, 219
Font Picker 219
incomingPort key 215
incomingServer key 215
launch commands 216, 218
launching application 218
option (email) 216
outbox 216
outgoingPort key 215
outgoingServerkey 215
password key 215
Plug-in Manager 220

replyTo key 215
rootMailbox key 215
serverType key 215
title key 215
useEncryptedPassword key 216
useEsmtp key 216
userName key 216
VersaMailAccConfig sample code 333
VersaMailAttchmntPlugin sample code 333
VersaMailFontApp sample code 333
vertical order 160
VFS Manager 225
vfsErrVolumeFull 155
vfsIncludePrivateVolumes constant 152, 153
vfsInvalidVolRef constant 128, 129
VFSRegisterDefaultDirectory function 130
vfsVolumeAttrNonRemovable constant 153
VFSVolumeEnumerate function 152
video
Camera Manager 62, 63
codecs 205
encoding 73
playback 72
streaming 199
streaming embedded in Blazer 264
views 163
Virtual File System (VFS) Manager 225
voice recordings 66
voicemail
interrupting data calls 78
voicemail indicator 102
voicemail messages 102
volRefNum constant 128, 129
volume
applications switching 128
partitions 147
private 147, 152
reference number 152
volumes, accessing internal file 152
VPHL_SER_DET parameter 291

W

WAP HTTP client profiles 85
WAP stack 201
warm reset 256
WAV 68
Web Browser API 199
web browsers 88, 200, 211, 273
See also Blazer web browser; Palm web browser
web browsing 29, 201, 269
web pages
accessing 201, 211, 275
cache 274, 275
creating 261
designing 261–274
displaying 212, 267, 272
downloading 199, 270
linking to 211, 212, 267
loading 262
reformatting 262
refreshing 274, 275
tables 262
testing 275
unsupported elements and 272
updating 275
viewing titles of 263
web sites 202, 261, 275
Wi-Fi
LifeDrive 32
Net Services API 93
Palm handhelds, by device 36
panel 93
software libraries, by device 40, 42
windows
dialog boxes 159
pop-up 61
WinModal function 159
wireless
data applications 29
features by device 35, 36, 37
fidelity functionality 93
service providers 200, 207
WML
forms 267
web pages 199, 201, 276
wrapper file formats 205
wrapper functions 128
writing areas 196
writing. *See* Graffiti 2

X

xHTML web pages 199
XML documents 182, 190
XML tags 190
XRD files 192

Y

YUV color model 61

Z

ZIP files 207
Zire handhelds
camera slider notification for 64

features not supported 33
hardware described 37
navigating through 177
zLib library 28, 30

This page intentionally blank