Developer's Notes

MIDP for Palm OS, Version 1.0 FCS

Java™ 2 Platform, Micro Edition

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
U.S.A. 650-960-1300

October, 2001

Please
Recycle

Adobe PostScript

Contents

Figures

Tables

Preface

The *Developer's Notes* for MIDP for Palm OS provides product-specific information to developers of Java applications for Palm OS devices.

MIDP for Palm OS is a Java runtime environment. Although the release targets end-users, not developers, it does contain tools that developers of Java applications for the MIDP environment can use when they write and test their applications. This guide contains information on those tools and on the features and characteristics of MIDP for Palm OS that will be useful for developers.

Before You Read This Book

To fully use the information in this document, you must have thorough knowledge of the topics discussed in these books:

- *ReadMe*
- *User's Guide*
- *MIDP Specification at* `http://java.sun.com/products/midp/`

How This Book Is Organized

Chapter 1 discusses the MIDP for Palm OS implementation of the MIDP specification.

Chapter 2 covers the protocols included with MIDP for Palm OS.

Chapter 3 tells you how to convert Spotlet applications to Java applications written for the MIDP environment.

Chapter 4 shows you how to use `MakeMIDPApp`, a command-line tool that changes a Java technology-based application written for MIDP into a PRC file.

Chapter 5 discusses the application, `Developer.prc`, that you can use to debug your Java applications written for the MIDP environment.

Appendix A contains the frequently-asked questions list (FAQ) for MIDP for Palm OS; it also has a pointer to the on-line version of the FAQ, which could include more recent information.

Appendix B has a list of error messages from the `MakeMIDPApp` tool.

Using OS Commands

This document may not contain information on basic operating-system commands and procedures such as changing directories or synchronizing your Palm OS device with your desktop system. See the software documentation that you received with your desktop system or Palm OS device for this information.

Typographic Conventions

Typeface	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`
`AaBbCc123`	What you type, when contrasted with on-screen computer output	`% su` `Password:`
AaBbCc123	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the *User's Guide*. These are called *class* options. You *must* be superuser to do this.
	Command-line variable; replace with a real name or value	To delete a file, type `rm` *filename*.

Related Documentation

MIDP for Palm OS and its documentation bundle include the following documentation:

Title	Description
User's Guide	Describes how to install and run Java applications written for MIDP on a Palm OS device.
Developer's Notes	Discusses issues important to the designers and developers of Java applications for Palm OS devices. It is *not* a full programming guide.
Example Java Applications	Shows screen shots of, and briefly describes, the samples that are included with MIDP for Palm OS.
ReadMe	Includes important last-minute information, including known bugs.

Accessing Sun Documentation Online

The `docs.sun.com`[sm] web site enables you to access Sun technical documentation on the Web. You can browse the `docs.sun.com` archive or search for a specific book title or subject at:

`http://docs.sun.com`

Ordering Sun Documentation

Fatbrain.com, an Internet professional bookstore, stocks select product documentation from Sun Microsystems, Inc.

For a list of documents and how to order them, visit the Sun Documentation Center on Fatbrain.com at:

`http://www1.fatbrain.com/documentation/sun`

Sun Welcomes Your Comments

We are interested in improving our documentation and welcome your comments and suggestions. You can email your comments to us at:

`midp4palm-comments@sun.com`

CHAPTER 1

MIDP for Palm OS and the MIDP Specification

MIDP for Palm OS is a compliant implementation of the *Mobile Information Device Profile (JSR-37) JCP Specification: Java 2™ Platform, Micro Edition, 1.0a* (from here on called the MIDP specification). There are some things, however, that the specification does not fully specify. These details it leaves to individual implementations. This chapter covers those implementation-specific details of MIDP for Palm OS. It contains the sections:

- Abstract Commands
- Mapping MIDP User-Interface Components to the Palm OS
- Color Support

1.1 Abstract Commands

Abstract commands enable a developer to specify application actions on a screen by screen basis, without specifying the user-interface that enables those actions to be carried out. MIDP provides abstract commands, instead of lower-level UI components that lead to command actions, such as buttons, so that MIDlets can run successfully on device hardware with different configurations. Some devices have soft buttons, others use menus or other UI mechanisms; abstract commands allow MIDlets to use the same code to run on all those devices.

MIDP for Palm OS shows all abstract commands in the menus, which the user accesses by tapping on the Palm OS device's Menu button (). If there are too many abstract commands of a particular type to fit on a single menu, then that command type is changed to a menu item, More..., that opens a native dialog with a list of commands that the user can scroll through.

The MIDP for Palm OS runtime maps abstract command types onto menus using the following organization:

- Actions Menu– Contains commands of type `ITEM` followed by type `SCREEN`. (As a reminder of usage guidelines, `ITEM` commands should affect only the selected item, while `SCREEN` commands should affect the entire screen.) The menu always contains one static menu item at the top, About Java™ HQ..., to accommodate a known Palm OS bug.
- Go Menu – Contains commands of type `BACK`, `OK`, `CANCEL`, `STOP`, and `EXIT`, in that order. The menu always contains one static menu item at the top, Beam App....
- Options Menu – Contains commands of type `HELP`. The menu always contains one static menu item at the top, Java Preferences.... To ensure that your application's About box shows up in the Options menu (the standard Palm OS place to access an About box) make the abstract command to show the About box of type `HELP`.

The menus use priority-order to organize multiple commands of a single abstract-command type. If a command type is not used by an application, then it is not shown in the menu.

MIDP for Palm OS also supports menu shortcuts. It determines the shortcut based on the first unique character of the abstract commands' labels, and indicates the shortcut in the standard Palm OS style.

In addition to menus, MIDP for Palm OS maps abstract commands to a single row of soft buttons at the bottom of the screen, leaving space for displaying Palm OS system indicators, such as the `SHIFT` graffiti mode. It maps the highest priority abstract commands of the following types in the following order: `BACK`, `OK`, `CANCEL`, `STOP`, `SCREEN`, and `ITEM`. It does not map the `EXIT` and `HELP` types. Due to space limitations, the runtime will typically not be able to provide a button for every command type in the list.

Note that abstract commands of type `EXIT` are *not* called when the user taps the Palm OS device's Home button (). When the user taps the Home button, MIDP for Palm OS calls the MIDlet's `destroyApp` method.

Tip – For usability reasons, avoid putting long-running procedures in your `destroyApp` methods.

1.2 Mapping MIDP User-Interface Components to the Palm OS

This section discusses how MIDP for Palm OS implements the user-interface components of MIDP to the Palm OS environment. It contains the topics:

- Choice
- Text Fields
- Implicit Lists
- Ticker Tapes
- Forms

1.2.1 Choice

The MIDP specification says that images are an optional part of choices in choice groups. MIDP for Palm OS follows the Palm OS user interface style, which does not support images. The following figure shows a choice in MIDP for Palm OS:

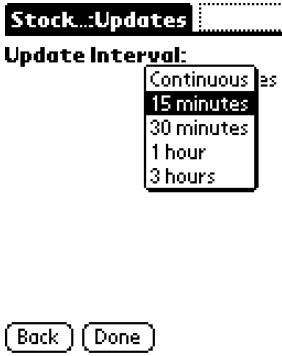

FIGURE 1 A Choice User-Interface Component

1.2.2 Text Fields

If there is more text in a text field than will fit on a screen, MIDP for Palm OS dynamically grows the text field, and the form that the text field is on, as the user enters text. This is consistent with the Palm OS user interface style.

1.2.3 Implicit Lists

These allow an item to have an implied action, and are designed for cell phones, where select and focus are separate actions. MIDP for Palm OS must explicitly represent the implied action so that the user can select it. To do this, MIDP for Palm OS uses an arrow button.

You can see the arrow buttons in the following figure, which shows the initial screen of a MIDlet suite that contains multiple programs. The user presses the arrow button beside the name of the application to run.

FIGURE 2 An Implicit List

1.2.4 Ticker Tapes

A ticker tape shows information, like stock quotes or news headlines, which is updated continuously or periodically depending upon the network connection and the application context. The data moves through the ticker tape from right to left, and starts over when the message has completed scrolling. The contents of the ticker-tape are not editable.

Ticker tapes interact with the titles of screens, and with alerts. This topic covers both interactions.

1.2.4.1 Titles

When an application uses a ticker tape along with a high level component screen, MIDP for Palm OS shows the ticker in the title area, and employs a dotted line frame to delineate the ticker area. (MIDP for Palm OS does not delineate the ticker area when a ticker-tape is not being used.) The following figure shows a ticker tape in the title area:

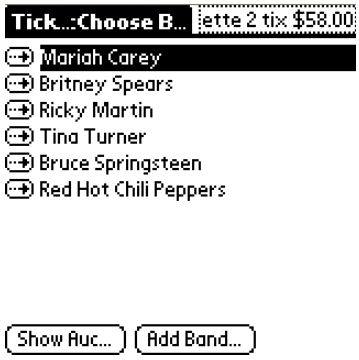

FIGURE 3 A Ticker-Tape and Title

The ticker tape and the title graphic share the title area dynamically. If the title is longer than 60% of the display width, then the ticker is clipped to use no more than 40% of the width of the display. Otherwise, the ticker can use whatever space the title is not using, and vice versa.

Users can tap on the ticker tape to popup a full screen-width version of the ticker tape, and tap again to reduce the size back to normal. FIGURE 4 shows a full screen-width version of the ticker-tape in FIGURE 3. The user can also drag within the ticker tape region to move the text string back and forth.

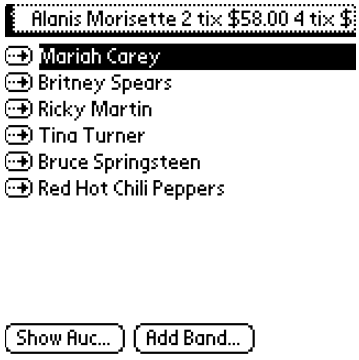

FIGURE 4 Full-Screen Width Ticker Tape

1.2.4.2 Alerts

MIDP for Palm OS supports both modal and timed alert boxes. When an application uses a ticker tape with a modal alert, MIDP for Palm OS shows the ticker tape between the Done button and the scroll indicator region at the bottom of the dialog. When an application uses a ticker tape with a timed alert, MIDP for Palm OS shows the ticker tape to the right of the timer region.

Note that, although you can use a ticker tape in an alert, it is not recommended. Alerts interrupt users and are very transitory.

1.2.5 Forms

A form is a screen that holds a small number of closely related UI objects, called items. The items can be for display, such as strings and images, or they can be interactive, such as choice groups, gauges, and text fields. Try to limit form screens to no more than 2-3 screens in length for both usability and memory usage.

This section discusses how MIDP for Palm OS lays out a form screen. It covers the topics:

- Positioning Form Items
- Scrolling
- Updating Visible Forms

1.2.5.1 Positioning Form Items

Items in a form can have associated labels, and can take up multiple lines of vertical space when the runtime lays them out on the screen. When the runtime lays out items in a form, it uses the following rules:

- An item label:
  - Can use the entire width of the screen.
  - Can be one or two lines in length.

    If the label has a newline in it, the runtime respects it.

    If a label is more than two lines long, MIDP for Palm OS clips the label and indicates the clipping with an ellipsis (`...`).
  - Will end with a colon (`:`) unless it is clipped.
  - Lines up with other labels on the colons unless it is the width of the full screen.

- The item content:
  - Is indented 64 pixels from the left edge.
  - Is placed on the same line as the label, unless the label is longer than the indent. If the label is longer than the indent, MIDP for Palm OS puts the item content on the next line.

In some cases, such as date and time fields, a special editing screen appears when the user selects the item. When this happens, MIDP for Palm OS uses the item label as the title of the editing screen.

Forms can also contain image items and string items that the user cannot select or edit. When MIDP for Palm OS lays out an image, it uses the following rules by default:

- Images wider than the item-content region are clipped on the right side.
- If the image is 12 pixels or less in height:
  - It is placed in-line with any text if it is the same width or narrower than one line of text, and the string item after the image has no associated label.
  - It is placed on its own line, left-justified in the item region, if it is too wide to share a line with text.
- If the image is taller than 12 pixels, it is placed on a line by itself, centered in the item region. If the image is as wide as the item region (after any required clipping), it is left justified in the item region.

You can override the default rules and request that images be placed left, right, or center:

- Left means that the image starts a new line and any text follows the image. For images too wide to permit text on the same line, any text is placed on the next line.
- If the image is set to right, the text starts the line and the image follows. If the image is too wide to share a line with any text, the image is placed on its own line below the text.
- If the image is set to center, the image is centered, with text before and after the image.

When MIDP for Palm OS lays out a string, it wraps the text on word boundaries or, if a word is too long to display on one line, it breaks the word on character boundaries. If the next string does not have an associated label, it will be concatenated with the previous string, unless it has a layout constraint requesting that it start on a new line.

1.2.5.2 Scrolling

Forms automatically use scroll indicators for vertical scrolling. All the items in a form screen scroll together as a unit. There is no horizontal scrolling.

When a form is scrolled up or down, it scrolls item by item. If an item is longer than what can be shown on one screen, then the item is scrolled screen by screen until all of the item has been displayed. Then the form returns to scrolling item by item. When scrolling, the form keeps one item of context from the prior screen. If the item that would be kept is more than one-half the screen height, the form does not keep any context, but instead scrolls past that item. If an item is more than one screen long, then the form keeps one line of context as it scrolls through the item.

When the user scrolls down, MIDP for Palm OS uses the top of an item as the reference point for showing the item. When the user scrolls up, MIDP for Palm OS uses the bottom of an item as the reference point for showing the item. Scrolling is symmetrical.

1.2.5.3 Updating Visible Forms

If your application changes items in a form while the form is displayed, MIDP for Palm OS adjusts what the user sees. If an item is changed, the runtime has to repaint all the items below it. You should avoid such updates because users can find them disconcerting; repainting an entire form can cause flickering. (Note that changing a `Gauge` or a `DateField` item will not cause the repainting of the items below them.)

1.3 Color Support

The MIDP specification does not require any particular color support. It states, in section 9.4.2:

> A 24-bit color model is provided with 8 bits each for the red, green, and blue components of a color. Not all devices support 24-bit color, so they will map colors requested by the application into colors available on the device. ... This enables applications to adapt their behavior to a device without compromising device independence.

MIDP for Palm OS provides black-and-white color support for 2-bit Palm OS devices, four and sixteen levels of grayscale for more capable Palm OS devices, and 256 colors or thousands of colors for more Palm OS devices that can handle colors.

A color device with 256 colors has the standard palette, sometimes referred to as the *web-safe palette*. This set has 16 shades of gray and the 216 colors that are made up of six levels each of red, green and blue. (This is a 6x6x6 color cube. It contains all combinations of 0x00, 0x33, 0x66, 0x99, 0xcc, 0xff for each of R, G, and B.)

A color device with thousands of colors has a larger range of colors, sometimes referred to as *5:6:5*. It has 32 gray levels, 32 levels of red, 64 levels of green, and 32 levels of blue. PNG images have access to the full 16 bits worth of color information, but primitive graphics (lines, text, arcs, and rectangles) have the same palette as on a 256-color device.

The number of colors used by the device is a preference that the user can set; only options that fit the device are given to the user. For example, the Color preference of a 2-bit device does not include 256 colors as a possible value. The best color option is used as the default for the device.

CHAPTER 2

Protocols

MIDP for Palm OS provides three extra protocols so that Java applications can take advantage of specific features provided by the Palm OS platform. These protocols are not part of the MIDP specification; applications that use these protocols may not run on other MIDP-compliant devices.

This chapter describes the extra protocols. It contains the sections:

- The inethttp and inethttps Protocols
- Sockets

Note that two of the protocols, inethttps and sockets, have not been thoroughly tested and are *not* supported features of MIDP for Palm OS.

2.1 The inethttp and inethttps Protocols

There are two networking libraries available on Palm OS: NetLib and INetLib. The HTTP protocol in MIDP for Palm OS always uses NetLib to perform network communication. This configuration works in most cases.

In some cases, however, the only way for the device to connect to the internet is through INetLib. For example, a PalmVIIx device with the built-in antenna and without any external modem attached must use INetLib. For such cases, MIDP for Palm OS provides two alternate protocols, inethttp and inethttps, that permit the use of INetLib.

To write an application that uses INetLib to communicate with web servers, have your code use the inethttp or inethttps protocols instead of http. The following example shows both the inethttp and inethttps protocols being used:

```
conn = (HttpConnection)Connector.open(
    "inethttp://www.yahoo.com/");
conn = (HttpConnection)Connector.open(
    "inethttps://banking.wellsfargo.com/");
```

To detect whether the device supports the inethttp and inethttps protocols, check the value of the `com.sun.midp.inethttp` property. A value of `true` means that the device supports the protocols. The following example shows this property being checked:

```
String value = System.getProperty("com.sun.midp.inethttp");
if ("true".equals(value.toLowerCase())) {
    // Yes, inethttp/inethttps protocols are supported.
    // This device has INetLib installed (probably a PalmVII)
    ...
}
```

The inethttp and inethttps protocols are somewhat limited compared to the HTTP protocol defined by the MIDP specification. Limitations include:

- The protocols expect your application to use only the standard port numbers (port 80 for the inethttp protocol and port 443 for the inethttps protocol). Other ports may not work.
- The protocols support only a very limited number of response headers.
- The protocols do not support request headers. Attempts by applications to set request headers have no effect.
- All POST data must be URL-encoded before written to the output stream of the inethttp or inethttps `Connection` object. You may need to modify your HTTP servers to understand the URL-encoded data.

If your application uses both the http and inethttp protocols, you should provide an appropriate interface for the user to choose which protocol to use.

2.2 Sockets

MIDP for Palm OS provides TCP client sockets. To use a TCP client socket, you open it with an `open` method of the `Connector` object. Every `open` method takes a URL string as an argument; the correct syntax of this string for TCP client sockets is:

`socket://`*serverHostName*`:`*serverPortNum*

The following example shows a TPC client socket being opened:

```
conn = (StreamConnection)Connector.open(
    "socket://www.yahoo.com:80");
```

After the connection is established, you can obtain `InputStream` and `OutputStream` objects from this connection and use them to communicate with the server.

CHAPTER 3

Converting Spotlets to MIDlets

This chapter explains how to migrate a Spotless application to a standard Java™ 2 Platform, Micro Edition (J2ME™) MID Profile application (a MIDlet). It also shows you how to enhance the converted MIDlet to ensure portability among all MIDP-compliant devices. If you have not developed Spotlets, you can skip this chapter.

This chapter contains the sections:

- Spotlets and MIDlets
- Converting a Spotlet to a MIDlet
- Packaging Your MIDlet
- Converting Your MIDlet into a PRC File
- Improving Your Converted MIDlet

This chapter assumes a basic knowledge of J2ME, CLDC (Connected Limited Device Configuration), and the MIDP API. Its examples are based on the CLDC 1.0.1 with Palm Overlay release and MIDP for Palm OS 1.0. They both run on Palm OS devices.

3.1 Spotlets and MIDlets

A Spotlet is an application that runs in the Spotless environment, a research project of Sun Microsystems Laboratories. Spotless was a proof-of-concept implementation of the Java virtual machine (JVM™) for the Palm OS, a popular Personal Digital Assistant (PDA). It was never meant to be a commercial product and is not supported by Sun Microsystems, Inc. See `http://www.sun.com/research/spotless/` for more information.

A MIDlet is an application that runs in the J2ME MID-Profile environment. MIDlets are packaged in a MIDlet suite, which is a JAD (Java application description) file and a JAR (Java archive) file. See `http://java.sun.com/products/j2me/` for more information.

The Spotless and MIDP environments differ in several ways, including:

- Their levels of support – Spotless was a proof-of-concept implementation for KVM running in a Palm OS environment; it is not supported. MIDP is standardized through the Java Community Process, which is a standard community organization. MIDP is supported.
- The availability of their runtime environments – Spotlets can run only on Palm OS devices. A MIDlet can run on any MIDP-compliant device.
- Their levels of security – Spotless posed a higher security concern because it gives you access to Palm-native functions. MIDlets do not have APIs that access native functions on MIDP-compliant devices, which reduces the chance of unauthorized use of the device.
- Their look-and-feel – In the Spotless environment, user interface (UI) components were drawn using basic graphics routines. MIDP for Palm OS 1.0 uses Palm native UI components to retain the look-and-feel of a Palm native application.

3.2 Converting a Spotlet to a MIDlet

This section shows you how to convert Spotlets to MIDlets. It covers the topics:

- Updating Import Statements
- Managing the Application Life Cycle
- Handling Events
- Updating the User Interface
- Migrating Calls to Database APIs
- Using the MIDP `Timer` class
- Converting Your Spotlet: Summary of Capabilities

3.2.1 Updating Import Statements

The Spotless package was `com.sun.kjava`, while the MIDlet package is `javax.microedition.midlet` and the MIDP LCDUI package is `javax.microedition.lcdui`. Update your import statements with the MIDP package names. (Other sections, which discuss specific APIs that you might have used in your Spotlet, note the MIDP packages that you need to import as part of your conversion.)

3.2.2 Managing the Application Life Cycle

Spotless lacked the idea of an application life cycle, but MIDP provides complete life cycle management. When you create a Midlet, you subclass the `javax.microedition.midlet.MIDlet` class and implement its life cycle methods, which are abstract. These methods are `startApp`, `pauseApp`, and `destroyApp`.

3.2.3 Handling Events

Spotless had pen-based event handling. When you write a Spotlet, you subclassed `com.sun.kjava.Spotlet`. You then registered with the Spotless system to get and relinquish the event focus using the `register` and `unregistered` methods. For buttons, you use the `pressed` method from the `penDown` or `keyDown` event to tell whether a button has been pressed. MIDP, on the other hand, splits its event handling APIs into high-level and low-level.

MIDP high-level event-handling APIs are for user-interface components such as lists and forms. You use a `CommandListener` to carry out the action of component, for example when the user selects an item in a list.

MIDP low-level event-handling APIs, on the other hand, are for unstructured blank canvases. Low-level events include the following callbacks: `keyPressed`, `keyRepeated`, `KeyReleased`, and `getGameAction`. Low-level events will also include pen events, if the device supports them. If your Spotlet used pen events, and you want to use them in your MIDlet, you must first query to see if a pen is available. To query, use the methods `hasPointerEvent` and `hasPointerMotionEvents.` If a pen is available to the MIDlet, you can use the low-level event callbacks `pointerPressed`, `pointerReleased`, and `pointerDragged` to handle pen input.

Spotless had a beaming event. MIDP for Palm OS does not have such an event. If you have your own creator ID and use it when you create a PRC file that is distributed to users, you can decide whether to lock your MIDlet or allow it to be beamed. If, instead, you distribute your MIDlet using a JAR/JAD file pair and your

users create a corresponding PRC file using the PRC Converter tool, they will be able to beam your MIDlet while the MIDlet is running using the Beam App... menu item shown in the following figure:

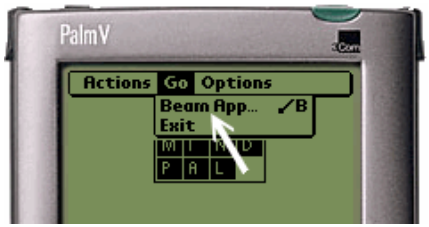

FIGURE 5 Beaming a MIDlet While It Is Running.

The following table summarizes Spotless events and their MIDP equivalents.

TABLE 1 Event API Summary for Spotlets and MIDlets

Event	Spotless Event API	MIDP Low-Level Event
Pen Down	`penDown(int x, int y)`	`pointerPressed( int x, int y)`
Pen Up	`penUp(int x,int y)`	`pointerReleased( int x, int y)`
Pen Moved	`penMove(int x, int y)`	`pointerDragged( int x, int y)`
Key Down	`keyDown(int keyCode)`	`keyPressed(int keyCode)`
Beam Received	`beamReceive( byte[] data)`	Unsupported

3.2.4 Updating the User Interface

A typical user-interface in a Spotlet included buttons, bitmaps, graphics drawn with low-level graphics methods, and sounds. This section shows you how to migrate them, and other user-interface components, into Java applications written for the MIDP environment. The section covers the topics:

- Managing Different Screen Sizes
- Converting `Button` Classes
- Interacting with the Physical Buttons of a Device
- Using Bitmaps
- Drawing Graphics

- Converting `HelpDisplay` Classes
- Using a Scroll Bar
- Playing Sounds

3.2.4.1 Managing Different Screen Sizes

Because a Spotlet was an application for a Palm OS device, you could rely on having a screen of size 160x160. A MIDlet is an application for many different mobile application devices, however, so you cannot rely on having a screen of any particular size. Even if you expect your MIDlet to be used only on a Palm OS device, you cannot rely on having a 160x160 screen, because MIDP allows users to turn on application buttons or game controls or both. Application buttons and game controls impact the size of the screen available to the application. For example, the following figures show an application with and without the game controls and application buttons. See the *User's Guide* for more information on the Game Controls and Application Buttons preferences.

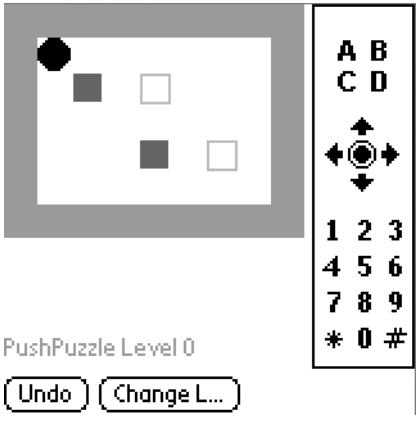

FIGURE 6 Application Using Both Buttons and Game Controls

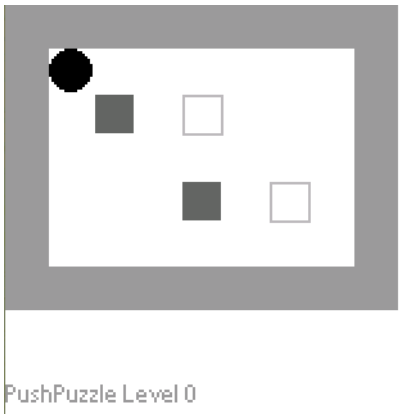

FIGURE 7 Application Using Neither Buttons nor Game Controls

Instead of relying on a particular size of screen, get the height and width of the screen by calling the `getHeight` and `getWidth` methods of the `Canvas` class.

3.2.4.2 Converting `Button` Classes

If your Spotlet had buttons that carried out actions, you can make those buttons either abstract commands or an implicit list. This section covers both topics.

Abstract Commands

An abstract command carries out an action, like a button does. The difference is that MIDP implementations on the various mobile devices can present the action to the user in different ways. In fact, the abstract command may not even map to a button on the device; it may map to a menu instead. The device manufacturer controls the mapping. See Section 1.1 “Abstract Commands” on page 1 for more information.

Although abstract commands do not provide the precise user-interface control of a Spotless button, they do make your application more portable. You no longer need to worry about the position of the button when your MIDlet runs in devices with different sized screens. For example, suppose you have a Spotlet with nine commands, including Exit. In Spotless, you created a `Button` class for each command, and provided the button’s string and its X and Y coordinates. You would have created and placed nine buttons, including an Exit button, as shown in FIGURE 8. In MIDP LCDUI, you create nine `Command` objects, and MIDP for Palm OS places them onto the Palm OS device. It will place one or more of them on buttons at the bottom of the screen, and the rest in pull-down menus. FIGURE 9 shows the

Actions pull-down menu, which holds most of the abstract commands. The Exit command is not shown since MIDP for Palm OS places it on the Go menu. You do not call a paint method for an abstract command, as you did for Spotless buttons.

FIGURE 8 Spotlet Buttons (Typical Layout)

FIGURE 9 MIDlet Abstract Commands
(Corresponding to the Spotlet Buttons in the Previous Figure)

Tip – When you design an abstract command, keep its label short. If the label is too long, MIDP for Palm OS will have to truncate it and append ellipses (...).

FIGURE 9 shows an example of the truncation and ellipses. Slower became Slow...

A final difference between Spotless buttons and abstract commands is the way that the command is carried out. In Spotless, you used the `pressed(x,y)` method from the `penDown` event (or `keyDown` event) to tell whether a button had been pressed, and then, if necessary, carried out the button's action. In MIDP, you implement a `CommandListener`, which carries out the actions of the commands. You implement one `CommandListener` for each screen.

Implicit Lists

Instead of converting your buttons to abstract commands, you could convert each button to a choice in a list. The list would be of type `IMPLICIT`, and would be on a separate screen, implemented with a `List` object. You would provide a single command to switch to this screen. The following figures show the single command to switch to the implicit list, and then the resulting implicit list.

FIGURE 10 Command That Leads to an Implicit List

FIGURE 11 Implicit List
(Corresponding to the Spotlet Buttons in FIGURE 8)

See "Implicit Lists" on page 4 for more information.

3.2.4.3 Interacting with the Physical Buttons of a Device

A Spotlet typically maps the up and down buttons of a Palm OS device to fixed actions; MIDP does not. Mapping to specific device buttons is not portable (some devices might not have up and down buttons) and the goal of MIDP is to be portable

across different mobile devices. Instead, MIDP provides abstract game actions as part of its `Canvas` class. These game actions are triggered not by specific keys, but by key codes that the device manufacturer can map to any keys, soft buttons, or menus on its device. The game-action key codes are `UP`, `DOWN`, `LEFT`, `RIGHT`, `FIRE`, `GAME_A`, `GAME_B`, `GAME_C`, and `GAME_D`. You determine which action the user requested by calling the `getGameAction(int keyCode)` method of the `Canvas` object.

Because the device manufacturer determines the mapping of game actions to the device, avoid device-specific instructions in your help text. For example, use generic text like this:

```
Use Up to move the game piece forward one square.
```

It is more portable than device-specific text like this:

```
Use the corner red button of the cell phone to move the game piece
    forward one square.
```

3.2.4.4 Using Bitmaps

In Spotless, a bitmap was closely tied to the Palm OS bitmap format. For example, it automatically pads odd-width bitmaps. In contrast, MIDP images use the PNG (Portable Network Graphics) format. If you do not want to change your original black and white bitmap format, the following code shows a way to draw the pixels onto a new image:

```
static Image createBlackAndWhiteImage(byte[] bitmap,
                                       int imageWidth) {
    int imageHeight;
    Image tempImage;
    Graphics g;
    int bit = 0;

    /* To get the height, divide the number of bits
     * by the width */
    imageHeight = (bitmap.length * 8) / imageWidth;

    tempImage = Image.createImage(imageWidth, imageHeight);

    /* the image is already blank, so draw the black bits */
    g = tempImage.getGraphics();
    g.setColor(0);
    for (int i = 0; i < bitmap.length; i++) {
        for (int j = 7; j >= 0; j--) {
            if ((bitmap[i] & (1 << j)) != 0) {
                g.fillRect(bit % imageWidth,
                           bit / imageWidth, 1, 1);
            }
            bit++;
        }
    }
    return tempImage;
}
```

Use raw PNG bytes for improved performance. Using raw data directly enables you to take advantage of any graphics acceleration hardware that the device might provide.

Note – MIDP does not support Sprite.

3.2.4.5 Drawing Graphics

Spotless provided access to Palm OS drawing routines; its APIs map directly to the Palm drawing APIs. MIDP does not provide this level of device-specific access. MIDP's drawing operations allow only pixel replacement. Spotless methods like `copyRegion` are not supported. Direct manipulation of pixel values, such as rasterops or alpha blending, is also not supported. The following table shows the Spotlet graphics APIs and their MIDP equivalents:

TABLE 2 Spotless and MIDP Graphics Methods

Spotless Graphics API	Corresponding MIDP LCDUI	Notes
`drawLine`	`drawLine( int x1, int y1, int x2, int y2)`	
`drawRectangle`	`fillRect( int x, int y, int width, int height)`	
`drawBorder`	`drawRect( int x, int y, int width, int height)`	Resulting rectangle is (width+1) x (height+1)
`drawString`	`drawString( String string, int x, int y, int anchor)`	
`setDrawRegion`	`setClip ( int x, int y, int width, int height)`	
`resetDrawRegion`	None	Can use `setClip(0,0, canvas.getWidth(), canvas.getHeight() )`
`copyRegion`	None	None
`clearScreen`	None	Can use the `fillRect` method with the whole screen.
`drawBitmap`	`drawImage ( Image image, int x, int y, int anchor)`	Spotless has a maximum bitmap width of 32 pixels

In Spotless you explicitly called the `paint` method. MIDP provides different levels of repainting support in its high-level and low-level APIs. The high-level user-interface components, such as alerts, lists, and text boxes, handle their own repainting automatically. The low-level component, the canvas, repaints asynchronously. That is, you can request that the screen be repainted by calling a `repaint` method, but the actual repainting might happen later. Further, several of your repaint requests might (and probably will) result in a single repainting of the screen. To flush the repaint request event queue, call the `serviceRepaints` method directly.

MIDP does not permit both low-level and high-level components in a single screen. You must redesign any screens in your Spotlet that had both graphics and high-level components such as list screens. One way is to, conceptually, split such Spotlet screens into high-level and low-level MIDP screens, as shown in the following figure:

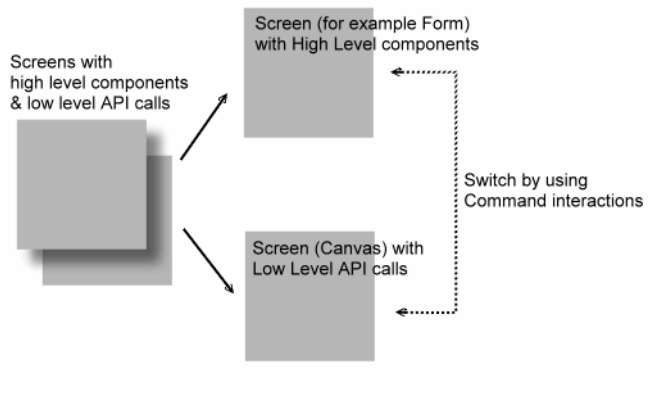

FIGURE 12 Redesigning a Spotlet Screen Using Both High- and Low-Level MIDP Screens

By adding abstract commands to your low-level screen, you can provide a way for users to switch from a low-level screen to a high-level screen that provides high-level user interaction.

3.2.4.6 Converting `HelpDisplay` Classes

The Spotless `HelpDisplay` class shows a scrollable text box with a button titled Done. If your Spotlet used the `HelpDisplay` class, you need to convert the help displays into screens in your MIDlet. There are various ways to do this:

- For a help display with a simple text string, put the string into an alert.
- For more complex help display, either:
  - Map them to the most suitable LCDUI screens, such as lists or text boxes.
  - Map them onto items in a form, such as text fields, lists, date fields, or gauges. Spotless and MIDP LCDUI both contain a text fields, while a MIDP gauge is similar to a Spotless slider.

If these do not fit the style of your Spotlet application, you can draw the item that was in the `HelpDisplay` using the `Canvas` class, as discussed in the previous section, "Drawing Graphics." This should be a last resort, however, after you have considered redesigning your application to fit the MIDP design paradigm.

The following example shows an alert being used to display a simple string from a Spotlet's help display:

```
Public class Sample extends MIDlet {
...
Public Sample() {
    ...
    private Alert directions;
    static final String HELP_TEXT =
        "Welcome to the Sample!. Instruction here";
    static final String HELP_TITLE = "Help"

    // Parameter is the title of the Alert
    directions = new Alert(HELP_TITLE);
    // HelpDisplay text is the Alert string
    directions.setString(HELP_TEXT);
    directions.setTimeout(Alert.FOREVER);
}

private void help() {
    Display.getDisplay(this).setCurrent(directions);
}
```

The following figures show an original Spotless `HelpDisplay` and the corresponding alert in the resulting MIDlet. The MIDlet uses code similar to the previous example to implement its alert.

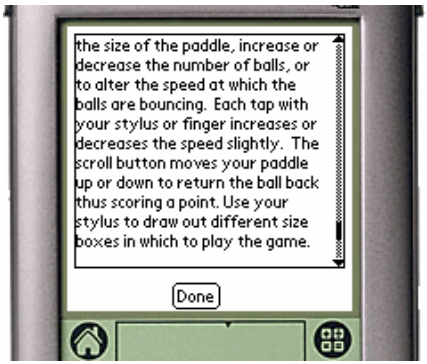

FIGURE 13 Help Screen in Spotless, using the `HelpDisplay` class

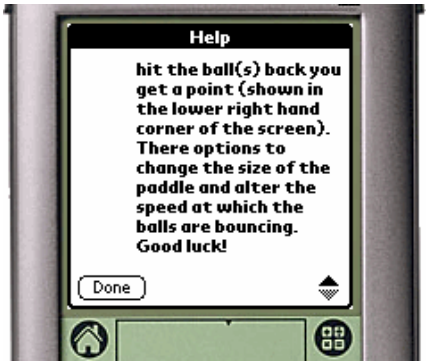

FIGURE 14 Help Screen in MIDP, using an `Alert` screen object

3.2.4.7 Using a Scroll Bar

MIDP LCDUI does not provide a scroll bar, because the MIDP runtime automatically controls scrolling indicators.

3.2.4.8 Playing Sounds

Both MIDP and Spotless provide simple sound playing capabilities. Spotless had a `playSound` method in the `Graphics` class, which played a Palm OS device's standard beep. MIDP LCDUI has a `playSound` method in the `AlertType` class. The `AlertType` class is a typesafe enum, with five enumeration constants:

`INFORMATION`, `WARNING`, `ERROR`, `ALARM`, and `CONFIRMATION`. The `playSound` method of each `AlertType` can play a different sound. The sound or sounds that the methods actually play is determined by the device manufacturer. (The device manufacturer can also choose to have one or more `playSound` methods play no sound at all.)

3.2.5 Migrating Calls to Database APIs

Spotless provided a database class, `com.sun.kjava.Database`, that directly manipulates a Palm OS database. Although MIDP does not enable you to manipulate a Palm OS database directly, it does provide a record-oriented database for your application's data. The record-oriented database, Record Management System (RMS), provides a layer of abstraction between any device database and your application logic. To use it, import the `javax.microedition.rms` package into your MIDlet.

In Spotless, you had to obtain and use a valid creator ID to create a Palm OS database. In MIDP for Palm OS, you still need a valid creator ID, but the converter and the MIDP for Palm OS runtime take care of creating and using the creator ID for you. (You can also deploy a MIDlet with your own Creator ID. To supply your own creator ID, use the command-line converter tool to change the MIDlet's JAR and JAD files into a PRC file. See Chapter 4 "Converting MIDlets to Palm OS Applications" for more information.)

Your MIDlet cannot directly access all RMS databases. It can only read from and write to the databases that it or another MIDlet in the same MIDlet suite created.

3.2.6 Using the MIDP `Timer` class

MIDP includes the `java.util.Timer` class, which schedules tasks for future execution. If you created your own timer or used the `System.currentTimeMillis` method to implement a timer in your Spotlet, convert it to the MIDP `Timer` and `TimerTask` classes. See the MIDP specification for more information. There is also example code that uses timers at this URL:

`http://developer.java.sun.com/developer/products/wireless/midp/samples/index.jsp#timers`

3.2.7 Converting Your Spotlet: Summary of Capabilities

The following table summarizes the major differences in the capabilities of Spotless and MIDP LCDUI.

TABLE 3 Comparison of Spotless and MIDP LCDUI Classes

Type of Component	Component	Spotless Support	MIDP LCDUI Support
High-Level Component	Abstract Command	No	Yes
	Button	Yes	No (Can use abstract commands.)
	Radio Button	Yes	No (Can use an exclusive choice list or an exclusive choice group on a form.)
	Text Area Input	Yes (With limited word processing power)	Yes (Can use either a text box or text field.)
	Text Field	Yes	Yes (Can specify different formatting.)
	Progress Bar or Gauge	No (Usually implemented with two buttons and a label)	Yes
	Scrollbar	Yes	No (Scrolling is made available automatically.)
	Choice Selection/List	No	Yes
	Ticker	No	Yes
	Timed Alert	No	Yes
Graphics	Bitmap Image	Yes (Black and white bitmaps)	Yes (PNG format)
	Canvas	Yes	Yes
	Color Support	No	Yes

TABLE 3 Comparison of Spotless and MIDP LCDUI Classes *(Continued)*

Type of Component	Component	Spotless Support	MIDP LCDUI Support
Dialogs	Modal	Yes	No (Modal alerts are similar.)
	Multiple buttons	Yes	No (Can use abstract commands.)
DataBase	Database support	Yes (Palm database)	Yes (RMS)
Special	Infrared	Yes	No
	HTTP Connection	No (Not officially supported)	Yes
	Timer	No	Yes

3.3 Packaging Your MIDlet

To distribute a MIDlet, MIDP requires at least a JAR file. You should also have a Java application description (JAD) file, since the PRC Converter Tool, with which end-users will convert MIDlets to Palm OS applications, needs a JAR/JAD file pair. A JAD file is a text-file description of the JAR file and the MIDlet suite. See the MIDP specification, Chapter 8, for a detailed explanation of the required JAR and JAD file formats.

3.4 Converting Your MIDlet into a PRC File

In Spotless, the Java application `MakePalmApp` converted a Spotlet into a PRC file. MIDP for Palm OS provides the `Converter.jar` file to convert a MIDlet into a PRC file. The `Converter.jar` file can be either a Swing-based or a command-line converter tool, depending on how you start it. See the *User's Guide* for information on the Swing-based PRC Converter Tool. See Chapter 4 "Converting MIDlets to Palm OS Applications" for information on the command-line converter tool. Note that the MIDP for Palm OS converter tools have more restrictions than the Spotless `MakePalmApp` application.

3.5 Improving Your Converted MIDlet

After you have completed the initial conversion of your Spotlet to a MIDlet, there are a few things that you might be able to improve. This section covers those topics:

- Graphics Performance
- Device Independence

3.5.1 Graphics Performance

Often, an initial conversion to a MIDlet results in an application that refreshes the screen too many times. You have this common problem if, when you run your MIDlet, the screen seems to flicker. Here are some ways to work around it:

- Use double-buffering – You can use the `isDoubleBuffered` method of the `Canvas` class see whether the device has double-buffering capability. If the device does not provide native double-buffering capability, you can provide your own implementation. As a rule of thumb, do not rely on the device to provide it.

  The user can turn on double-buffering by setting the Drawing Speed preference to Smooth. See the *User's Guide* for more information.

- Erase and redraw only small parts of the screen – This technique minimizes the time required to repaint the screen. (Also, depending on the MIDP implementation, the runtime may be able to save several of these repaints and carry them out as a single action, as an optimization.) This is an important technique because not all devices have double-buffering: it consumes a lot of memory and some devices lack both the system resources and a powerful enough graphics chip.

3.5.2 Device Independence

A well-written MIDlet should be easy to use on a variety of MIDP-enabled devices (such as PDAs, cell phones, and pagers) that have different form factors and input mechanisms. Check your MIDlet to ensure you have followed these rules of thumb:

- Keep the labels of abstract commands short and clear. (See Section 3.2.4.2 "Converting `Button` Classes" on page 20 for more information.)
- Do not assume that any specific keyboard, numerical keypad, or keys will be available. (See Section 3.2.4.3 "Interacting with the Physical Buttons of a Device" on page 22 for more information.)

- Do not assume there will be Stylus input. (See Section 3.2.3 “Handling Events” on page 17 for more information.)
- Do not assume any particular screen dimensions. (See Section 3.2.4.1 “Managing Different Screen Sizes” on page 19 for more information.)

Following these rules makes it more likely that your MIDlet will prove to be easy to use and truly device-independent when you run it on other MIDP implementations.

CHAPTER 4

Converting MIDlets to Palm OS Applications

This chapter covers the command-line Java technology-based application, `MakeMIDPApp`, that converts a Java application written for the MIDP environment (a *MIDlet*) into a Palm OS application (a PRC file). This chapter contains the sections:

- Overview
- Location of `MakeMIDPApp`
- Syntax of Starting `MakeMIDPApp`
- Options to `MakeMIDPApp`

This chapter does not discuss how to set environment variables for the Java platform. It assumes that you know how to set up your environment.

4.1 Overview

When you write a MID Profile application (a MIDlet), you package it in a pair of files. The files have the same name, but different extensions: one is a JAR file with a `.jar` extension, the other is a JAD file with a `.jad` extension. (See the MIDP specification at `http://java.sun.com/products/midp/` for more information.) The JAR/JAD file pair must be converted to a resource application file (a PRC file) before the application can be downloaded onto a Palm OS device.

There are two ways to change a JAR/JAD file pair to a PRC file: with the PRC Converter Tool and with the `MakeMIDPApp` application. The PRC Converter Tool is a Swing-based application, while `MakeMIDPApp` is a command-line utility. Using `MakeMIDPApp` gives you more flexibility in how your JAR/JAD file is converted; the PRC Converter Tool is easier to use but has fewer options. This chapter covers `MakeMIDPApp`. For more information on the PRC Converter Tool, see the *User's Guide*.

The Java class file `com.sun.midp.palm.database.MakeMIDPApp` is an application that converts a MIDlet into a PRC file. The resulting PRC file contains your application's classes and classes that launch MIDP for Palm OS. The PRC file also contains the non-Java files from your MIDlet's JAR file (such as graphics files).

After you run the `MakeMIDPApp` application, you can download the resulting PRC file onto a Palm OS device. It will then appear in the device's Launcher just like any other Palm OS application. When the application is on a Palm OS device with the MIDP for Palm OS runtime environment, you can run the application. You launch your application just like any other Palm OS application: by tapping its icon.

`MakeMIDPApp` has been tested with all versions of JDK1.3.x but has not been tested with JDK1.0.x.

4.2 Location of `MakeMIDPApp`

`MakeMIDPApp` is in the `Converter.jar` file, along with the Swing-based PRC Converter Tool. The `Converter.jar` file is part of the MIDP for Palm OS installation. See the *ReadMe* file for a complete description of the contents of the MIDP for Palm OS release. See the *User's Guide* for information on the PRC Converter Tool.

4.3 Syntax of Starting `MakeMIDPApp`

Use the `java` command to run `MakeMIDPApp`. Running the application has the following syntax:

```
java -cp Converter.jar com.sun.midp.palm.database.MakeMIDPApp
    [options...] JARfile
```

The *JARfile* argument names the JAR file that contains the application class files and other resources that are to be converted to a Palm OS application (a PRC file). The JAR file specified must be in a valid Java archive format. See the *Jar File Specification* at `http://java.sun.com/j2se/1.3/docs/guide/jar/jar.html` for information on JAR files and their manifests. JAR files for the MIDP environment have additional properties requirements. For information on those requirements, see the MIDP specification.

See Section 4.4 "Options to `MakeMIDPApp`" on page 37 for information on the options to `MakeMIDPApp`.

For example, to create a `CoolApp.prc` file, you could execute this command:

```
java -cp Converter.jar com.sun.midp.palm.database.MakeMIDPApp
    -type Data CoolApp.jar
```

Note – If you use the `-jar` option to `java`, JDK1.3 will run the PRC Converter Tool. That is, the following command will start the PRC Converter Tool:

```
java -jar Converter.jar
```

4.4 Options to `MakeMIDPApp`

The tables below provide a complete list of options and short descriptions. Each option is described more fully the sections that follow.

TABLE 4 Supported Options for the `MakeMIDPApp` Application

Option	Brief Description
`-v[erbose]`	Requests verbose output (-v -v gives you even more information)
`-jad` *JADFile*	Packaging file for the MIDlet.
`-icon` *file*	File that contains the large Palm OS icon (32x32) for the icon-view of the application.
`-smallicon` *file*	File that contains the small Palm OS icon (15x9) for the list-view of the application.
`-name` *name*	Short name of the application, for the Palm OS device's icon view.
`-longname` *name*	Long name of the application, for the Palm OS device's list views.
`-creator` *crid*	Unique Creator ID for the application.
`-type` *type*	Type of application you are converting.
`-o[utfile]` *PRCfile*	PRC file to create.
`-help`	Requests a help message showing the options to `MakeMIDPApp`.

4.4.1 The `-creator` Option

The `-creator` option specifies the unique, four-character identifier for this application. The option takes the identifier as an argument:

`-creator` *crid*

Every Palm OS application must have such an identifier. Creator IDs are case sensitive, and IDs of all lower-case letters are reserved for use by Palm, Inc. Palm Inc., recommends that you register your application's unique identifier at its web site; do this if you plan to use databases with your Java application, or plan to deploy your MIDlet specifically on Palm OS.

If you do not provide the `-creator` option, `MakeMIDPApp` will automatically generate a creator ID for you. If you specify `-v[erbose] -v[erbose]`, `MakeMIDPApp` prints out the creator ID that it assigned to your application. You can also see the creator ID of your application after you have installed it on your Palm OS device by using the `Developer.prc` application. See Section 5.2.2 "Developer Options in Java Preferences" on page 45 for more information.

If you do not provide a creator ID with the `-creator` option, you must provide the `-type` option and give it an argument of `Data`.

4.4.2 The `-help` Option

The `-help` option prints a message that briefly explains the options to `MakeMIDPApp`.

4.4.3 The `-icon` Option

The `-icon` option specifies the name of the file that contains the bitmap that will appear in a Palm OS device's Launcher when the user has the device set to Icon View. (This icon is commonly thought of as the large icon. To specify the small icon, which appears when the user has the device set to List view, use the `-smallicon` option, described in Section 4.4.8 "The `-smallicon` Option.")

The `-icon` option takes the filename as an argument:

`-icon` *file*

Palm OS expects the icon to be a 32x32 bitmap. It expects the five left-most columns, five right-most columns, and ten bottom rows to be white. If your file contains a bitmap of a different size (the most common is 22x22), `MakeMIDPApp` will convert

the bitmap into a 32x32 bitmap by top-adjusting it and centering it horizontally. If possible, provide icons of the correct size. Resizing bitmaps usually results in poor quality icons.

`MakeMIDPApp` determines the type of bitmap that a file contains from the extension of the filename. Bitmap files can have the following formats and extensions:

- Windows bitmap format (`.bmp`)
- Portable bitmap format (`.pbm`)
- Raw Palm OS format bitmaps (`.bin`)

This version of MIDP for Palm OS does not support color or compressed-BMP bitmaps. Any BMP, PBM, or BIN files that you provide should be black and white.

If you omit the `-icon` option, your PRC file will contain the default large icon. (See the *User's Guide* for a figure showing the default icons.)

4.4.4 The `-jad` Option

The `-jad` option specifies the name of your MIDlet's Java application descriptor (JAD) file. The option takes the file name as an argument:

`-jad` *JADFile*

The JAD file must have a valid JAD format, as specified in the MIDP specification. `MakeMIDPApp` reads the application attributes from the named JAD file and converts them into a Palm resource.

A JAD file is recommended, although it is not strictly required.

4.4.5 The `-longname` Option

The `-longname` option specifies the true name of the application: the name that appears in the Palm OS device's Beam and Delete dialogs. The option takes the name as an argument:

`-longname` *name*

The name can be at most 31 characters long, but should be shorter.

If you omit the `-longname` option, `MakeMIDPApp` uses the values of the `MIDlet-Name` and `MIDlet-Vendor` properties in the JAD file or the JAR file's manifest to create one.

4.4.6 The `-name` Option

The `-name` option specifies the name that appears with the application's icon in the Palm OS device's Launcher. The option takes the name as an argument:

`-name` *name*

If the name is more than nine characters long, it may be truncated by the Palm OS device. (The Launcher, when it is in icon view, can display names of about nine characters or less. It can display names of about eleven characters or less when it is in list view.)

If you do not specify the `-name` option, `MakeMIDPApp` uses the name specified by the `MIDlet-Name` property in the JAD file or the JAR file's manifest.

4.4.7 The `o[utfile]` Option

The `-o[utfile]` option specifies the name of the PRC file to which the output is written. It takes the file name as an argument:

`-o[utfile]` *PRCFile*

The resulting PRC file can be downloaded onto any Palm OS device.

If you do not specify `-o[utfile]`, `MakeMIDPApp` names the PRC file by appending `.prc` to the name of the JAD file. If you do not use the `-jad` option, `MakeMIDPApp` names the PRC file by appending `.prc` to the name of the JAR file.

4.4.8 The `-smallicon` Option

The `-smallicon` option specifies the name of a file that contains the bitmap that will appear in a Palm OS device's Launcher when the user has the device set to List View. (To specify the large icon, which appears when the user has the device set to Icon view, use the `-icon` option, described in Section 4.4.3 "The `-icon` Option.")

The `-smallicon` option takes the filename as an argument:

`-smallicon` *file*

Palm OS expects the icon to be a 15wx9h bitmap. If your file contains a bitmap of a different size, `MakeMIDPApp` will convert the bitmap into a 15x9 bitmap by top-adjusting it and centering it horizontally. If possible, provide icons of the correct size. Resizing bitmaps usually results in poor quality icons.

`MakeMIDPApp` determines the type of bitmap that a file contains from the extension of the filename. Bitmap files can have the following formats and extensions:

- Windows bitmap format (`.bmp`)
- Portable bitmap format (`.pbm`)
- Raw Palm OS format bitmaps (`.bin`)

This version of MIDP for Palm OS does not support color or compressed-BMP bitmaps. Any BMP, PBM, or BIN files that you provide should be black and white.

If you omit the `-smallicon` option, your PRC file will contain the default small icon. (See the *User's Guide* for a figure showing the default icons.)

4.4.9 The `-type` Option

The `-type` option indicates the type of application that is being converted to a Palm OS application. The option takes the type as an argument:

`-type` *type*

The value of *type* can be either `appl` or `Data`; the value is case sensitive.

If you do not provide the `-type` option, `MakeMIDPApp` assigns your application the default type, `appl`.

If you do not provide a creator ID with the `-creator` option, you must provide the `-type` option and give it an argument of `Data`.

4.4.10 The `-v[erbose]` Option

The `-v[erbose]` option causes `MakeMIDPApp` to produce more output explaining its actions. The effect of the option is cumulative: using `-v[erbose] -v[erbose]` produces more output than using `-v[erbose]` alone. It is not useful to use the option more than two times.

CHAPTER 5

Tool for MIDlet Testing

This chapter covers an application, `Developer.prc`, that MIDP for Palm OS provides to help you test your MIDlet. The application enables MIDP for Palm OS to show developers the internal execution status of Java applications. In addition, it enables MIDP for Palm OS capture any output from your MIDlet that you write with any print method of `System.out` or `System.err`. The `Developer.prc` application does this by activating the MIDP for Palm OS Developer Preferences screen.

This chapter contains the sections:

- Installing `Developer.prc`
- Using `Developer.prc`
- Finding a Bug In MIDP for Palm OS

5.1 Installing `Developer.prc`

The `Developer.prc` application is intended to be used by developers, not end users. It is to be used with the MIDP for Palm OS implementation, `MIDP.prc`.

Install `Developer.prc` as you would any other Palm OS application. See the *User's Guide* for detailed instructions. Installing the application puts the following icon in the Launcher of your Palm OS device:

FIGURE 15 Developer Icon

5.2 Using Developer.prc

This section covers the topics:

- Launching and Using the Developer Application
- Developer Options in Java Preferences
- Developer Options in a MIDlet
- Viewing the Contents of a MIDlet's Output Streams

5.2.1 Launching and Using the Developer Application

Launch the Developer application by tapping its icon. A screen like the one in the following figure will appear:

FIGURE 16 Main Screen of the Developer Application

This screen gives you the choice of whether to use the Developer preferences:

- Show – You can view and set developer preferences.
- Hide – The developer preferences are not available: You cannot set developer preferences. (Selecting Hide does not change the values of any Developer preferences that you had set while the Developer preferences were set to Show.) This is the default.

After you make your choice, tap the Home button ( ) to return to the Launcher of your Palm OS Device.

5.2.2 Developer Options in Java Preferences

When you choose to show the developer preferences, as described in the previous section, the new set of preferences becomes available. These developer preferences are available from the Java™ HQ application when you tap its Preferences button, and from any MIDlet when you choose Java Preferences... from the Options menu. (See the *User's Guide* for more detailed instructions.) The following figure shows the Developer option as it appears in the Java™ HQ preferences:

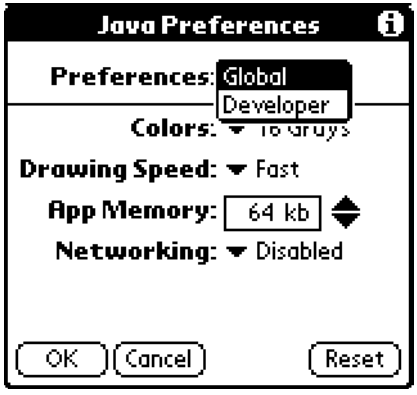

FIGURE 17 The Developer Option in the Java Preferences Screen

If you choose Developer, a Java Preferences dialog box like the one in the following figure appears:

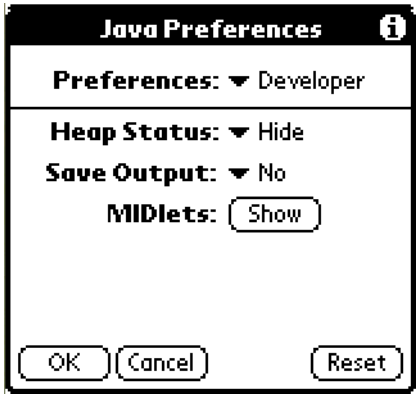

FIGURE 18 Developer Preferences Dialog Box

The developer preferences are:

- Heap Status – Whether to show the amount of memory MIDP for Palm OS allocates to the Java heap when it starts. The default is not to show this information (Hide).
- Save Output – Whether to save the data written to the `System.out` and `System.err` streams on the Palm OS device. The default is not to save this information (No).

  If you choose Yes, an Output button appears in the Java Preferences panel, as shown in the following figure:

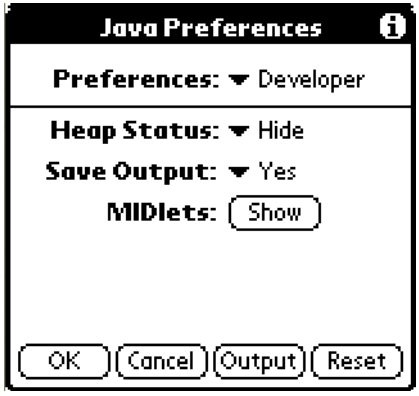

FIGURE 19 The Output Button on the Developer Preferences Screen

  When you tap this button you can examine the content of the output streams. See Section 5.2.4 "Viewing the Contents of a MIDlet's Output Streams" on page 48 for more information.

- MIDlets – When you tap the Show button, MIDP for Palm OS displays all the MIDlets installed on the device whose creator IDs are within the VM00 to VM99 range. This screen is useful for detecting what creator has been assigned to your MIDlet. See Section 4.4 "Options to `MakeMIDPApp`" on page 37 for more information on creator IDs.

5.2.3 Developer Options in a MIDlet

When you choose to show the developer preferences, as described in Section 5.2.1 “Launching and Using the Developer Application” on page 44, a new menu item becomes available in the Options menu of your MIDlets. This new menu item, Memory Info..., is shown in the following figure:

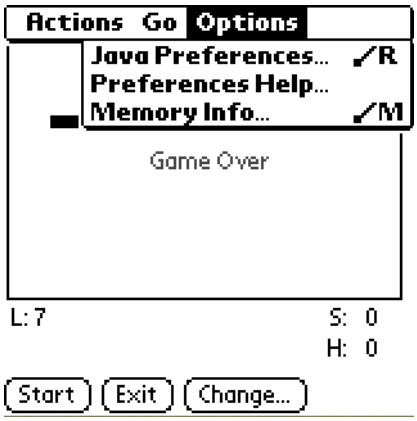

FIGURE 20 Memory Info... Item in the Options Menu of a MIDlet

If you choose the Memory Info... item, a dialog box with information similar to that shown in the following figure will appear:

FIGURE 21 Memory Information Dialog Box

The data shown in this dialog box are:

- rom – size of the ROM image on this Palm OS device
- ram – combined size of all types of RAM on this Palm OS device
- freeram – combined size of the available storage heap and dynamic memory heap on this Palm OS device

- freeheap – amount of the available storage heap on this Palm OS device
- maxheapchunk – largest chunk of free memory inside this Palm OS device's dynamic heap
- javafreeheap – amount of free memory inside the Java heap of this Palm OS device; this is the type of memory that MIDP for Palm OS uses when it creates new Java objects.
- permanent – amount of memory allocated internally by the KVM.

The information in this dialog box helps you understand the memory usage of your application. For example, if javafreeheap becomes very low, and you attempt to allocate large Java objects, the allocation will fail with an `OutOfMemoryError`.

5.2.4 Viewing the Contents of a MIDlet's Output Streams

When you create a MIDlet, you can add debugging print statements that write to `System.out` and `System.err`. By showing the developer preferences, you can save and view that output on the Palm OS device. Specifically, the steps to do this are:

1. **Add print statements that send output to `System.out`, `System.err`, or both, to your MIDlet.**
2. **Compile the MIDlet.**
3. **Package the MIDlet in a JAR/JAD file pair.**
4. **Convert the MIDlet to a PRC file.**

   You can use either the PRC Converter Tool that the *User's Guide* describes, or the command-line converter, `MakeMIDPApp`, that Chapter 4 "Converting MIDlets to Palm OS Applications" describes.
5. **Install your MIDlet on a Palm OS device (or device emulator) on which MIDP for Palm OS and `Developer.prc` have also been installed.**
6. **Show the developer preferences.**

   See Section 5.2.1 "Launching and Using the Developer Application" on page 44 for more information.
7. **Have MIDP for Palm OS capture and save the `System.err` and `System.out` output on the device.**

   See Section 5.2.2 "Developer Options in Java Preferences" on page 45 for more information.
8. **Run your application.**

After you run your application, you can examine the output streams by following these steps:

1. **Choose Java Preferences... from the Options menu of your MIDlet.**
2. **Choose Developer from the list of preference types at the Preferences prompt.**
3. **Tap the Output button. A screen like the one in the following figure will appear:**

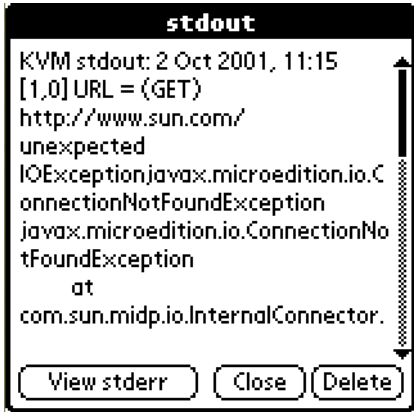

FIGURE 22 Viewing the `System.out` Output Stream of a MIDlet

- Tap the View stderr button to see the contents of the `System.err` output stream:

FIGURE 23 Viewing the `System.err` Output Stream of a MIDlet

- Tap the Close button to return to the Java Preferences dialog box
- Tap the Delete button to clear the contents of the output stream that is currently being displayed.

5.3 Finding a Bug In MIDP for Palm OS

If, while you are debugging your Java application, you think that you have found a bug in the MIDP for Palm OS implementation:

- Try to find out whether others have had this problem, and what they have done about it. You can do this at the Wireless Support page, at this URL: `http://developer.java.sun.com/developer/products/wireless/allsupport/`
- If you find that yours is a new problem, report the bug to the MIDP for Palm OS team at this URL: `http://java.sun.com/cgi-bin/bugreport.cgi`

If you report a bug, the team will need certain pieces of information from you so that they can reproduce the bug, find the problem in the code, and fix it:

- A concise description of the problem, including any error messages that you received.
- The release of MIDP for Palm OS that you were using.
- The Palm OS device or device emulator you were using, and the Palm OS version that it was running.
- Sample code that shows the problem.

APPENDIX **A**

Frequently Asked Questions on MIDP for Palm OS

The following are frequently asked questions (FAQ) about MIDP for Palm OS. This list is kept updated at the following URL:

`http://java.sun.com/products/midp4palm/faq.html`

A.1 Question List

Clicking on a question will bring you to that question and its answer.

- Is MIDP for Palm OS free of charge? Can I redistribute it to third parties?
- Which versions of Palm OS are supported?
- In what languages is MIDP for Palm OS available?
- Does MIDP for Palm OS provide access to Palm OS native features?
- Does MIDP for Palm OS support the datagram or comm protocols?
- Does MIDP for Palm OS support server-side sockets?
- How do I create a button using MIDP for Palm OS?
- Why aren't all my abstract commands displayed as buttons on a Palm OS device?
- Does MIDP for Palm OS have an API to access the IR port of a Palm OS device?
- Is there an API that supports beaming arbitrary data through a MIDlet?
- How can I preinitialize an RMS database for my MIDlet?
- Does MIDP for Palm OS support data transfer with the cradle using HotSync?
- Does MIDP for Palm OS support international character sets?
- Does MIDP replace KVM as the Java interpreter on Palm OS?
- Can I have the source code for MIDP for Palm OS.?
- Where can I get the source code to the samples included in MIDP for Palm OS?
- Is there API documentation for MIDP for Palm OS?

A.2 Questions and Answers

1. Is MIDP for Palm OS free of charge? Can I redistribute it to third parties?

   You can download a copy of MIDP for Palm OS for evaluation and individual use.

   You must contact Sun to discuss licensing terms before distributing MIDP for Palm OS with an application or a Palm OS device. Send e-mail to `midp4palm-comments@sun.com` for more information.

2. Which versions of Palm OS are supported?

   MIDP for Palm OS is for Palm OS devices running Palm OS 3.5.x. More recent versions of Palm OS have not been tested thoroughly and are not officially supported. There is no plan to support earlier Palm OS versions.

   Most Palm OS devices either already run Palm OS 3.5 or can be upgraded to it. See `http://www.palmos.com/dev/tech/hardware/compare.html` for a breakdown of Palm OS versions on Palm OS devices.

3. In what languages is MIDP for Palm OS available?

   MIDP for Palm OS is available in English. Support for other locales might be added in the future.

4. Does MIDP for Palm OS provide access to Palm OS native features?

   MIDP for Palm OS does not have APIs for accessing native Palm features. For example, the following features are not supported:

   - Direct access to arbitrary Palm databases (datebook, phonebook, etc.)
   - Send/receive data using beaming.

   MIDP for Palm OS does support the standard MIDP RMS API for storing persistent data, however.

5. Does MIDP for Palm OS support the datagram or comm protocols?

   MIDP for Palm OS does not support the datagram or comm protocols at this time. The MIDP specification does not require their support, so MIDlets should not assume that this protocol exists on a MIDP-enabled device. (See `http://java.sun.com/j2me/docs` for the MIDP Specification.)

6. Does MIDP for Palm OS support server-side sockets?

   MIDP for Palm OS provides an unsupported implementation of basic sockets, but not server-side sockets. The MIDP specification does not require their support, so MIDlets should not assume that server-side sockets exist on a MIDP-enabled device. (See `http://java.sun.com/j2me/docs` for the MIDP Specification.)

7. How do I create a button using MIDP for Palm OS?

   The MIDP specification does not include support for buttons. Instead it provides abstract commands, which provide command actions while leaving the specifics of their presentation up to individual MIDP implementations. MIDP for Palm OS maps abstract commands to buttons and menus, as briefly described in the next question and detailed in the MIDP for Palm OS documentation.

   The advantage of abstract commands is portability. They enable MIDlets to use the same code to run on different devices, each of which may use different input methods. You create abstract commands with the `javax.microedition.lcdui.Command` class.

8. Why aren't all my abstract commands displayed as buttons on a Palm OS device?

   MIDP for Palm OS displays abstract commands by `TYPE` first, and then `PRIORITY`. Once the runtime generates this ordering, it places some commands in buttons on the screen. Due to the limited screen size of a Palm OS device, not all commands will be able to be placed in buttons. All commands will, however, be shown in the menus. Tap the menu button () of your Palm OS device to access the menus.

9. Does MIDP for Palm OS have an API to access the IR port of a Palm OS device?

   No, there is no API to access the IR port of a Palm OS device.

10. Is there an API that supports beaming arbitrary data through a MIDlet?

    There is no beaming API within MIDP. However, developers can make MIDlet suites beamable. If a MIDlet suite is beamable you can beam it from one Palm OS device to another, if the receiving device has also installed MIDP for Palm OS.

11. How can I preinitialize an RMS database for my MIDlet?

    Because of a variety of Palm OS naming issues, it is not possible to accurately predict the name that the underlying Palm OS database should have. Therefore you must include code in your MIDlet to create and populate the RMS database the first time the MIDlet is run.

12. Does MIDP for Palm OS support data transfer with the cradle using HotSync?

    MIDP for Palm OS supports standard backup of applications and application data at HotSync. It does not support HotSync for other purposes, however. Instead, to transfer data through the cradle:

    a. Use an application such as Mocha W32 PPP to establish a TCP/IP connection from the Palm OS device to your PC through the cradle.

    b. Use an HTTP connection to transfer data.

13. Does MIDP for Palm OS support international character sets?

    MIDP for Palm OS supports the Latin character sets, such as English, French, and Spanish. Support for other character sets might be added in a future release.

14. Does MIDP replace KVM as the Java interpreter on Palm OS?

    MIDP does not replace the K Virtual Machine (KVM), because MIDP is not a virtual machine. MIDP is a Java™ 2 Platform, Mobile Edition (J2ME™) Profile that sits on top of the Connected Limited Device Configuration (CLDC). CLDC includes KVM. MIDP, along with CLDC and KVM, provides a portable, small-footprint, standardized environment to run Java applications (MIDlets).

    In addition, the Palm overlay package (Spotless) is no longer available for download. The package was unsupported and is not part of the CLDC 1.0 specification.

    To develop a Java application for a Palm OS device, use MIDP for Palm OS.

15. Can I have the source code for MIDP for Palm OS.?

    No, but you can download the MIDP reference implementation source code at `http://java.sun.com/products/midp`. You can also contact Sun's Java Partner Engineering group at `http://www.sun.com/software/jpe/` about customizing MIDP for Palm OS for you on a contractual basis.

16. Where can I get the source code to the samples included in MIDP for Palm OS?

    Sample source code is included with the J2ME Wireless Toolkit, at `http://java.sun.com/products/j2mewtoolkit/`, and the MIDP reference implementation, at `http://java.sun.com/products/midp/`.

17. Is there API documentation for MIDP for Palm OS?

    MIDP for Palm OS is an implementation of MIDP for the Palm OS platform. The MIDP APIs are documented in the MIDP specification, which you can download from `http://java.sun.com/products/midp/`.

    Note that, since the goal of MIDP is to provide a common set of APIs across mobile information devices (phones, PDAs, pagers, and so on), MIDP for Palm OS has no Palm-specific APIs.

For more information on MIDP and its APIs, see to the MIDP product web site at http://java.sun.com/products/midp/ or the book, *Programming Wireless Devices with the Java™ 2 Platform, Micro Edition* by Roger Riggs, Antero Taivalsaari and Mark VandenBrink (Addison-Wesley 2001) promoted at http://java.sun.com/docs/books/j2mewireless/index.html.

APPENDIX **B**

Error Messages from `MakeMIDPApp`

This appendix lists the error messages that you could receive from `MakeMIDPApp`. It divides the error messages into two categories:

- JAD-related Errors
- JAR-related Errors

There is a number at the end of each error message. These numbers will also appear within the PRC Converter Tool. End-users can refer to these numbers when reporting problems to application providers or developers.

B.1 JAD-related Errors

- `The name of MIDlet suite is missing.` (13)
- `The vendor is missing.` (14)
- `The version is missing.` (15)
- `The URL for the JAR is missing.` (18)
- `The JAR size is missing.` (21)
- `The MIDlet suite name does not match the one in the JAR manifest.` (25)
- `The version does not match the one in the JAR manifest.` (26)
- `The vendor does not match the one in the JAR manifest.` (27)

B.2 JAR-related Errors

- `The name of MIDlet suite is missing.` (1000)
- `The vendor is missing.` (1010)
- `The version is missing.` (1020)
- `The MIDlet-<n> is missing.` (1050)

  This error message could appear because the manifest of a JAR file must have at least one `MIDlet-<n>` tag.
- `The JAR Manifest is missing.` (1060)
- `The configuration is missing from the manifest.` (1070)
- `The profile is missing from the manifest.` (1080)
- `The MIDlet suite name does not match the one in the JAR manifest.` (2000)
- `The version does not match the one in the JAR manifest.` (2010)
- `The vendor does not match the one in the JAR manifest.` (2020)
- `The JAR file was not found.` (9000)
- `The JAR file size = 0.` (9010)

Index