Oracle® Java ME Embedded
Getting Started Guide for the Windows 32 Platform
Release 3.2
E35132-01

September 2012
This book describes how to install and run Oracle Java ME Embedded software on the Windows 32 platform.
## Contents

### Preface

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audience</td>
<td>vii</td>
</tr>
<tr>
<td>Documentation Accessibility</td>
<td>vii</td>
</tr>
<tr>
<td>Related Documents</td>
<td>vii</td>
</tr>
<tr>
<td>Operating System Commands</td>
<td>vii</td>
</tr>
<tr>
<td>Shell Prompts</td>
<td>viii</td>
</tr>
<tr>
<td>Conventions</td>
<td>viii</td>
</tr>
</tbody>
</table>

### 1 Running the Oracle Java ME Embedded Software on Windows

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verifying the Java SE Platform</td>
<td>1-1</td>
</tr>
<tr>
<td>Finding the Java ME Embedded Emulator</td>
<td>1-1</td>
</tr>
<tr>
<td>Using the Java ME Embedded Emulator</td>
<td>1-2</td>
</tr>
<tr>
<td>The GPIO Tab</td>
<td>1-3</td>
</tr>
<tr>
<td>The I2C Tab</td>
<td>1-3</td>
</tr>
<tr>
<td>The SPI Tab</td>
<td>1-4</td>
</tr>
<tr>
<td>The MMIO Tab</td>
<td>1-5</td>
</tr>
<tr>
<td>Configuring the Emulated Device</td>
<td>1-6</td>
</tr>
<tr>
<td>Using the Device Manager</td>
<td>1-8</td>
</tr>
<tr>
<td>Running Sample Projects with the Emulator</td>
<td>1-8</td>
</tr>
<tr>
<td>GPIODemo</td>
<td>1-9</td>
</tr>
<tr>
<td>I2C Demo</td>
<td>1-9</td>
</tr>
<tr>
<td>NetworkDemoIMPNG</td>
<td>1-9</td>
</tr>
<tr>
<td>PDAPDemoIMPNG</td>
<td>1-10</td>
</tr>
<tr>
<td>Other Common Emulator Commands</td>
<td>1-10</td>
</tr>
</tbody>
</table>

### 2 Using the Oracle Java ME Embedded Software with NetBeans

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing on NetBeans</td>
<td>2-1</td>
</tr>
<tr>
<td>Creating a New Project</td>
<td>2-2</td>
</tr>
<tr>
<td>Including the Oracle Java ME Embedded Class Libraries</td>
<td>2-5</td>
</tr>
</tbody>
</table>

### Glossary

### Index
List of Examples

2–1  Skeleton Code for an Oracle Java ME Embedded Project.................................................... 2–4
List of Figures

1–1  The Java ME Embedded Emulator Screen on Windows............................................. 1-2
1–2  Emulator General Purpose I/O (GPIO) Tab ............................................................... 1-3
1–3  Inter-Integrated Circuit (I2C) Tab .............................................................................. 1-4
1–4  Serial Peripheral Interface Tab .................................................................................. 1-5
1–5  Memory-Mapped IO (MMIO) Tab ............................................................................... 1-6
1–6  GPIO Tab of the External Events Generator ............................................................. 1-7
1–7  MMIO Tab of the External Events Generator ............................................................ 1-8
2–1  NetBeans Plugins Dialog .......................................................................................... 2-1
2–2  Add Java Platform Dialog ......................................................................................... 2-2
2–3  New Project Dialog .................................................................................................... 2-3
2–4  New Embedded Application Dialog ........................................................................... 2-3
2–5  Modifying the Project Properties ............................................................................... 2-5
This book describes how to install Oracle Java ME Embedded software onto the Windows 32 platform. Readers using this guide should be familiar with the Information Module Profile - Next Generation (IMP-NG) 1.0 Specification.

Audience

This document is intended for developers who want to run Oracle Java ME Embedded on a Windows platform.

Documentation Accessibility

For information about Oracle’s commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For a complete list of documents with the Oracle Java ME Embedded software, see the Release Notes.

Operating System Commands

This document does not contain information on basic commands and procedures such as opening a terminal window, changing directories, and setting environment variables. See the software documentation that you received with your system for this information.
Shell Prompts

<table>
<thead>
<tr>
<th>Shell</th>
<th>Prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bourne shell and Korn shell</td>
<td>$</td>
</tr>
<tr>
<td>Windows</td>
<td>directory&gt;</td>
</tr>
</tbody>
</table>

Conventions

The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>boldface</td>
<td>Boldface type indicates graphical user interface elements associated</td>
</tr>
<tr>
<td></td>
<td>with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td>italic</td>
<td>Italic type indicates book titles, emphasis, or placeholder variables</td>
</tr>
<tr>
<td></td>
<td>for which you supply particular values.</td>
</tr>
<tr>
<td>monospace</td>
<td>Monospace type indicates commands within a paragraph, URLs, code in</td>
</tr>
<tr>
<td></td>
<td>examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
Running the Oracle Java ME Embedded Software on Windows

The Windows emulation environment provides you with a platform for testing and running IMP-NG IMlet suites without having to install those IMlet suites onto an embedded device. This chapter describes running the emulation environment.

This chapter also shows you how to run the software on a Microsoft Windows system using the emulation environment that is provided in the reference binary. It assumes that you have downloaded the reference binary onto a Windows-based desktop platform.

Verifying the Java SE Platform

The first step is to verify your current Java SE Platform. Tools that are provided with the software are based on the Java Platform, Standard Edition (Java SE), Version 7, Update 5. To properly use these tools, you must have a compatible version of the Java Runtime Environment (JRE) installed on your computer and set as the default version.

To check if you have an appropriate version of the Java SE platform installed on your computer, use the `java -version` command, as shown here:

```
C:\> java -version
java version '1.7.0_03'
Java(TM) SE Runtime Environment (build 1.7.0_03-b05)
Java HotSpot(TM) Client VM (build 22.1-b02, mixed mode, sharing)
```

The version number shown in the output should be version 1.7.0_01 or higher.

If you need to install a compatible version of the Java SE platform, you can download it from the following address:

http://www.oracle.com/technetwork/java/index.html

Finding the Java ME Embedded Emulator

You can find the Java ME Embedded emulator within the `bin` directory of the Java ME SDK 3.2 installation.

For example, if the Java ME SDK 3.2 is installed in `C:\Java_ME_platform_SDK_3.2`, then the emulator would be located at `C:\Java_ME_platform_SDK_3.2\bin\emulator.exe`
Using the Java ME Embedded Emulator

To start the emulator and open an emulator window, enter the following command from the bin directory:

```
EmulatorDir\emulator.exe -Xdevice:IMPNGPhone1 -Xdescriptor:location_of_jad_file
```

The emulator’s main screen appears as shown in Figure 1–1. The first tab in the emulator, AMS (Application Management System), displays which Java ME Embedded applications are installed, including those that are currently running or stopped. You can use the buttons on the right side to install or update additional applications, obtain information about the currently selected application, start or stop an application, or remove (uninstall) an application from the AMS.

The current status of each Java ME Embedded application is shown in the panel on the left side as shown in Figure 1–1.

**Figure 1–1  The Java ME Embedded Emulator Screen on Windows**

![Java ME Embedded Emulator Screen](image)

**Note:** You can run the emulator command without the .exe extension. It works both ways, with the extension and without.
If no -Xdevice option is specified, the IMPNGPhone1 device is started by default.

For information on other emulator commands and options, see "Other Common Emulator Commands" on page 1-10.

The GPIO Tab

The GPIO tab lists the emulator’s current General Purpose I/O (GPIO) pins and ports, and their directional states (input or output). For GPIO pins, the current value is shown as a color-coded circle on the right side: high is colored green, while low is red. For GPIO ports, the maximum and current value is shown. If a GPIO port or pin is named, it is shown in this tab as well.

The GPIO tab is shown in Figure 1–2:

**Figure 1–2 Emulator General Purpose I/O (GPIO) Tab**

![Emulator General Purpose I/O (GPIO) Tab](image)

The I2C Tab

The Inter-Integrated Circuit (I2C) tab emulates a simple peripheral slave device that echoes back any data that is sent to it. Both the sent and received data are shown in their appropriate window panes, as shown in Figure 1–3:
The SPI Tab

The Serial Peripheral Interface (SPI) tab is similar to the I2C tab. It emulates a simple peripheral slave device that echoes back any data that is sent to it. Both the sent and received data are shown in their appropriate tabs, as shown in Figure 1–4:
The MMIO Tab

Finally, the MMIO tab emulates the Memory-Mapped I/O (MMIO) interface bus. The MMIO interface creates four separate devices that can be used for testing: TEST_DEVICE, WDOG_LOG, RTC, and BIG_ENDIAN_DEVICE. Each type of device displays its appropriate block configuration in an information table, as shown in Figure 1–5:
Configuring the Emulated Device

Selecting the Device menu allows the user to configure several items on the device. The Access Points tab allows the user to configure the settings returned by various methods of the Access Point API, including Wi-Fi and carrier networks.

The File Connection tab allows the user to mount external file systems. In addition, the Location tab allows the device to specify the simulated location, orientation, and speed of the device. The SIM 0 and SIM 1 tabs allow the user to specify the hardware values of one of two subscriber identity module (SIM) cards that are typically installed in mobile devices.

To generate input events for General Purpose I/O (GPIO), select the GPIO menu item under the Device menu. This action raises the External Events Generator window. Here, you can toggle the value of each of the pins from high to low and vice versa, and use a wave generator to simulate a more complex signal to the emulator. The GPIO External Events Generator is shown in Figure 1–6:
In addition, selecting the MMIO tab of the External Events Generator allows the user to simulate sending input event IDs from one of the four different peripheral device types. This tab is shown in Figure 1–7:
Using the Device Manager

The Device Manager is started the first time the emulator is started. Its purpose is to manage multiple devices and device storage, so that multiple devices can be emulated without overwriting or colliding with each other.

When the Device Manager is started, it stays running as an icon in your system tray. To interact with the Device Manager, right-click the icon to display the Device Manager menu.

Running Sample Projects with the Emulator

The Oracle Java ME Embedded platform comes with a number of sample applications. This section describes how to use demos created specifically for the Oracle Java ME Embedded platform. Because IMP-NG is headless, the only user interface is to observe application status in the emulator’s external events generator, or in the Output window (or the console if you execute the demo from the command line).

With the exception of I2CDemo, the sample projects provided with the Oracle Java ME Embedded release can be run on the emulator or on a real device.

Note that in the _policy.txt file of the distribution (typically located in the runtimes/impng/lib directory), the developer may need to add the following entries to avoid security exceptions when running the samples:

To both the domains:

- domain: untrusted, unsecured
- domain: unidentified_third_party, unsecured
Add the following entry:

allow: device_access

**GPIODemo**

This demo can be run on an IMP-NG emulator using the external events generator:

- Click the GPIO tab. This view approximates the device actions.
- Choose Device > GPIO to open the external events generator, and click the GPIO tab. A single click on a button turns on LEDs indicating the button pushed and the pin affected. This information is also written to the Output window.
  
  Beneath each pin you can click the blue wave button to open the wave generator. The wave generator simulates the frequency and duration of the signal to the LED.

  - Press Pin 5 (button 1) to turn on LED 1, press again to turn off LED 1.
  - Press Pin 6 (button 2) to turn on LED 2, press again to turn off LED 2.
  - Press Pin 7 (button 3) and check whether PORT 1's output value is 3. Press PIN 7 and check whether PORT 1's output value is 0.

**I2CDemo**

This demo is designed to work with the IMP-NG runtime for Windows 32. It has no user interaction.

- Launch the I2C demo.
- In the emulator, click the I2C tab.

  The demo acquires a slave named I2C_Echo, writes data to the slave, and retrieves it. The demo is successful if the Sent Data and Received Data matches.

**NetworkDemoIMPNG**

This demo can be configured as a server or as a client by editing the application descriptor. You launch two instances of this demo, the first one acts as a server and the second one acts as a client. The client instance attempts to connect to the server instance and if the connection is successful they exchange a message.

- Create two instance projects of the NetworkDemoIMPNG sample project.
- Right click on the first project and select Properties. In the Platform category choose the device IMPNGPhone1. In the Application Description category set the value of the property Oracle-Demo-Network-Mode to **Server** and click OK.
- Launch the first project. It opens on the emulator IMPNGPhone1 and waits for a connection.
- Right click on the second project and select Properties. In the Platform category choose the device IMPNGPhone2. In the Application Description category set the value of the property Oracle-Demo-Network-Mode to **Client** and click OK.
- Launch the second project. It opens on the emulator IMPNGPhone2.
- The client attempts to connect to the server. If successful, you see the following in the output tab of the first project (the server):

  Connection accepted
  Message received - Client messages
The output of the second project (the client) shows the following:

Connected to server localhost on port 5000
Message received - Server String

PDAPDemoIMPNG

Follow these steps to run the demo on the IMP-NG emulator:

- Create test files and directories inside the emulator's file system:
  Documents and Settings\user\javame-sdk\version\work\IMPNGPhone1\appdb\filesystem\root1
- Right click on the project and select Properties. In the Platform category choose the device IMPNGPhone1 and click OK.
- Launch the project. It runs on IMPNGPhone1.
- On the emulator menu, select Device > File Connection to see a list of mounted file systems.
- Open a terminal emulator and create a raw connection to localhost on port 5001.
- A command line opens where you can browse the emulator’s file system. You can use the following commands:
  - cd - change directory
  - ls - list information about the FILES for the current directory
  - new - create new file or directory
  - prop - show properties of a file
  - rm - remove the file
  - view - View a file's content

Other Common Emulator Commands

This section provides emulator commands.

Note: For a full list of Emulator commands, type: emulator -help.

- To show a list of installed IMlets, use the -Xjam:list subcommand:
  EmulatorDir> emulator -Xjam:list
- To see a list of all supported devices, use the -Xquery subcommand:
  EmulatorDir> emulator -Xquery
- To install a JAD over the air (OTA) and execute an IMlet, use the -Xjam:install subcommand:
  EmulatorDir> emulator -Xjam:install=<JAD_file_URL>
  For example:
  EmulatorDir> emulator -Xjam:install=http://www.appstore.com/TestJAD.jad
- To run an installed IMlet, use the -Xjam:run subcommand:
Provide either the storage name or storage number for the IMlet to run. You can get the storage name and storage number from the list of IMlets shown by the -Xjam:list subcommand.

■ To remove an installed IMlet, use the -Xjam:remove subcommand:

    EmulatorDir>emulator -Xjam:remove=[storage_name | storage_number | all]

Provide either the storage name or storage number for the IMlet to remove. To remove all IMlets, use all. You can get the storage name and storage number from the list of IMlets shown by the -Xjam:list subcommand.

■ To install a JAD file, execute the IMlet locally, and remove the IMlet when completed, use the -Xdescriptor subcommand:

    EmulatorDir>emulator -Xdescriptor:<JAD_file_name>

■ To set an IMlet’s security domain, use the -Xdomain subcommand:

    EmulatorDir>emulator -Xdomain:<domain_name>

■ To run in autotest mode, use the -Xautotest subcommand:

    EmulatorDir>emulator -Xautotest:<JAD_file_URL>

For example:

    EmulatorDir>emulator -Xautotest:http://127.0.0.1:8080/getNextApp.jad
This chapter discusses how to install the Oracle Java ME Embedded software for the Windows platform on the NetBeans Integrated Development Environment (IDE).

The examples below use NetBeans 7.1, although it works with NetBeans 7.0 as well. With the NetBeans integrated development environment, you can create and test mobile applications using a graphical development environment.

Installing on NetBeans

This section walks you through installing the Oracle Java ME Embedded software.

1. Start the NetBeans IDE and choose the Tools>Plugins menu item.
   A dialog box appears, as shown in Figure 2–1:

   **Figure 2–1  NetBeans Plugins Dialog**

   ![NetBeans Plugins Dialog](image)

   2. Choose the Installed tab, and ensure that the Java ME plugin is activated.
If it is not, select Java ME from the list and press the Activate button. Activating the Java ME plugin requires installing the Java Profiler plugin as well.

3. Close the NetBeans Plugin dialog box when completed.

4. Install the Oracle Java ME Embedded platform in NetBeans.
   a. Choose the Tools>Java Platforms menu item.
   b. Add the new Oracle Java ME Embedded platform by pressing the “Add Platform...” button in the lower-left corner of the dialog.

   This step shows the dialog in Figure 2–2:

   **Figure 2–2 Add Java Platform Dialog**

   ![Add Java Platform Dialog](image)

   c. Choose the second option, “Java ME MIDP Platform Emulator”, as the Oracle Java ME Embedded is a Java ME MIDP Platform Emulator, and press the Next button.

d. NetBeans lists all known Java ME MIDP platforms. If the Oracle Java ME Embedded is not shown, press the “Find More Java ME Platform Folders...” button near the bottom. Then, select the distribution folder and press the Open button.

   NetBeans automatically detects the Oracle Java ME Embedded platform.

e. Press the Next button.

   If successful, NetBeans presents a report of the detected platform.

f. Press the Finish button.

   The platform is added to the Java Platform Manager.

g. Finally, close the Java Platform Manager dialog.

**Creating a New Project**

This section walks you through creating a new project using the Oracle Java ME Embedded platform.
1. Choose File>New Project. The New Project dialog appears, as shown in Figure 2–3:

**Figure 2–3  New Project Dialog**

2. Choose Java ME from the Categories list, and Mobile Application from the Projects list, then press the Next button.

   The dialog shows the “Name and Location” panel.

3. Choose an appropriate name and location for the project. Be sure to uncheck the "Create Hello MIDlet" option, as this may create a sample project that uses the mobile libraries, and then press Next to arrive at the panel shown in Figure 2–4:

**Figure 2–4  New Embedded Application Dialog**
Here, you can choose the desired platform for the Java Wireless Client project.

4. Choose “Oracle Java ME Embedded 3.2” in the drop-down selection at the top of the dialog.

5. Press Finish.

A new Oracle Java ME Embedded project is created. At this point, you can use the skeleton code shown in Example 2–1 to start building an embedded project.

**Example 2–1  Skeleton Code for an Oracle Java ME Embedded Project**

```java
import com.oracle.deviceaccess.PeripheralTypeNotSupportedException;
import javax.microedition.midlet.MIDlet;

public class GPIODemo extends MIDlet {

    boolean bFirst = false;
    boolean loopFlag = true;

    public void startApp() {
        if (bFirst == false) {
            try {
                // Perform startup operations
                } catch (PeripheralTypeNotSupportedException ex) {
                    ex.printStackTrace();
                    return;
                } catch (Exception ex) {
                    ex.printStackTrace();
                    return;
                }
            }
            bFirst = true;
        } else {
            System.out.println("GPIO Demo is already started..");
        }

        // Start program here, including accessing peripheral devices
    }

    public void pauseApp() {
        // Pause the application
    }

    public void destroyApp(boolean unconditional) {
        bFirst = false;

        // Close all resources that have been opened
    }
}
```

Note that in the _policy.txt_ file of the Oracle Java ME Embedded distribution (typically located in the _runtimes/impng/lib_ directory), the developer must add the following entries to avoid security exceptions when running the example:
To both the domains:
domain: untrusted, unsecured
domain: unidentified_third_party, unsecured
Add the following entry:
allow: device_access

**Including the Oracle Java ME Embedded Class Libraries**

The final step involves modifying the project properties to include the desired Oracle Java ME Embedded class definitions. The following packages are available.

- Application Management System (AMS) API
- CLDC Logging API
- File Connection and PIM Optional Packages 1.0
- Location Based API
- OMDA Access Point API
- Security and Trust Services API for J2ME
- Wireless Messaging API

Here are the steps to include optional packages.

1. Choose File>Project Properties for the project you just created.

You should see a dialog similar to Figure 2–5:

*Figure 2–5  Modifying the Project Properties*
2. Select the Platform option from the list on the left side, and note the optional packages on the bottom right side of the dialog.

3. Ensure that any libraries used by the embedded application are checked.

4. Click OK to close the dialog.

   At this point, you can compile and run the newly-created Oracle Java ME Embedded project. If it is successful, you should see the Oracle Java ME Embedded emulator.
3GPP
Third Generation Partnership Project. A collaboration between groups of telecommunications associations, for the purpose of making a globally applicable third generation (3G) mobile phone system specification.

Access Point
A network-connectivity configuration that is predefined on a device. An access point can represent different network profiles for the same bearer type, such as different cellular network access point names (APN), or for different bearers that may be available on a device, such as WiFi or bluetooth.

AID
Application Identifier. A string used to uniquely identify card applet applications and certain types of files in card file systems. An AID consists of two distinct pieces: a 5-byte RID (resource identifier) and a 0 to 11-byte PIX (proprietary identifier extension).

AMS
Application Management Service. The system functionality that completes tasks such as installing applications, updating applications, and switching foregrounds.

APDU
Application Protocol Data Unit. A communication mechanism used by SIM Cards and smart cards to communicate with card reader software or a card reader device.

API
Application Programming Interface. A set of classes used by programmers to write applications that provide standard methods and interfaces and eliminate the need for programmers to reinvent commonly used code.

Applet
A small program that runs in the APDU application environment.

Application list
The screen that lists all of the installed applications. The user gets to this screen by pressing the Apps soft key on the home screen. The application list uses text color to show which applications are running. It also provides a system menu that enables the user to perform application management tasks on the highlighted application.
ARM
A reduced instruction set computer (RISC) instruction set architecture (ISA) developed by ARM Holdings.

AXF
ARM Executable Format is an ARM executable image generated by ARM tools.

Background
An application state in which the application does not receive events from its input stream and its display is not rendered to the screen.

BIP
Bearer Independent Protocol. Allows an application on the SIM Card to establish a data channel with a terminal (that is, an Oracle Java Wireless Client-enabled handset), and through the terminal to a remote server in the network.

CDC
Connected Device Configuration. A Java ME platform configuration for devices. It requires a minimum of 2 megabytes of memory and a network connection that is always on.

CDMA
Code Division Multiple Access. A mobile telephone network standard used primarily in the United States and Canada as an alternative to GSM.

CLDC
Connected Limited Device Configuration. A Java ME platform configuration for devices with limited memory and network connectivity. It uses a low-footprint Java virtual machine such as the CLDC HotSpot Implementation, and several minimalist Java platform APIs for application services.

Configuration
Defines the minimum Java runtime environment (for example, the combination of a Java virtual machine and a core set of Java platform APIs) for a family of Java ME platform devices.

ETSI
European Telecommunications Standards Institute. An independent, non-profit standardizations group responsible for the standardization of Information and Communication Technologies (TCI) within Europe. Although based in Europe, it carries worldwide influence in the telecommunications industry.

Foreground
The application state in which the application is rendered to the device display and the input stream is passed to it.

Foreground switching
Changing which application is in the foreground by shifting the focus from one application to another.
GCF
Generic Connection Framework. A part of CLDC, it is a Java ME API consisting of a hierarchy of interfaces and classes to create connections (such as HTTP, datagram, or streams) and perform I/O.

GSM
Global System for Mobile Communications. A 3G mobile telephone network standard used widely in Europe, Asia, and other parts of the world.

HTTP
HyperText Transfer Protocol. The most commonly used Internet protocol, based on TCP/IP that is used to fetch documents and other hypertext objects from remote hosts.

HTTPS

ICCID
Integrated Circuit Card Identification. The unique serial number assigned to an individual SIM Card.

IMP-NG
Information Module Profile Next Generation. A profile for embedded "headless" devices, the specification for JSR 228, is a subset of MIDP 2.0 that leverages the latest security and networking types and APIs of MIDP 2.0 but does not include UI APIs.

IMEI
International Mobile Equipment Identifier. A number unique to every mobile phone. It is used by a GSM or UMTS network to identify valid devices and can be used to stop a stolen or blocked phone from accessing the network. It is usually printed inside the battery compartment of the phone.

IMlet
An application written for IMP-NG. An IMlet does not differ from MIDP 2.0 MIDlet, except by the fact that an IMlet can not refer to MIDP classes that are not part of IMP(-NG). An IMlet can only use the APIs defined by the IMP(-NG) and CLDC specifications.

IMSI
International Mobile Subscriber Identity. A unique number associated with all GSM and UMTS network mobile phone users. It is stored on the SIM Card inside a phone and is used to identify itself to the network.

JAD file
Java Application Descriptor file. A file provided in a MIDlet suite that contains attributes used by application management software (AMS) to manage the MIDlet's life cycle, and other application-specific attributes used by the MIDlet suite itself.

JAR file
Java Archive file. A platform-independent file format that aggregates many files into one. Multiple applications written in the Java programming language and their required components (class files, images, sounds, and other resource files) can be bundled in a JAR file and provided as part of a MIDlet suite.
Java ME platform
Java Platform, Micro Edition. A group of specifications and technologies that pertain to running the Java platform on small devices, such as cell phones, pagers, PDAs, and set-top boxes. More specifically, the Java ME platform consists of a configuration (such as CLDC or CDC) and a profile (such as MIDP or Personal Basis Profile) tailored to a specific class of device.

Java Specification Request (JSR)
A proposal for developing new Java platform technology, which is reviewed, developed, and finalized into a formal specification by the JCP program.

Java Virtual Machine
A software “execution engine” that safely and compatibly executes the byte codes in Java class files on a microprocessor.

KVM
A Java virtual machine designed to run in small devices, such as cell phones and pagers. The CLDC configuration was initially designed to run in a KVM.

LCDUI
Liquid Crystal Display User Interface. A user interface toolkit for interacting with Liquid Crystal Display (LCD) screens in small devices. More generally, a shorthand way of referring to the MIDP user interface APIs.

LWUIT
Lightweight User Interface Toolkit (LWUIT). A versatile and compact API for creating attractive mobile user interfaces. LWUIT provides sophisticated Swing-like capabilities and employs a similar design as Swing, but without the tremendous power and complexity. LWUIT makes it easy to apply consistent look-and-feel’s, called themes, across disparate devices using an advanced graphical user interface (GUI) customization tool.

MIDlet
An application written for MIDP.

MIDlet suite
A way of packaging one or more midlets for easy distribution and use. Each MIDlet suite contains a Java application descriptor file (.jad), which lists the class names and files names for each MIDlet, and a Java Archive file (.jar), which contains the class files and resource files for each MIDlet.

MIDP
Mobile Information Device Profile. A specification for a Java ME platform profile, running on top of a CLDC configuration that provides APIs for application life cycle, user interface, networking, and persistent storage in small devices.

MSISDN
Mobile Station Integrated Services Digital Network. A number uniquely identifying a subscription in a GSM or UMTS mobile network. It is the telephone number to the SIM Card in a mobile phone and used for voice, FAX, SMS, and data services.
Obfuscation
A technique used to complicate code by making it harder to understand when it is decompiled. Obfuscation makes it harder to reverse-engineer applications and therefore, steal them.

Optional Package
A set of Java ME platform APIs that provides additional functionality by extending the runtime capabilities of an existing configuration and profile.

Oracle Java Device Test Suite
A set of Java programming language tests developed specifically for the wireless marketplace, providing targeted, standardized testing for CLDC and MIDP on small and handheld devices.

Preemption
Taking a resource, such as the foreground, from another application.

Preverification
Due to limited memory and processing power on small devices, the process of verifying Java technology classes is split into two parts. The first part is preverification which is done off-device using the preverify tool. The second part, which is verification, occurs on the device at runtime.

Profile
A set of APIs added to a configuration to support specific uses of a mobile device. Along with its underlying configuration, a profile defines a complete and self-contained application environment.

Provisioning
A mechanism for providing services, data, or both to a mobile device over a network.

Push Registry
The list of inbound connections, across which entities can push data, maintained by the Oracle Java Wireless Client software. Each item in the list contains the URL (protocol, host, and port) for the connection, the entity permitted to push data through the connection, and the application that receives the connection.

RL-ARM
Refers to the Real-Time Library that is a group of tightly coupled libraries designed to solve the real-time and communication challenges of embedded systems based on ARM processor-based microcontroller devices.

RMI
Remote Method Invocation. A feature of Java SE technology that enables Java technology objects running in one virtual machine to seamlessly invoke objects running in another virtual machine.

RMS
Record Management System. A simple record-oriented database that enables a MIDlet to persistently store information and retrieve it later. MIDlets can also use the RMS to share data.
RTSP
Real Time Streaming Protocol. A network control protocol designed to control streaming media servers and media sessions.

SCWS
Smart Card Web Server. A web server embedded in a smart card (such as a SIM Card) that allows HTTP transactions with the card.

SIM
Subscriber Identity Module or Subscriber Identification Module. An integrated circuit embedded into a removable SIM card that securely stores the International Mobile Subscriber Identity (IMSI) and the related key used to identify and authenticate subscribers on mobile telephony devices such as mobile phones and computers.

Smart Card
A card that stores and processes information through the electronic circuits embedded in silicon in the substrate of its body. Smart cards carry both processing power and information. A SIM Card is a special kind of smart card for use in a mobile device.

SMS
Short Message Service. A protocol allowing transmission of short text-based messages over a wireless network. SMS messaging is the most widely-used data application in the world.

SMSC
Short Message Service Center. The SMSC routes messages and regulates SMS traffic. When an SMS message is sent, it goes to an SMS center first, then gets forwarded to the destination. If the destination is unavailable (for example, the recipient’s handset is turned off), the message is stored in the SMSC until the recipient becomes available.

SOAP
Simple Object Access Protocol. An XML-based protocol that enables objects of any type to communicate in a distributed environment, it is most commonly used to develop web services.

SSL
Secure Sockets Layer. A protocol for transmitting data over the Internet using encryption and authentication, including the use of digital certificates and both public and private keys.

SVM
Single Virtual Machine. A software mode that can run only one MIDlet or IMlet at a time.

Task
At the platform level, each separate application that runs within a single Java virtual machine is called a task. The API used to instantiate each task is a stripped-down version of the Isolate API defined in JSR 121.

TCP/IP
Transmission Control Protocol/Internet Protocol. A fundamental Internet protocol that provides for reliable delivery of streams of data from one host to another.
**Terminal Profile**

Device characteristics of a handset (terminal) passed from the handset to the SIM Card along with the **IMEI** at SIM Card initialization. The terminal profile tells the SIM Card what values are supported by the device.

**UICC**

Universal Integrated Circuit Card. The smart card used in mobile terminals in **GSM** and **UMTS** networks. The UICC ensures the integrity and security of personal data on the card.

**UMTS**

Universal Mobile Telecommunications System. A third-generation (3G) mobile communications technology. It utilizes the radio spectrum in a fundamentally different way than **GSM**.

**URI**

Uniform Resource Identifier. A compact string of characters used to identify or name an abstract or physical resource. A URI can be further classified as a uniform resource locator (URL), a uniform resource name (URN), or both.

**USAT**

Universal SIM Application Toolkit. A software development kit intended for 3G networks. It enables **USIM** to initiate actions that can be used for various value-added services, such as those required for banking and other privacy related applications. USAT is defined in standard 3GPP 31.111 for 3G.

**USIM**

Universal Subscriber Identity Module. An updated version of a **SIM** designed for use over 3G networks. USIM is able to process small applications securely using better cryptographic authentication and stronger keys. Larger memory on USIM enables the addition of thousands of contact details including subscriber information, contact details, and other custom settings.

**WAE**

Wireless Application Environment. It provides an application framework for small devices, by leveraging other technologies such as Wireless Application Protocol (**WAP**), Wireless Transaction Protocol (**WTP**), and Wireless Session Protocol (**WSP**).

**WAP**

Wireless Application Protocol. A protocol for transmitting data between a server and a client (such as a cell phone) over a wireless network. WAP in the wireless world is analogous to **HTTP** in the World Wide Web.

**WCDMA**

Wideband Code Division Multiple Access. A detailed protocol that defines how a mobile phone communicates with the tower, how its signals are modulated, how datagrams are structured, and how system interfaces are specified.

**WMA**

Wireless Messaging API. A set of classes for sending and receiving Short Message Service (**SMS**) messages.
XML Schema

A set of rules to which an XML document must conform to be considered valid.
Index

D
device manager, 1-8

E
emulator
  commands, 1-10
  starting, 1-1