Oracle® Java ME Embedded

Getting Started Guide for the Windows Platform

Release 3.3

E35132-02

June 2013

This book describes how to use Oracle Java ME SDK to develop embedded applications, using both the NetBeans and Eclipse Integrated Development Environments, on a Windows XP or Windows 7 platform.

Oracle Java ME Embedded Getting Started Guide for the Windows Platform, Release 3.3

E35132-02

Contents

List of Figures

List of Examples

Preface

This book describes how to use the Oracle Java ME SDK 3.3 platform to quickly develop embedded applications on both the NetBeans and Eclipse Integrated Development Environments (IDEs). The Oracle Java ME SDK 3.3 platform contains a complete implementation of the Oracle Java ME Embedded 3.3 software runtime.

Together, these products support embedded software development on:

- Windows 32 platform in emulation (this guide)
- Raspberry Pi platform (*Oracle Java ME Embedded Getting Started Guide for the Reference Platform (Raspberry Pi*))
- Keil MCBSTM32F200 platform, running the RTX operating system (*Oracle Java ME Embedded Getting Started Guide for the Reference Platform (Keil))*

Audience

This document is intended for embedded developers who want to develop applications on Oracle Java ME SDK on a Windows XP or Windows 7 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, 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

Shell	Prompt
Bourne shell and Korn shell	`$`
Windows	*directory*>

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Before You Begin

The Oracle Java ME SDK 3.3 platform is a sophisticated and useful tool for programmers who want to develop embedded applications. The Oracle Java ME SDK 3.3 platform can be used successfully and easily with the NetBeans 7.3 Integrated Development Environment, as well as the Eclipse (Indigo 3.7 or Juno 4.2) IDE.

Note: The Oracle Java ME Embedded 3.3 runtime is fully implemented inside Oracle Java ME SDK 3.3 software. So, there is no need for you to independently install Oracle Java ME Embedded 3.3 software.

This chapter provides information you need to ensure that your Microsoft Windows XP (32-bit) or Windows 7 (32-bit or 64-bit) platform is correctly set up for working with Oracle Java ME SDK. Both Windows XP and Windows 7 must include the most recent Microsoft service packs.

Installing the Java SE Platform

To properly run the Oracle Java ME SDK software and its associated Tools, you must have Java Platform, Standard Edition (Java SE), Version 7, Update 21 (or later) installed on your computer.

This guide assumes you have already installed the Java SE platform. If you have not installed Java 7, Update 21, you can download it from the following location:

`http://www.oracle.com/technetwork/java/javase/downloads`

The installation of Eclipse Indigo 3.7 requires Java 6, Update 17 as the reference platform. It may work with other Java versions, but it is not recommended by Eclipse. To get Java 6, Update 17:

`http://www.oracle.com/technetwork/java/javase/archive-139210.html`

The Java SE platform must also be in your PATH.

Setting and Verifying Your Java SE PATH

To verify if Java SE platform is set in your PATH:

1. In the Windows command line, type:

   C:\>**echo %PATH%**

2. If Java SE is properly installed, you see a path to the default installation directory:

   C:\>Program Files\Java\jdk1.7.0_x

3. If not, you need to add Java SE to your PATH.

> **Note:** Setting the PATH may require using a Windows short name. To see top-level Windows short names, type C:\>**dir /x**

4. Set the Java SE variable, JDK_DIR:

   C:\>**set JDK_DIR=C:\Program Files\Java\jdk1.7.0_21**

5. Add JDK_DIR to your PATH:

   C:\>**set PATH=%PATH%;%JDK_DIR%\bin**

6. To verify the version of your Java SE platform, type:

   C\:>**java -version**

The version number shown in the output should be version 1.7.0_21 or higher.

Installing the Oracle Java ME SDK Platform

Follow these steps to install the Oracle Java ME SDK 3.3.

1. If you have previously installed an earlier version of Oracle Java ME SDK, uninstall the previous version as shown below.

   > **Note:** If you are installing Oracle Java ME SDK for the first time, skip to Step 2.

   - If you have Oracle Java ME SDK data to save, copy it to a safe location before continuing.
   - In the Windows system tray, right click the emulator icon and choose Exit.
   - From the Windows Programs menu, select the previous version and choose Uninstall from the submenu. The Installer opens.
   - On the first page check the option to remove the user data directory.
   - Follow the prompts.

2. Download the SDK from:

   `http://www.oracle.com/technetwork/java/javame/javamobile/download/sdk`

3. Double-click the executable. When the installer starts, follow the prompts.

Installing and the Starting the NetBeans IDE

If you do not already have the NetBeans 7.3 IDE installed on your system, you can download it here:

`http://dlc.sun.com.edgesuite.net/netbeans/7.3/final`

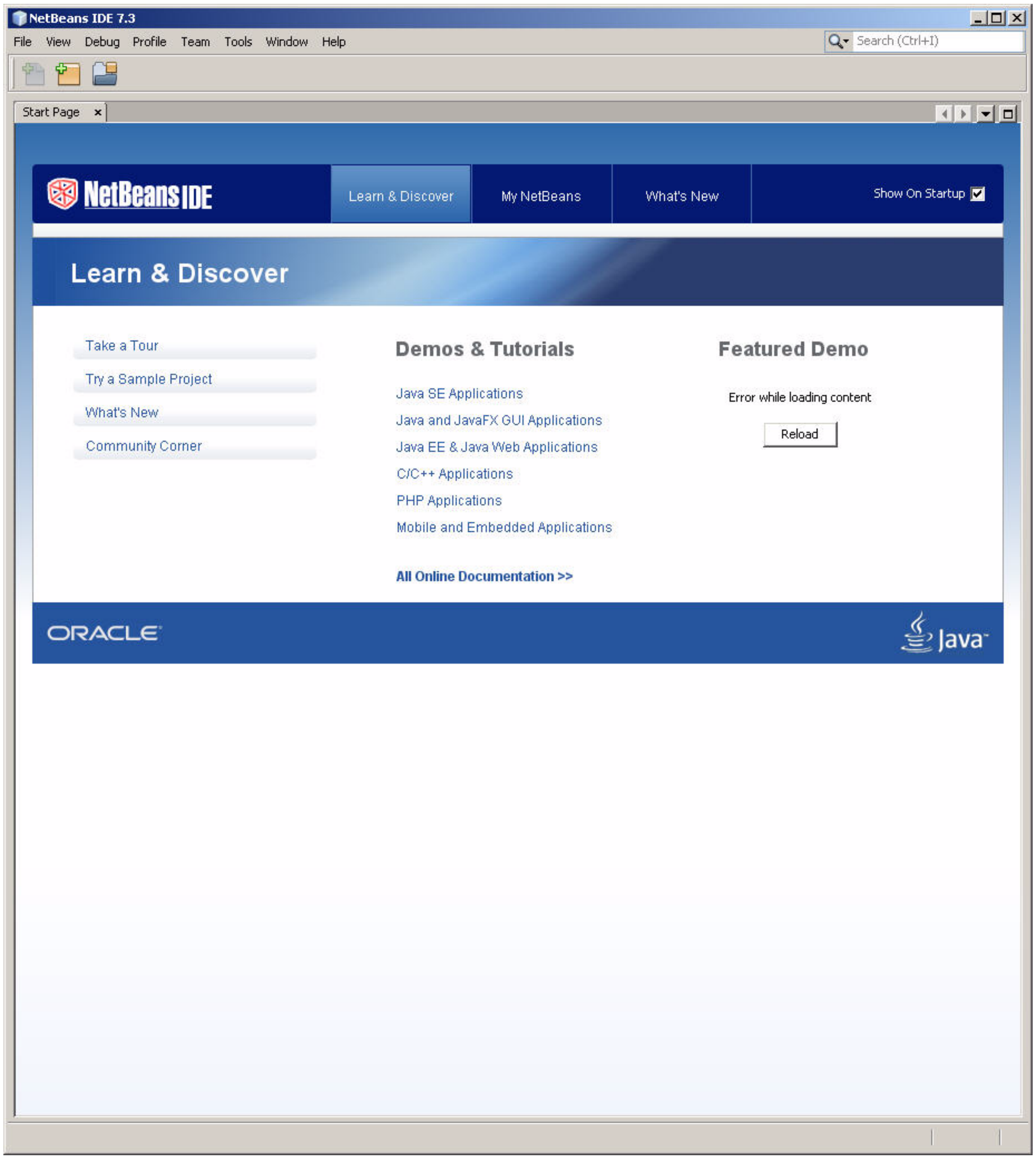
Once you have downloaded the NetBeans installation module, do the following.

1. Double click the NetBeans executable file, `netbeans-7.3-windows.exe`.
2. When the NetBeans Installer starts, follow the prompts.

   When the installation is complete, NetBeans creates a shortcut on your desktop.

3. Double click the NetBeans IDE 7.3 shortcut to launch NetBeans.

Figure 1–1 The NetBeans Start Screen



For more information on working with the NetBeans IDE, see "Using the Oracle Java ME SDK Software with NetBeans."

Installing and Starting the Eclipse IDE

If you do not already have Eclipse Indigo 3.7 installed on your system, you can download it here:

http://www.eclipse.org/downloads/packages/release/indigo/sr2

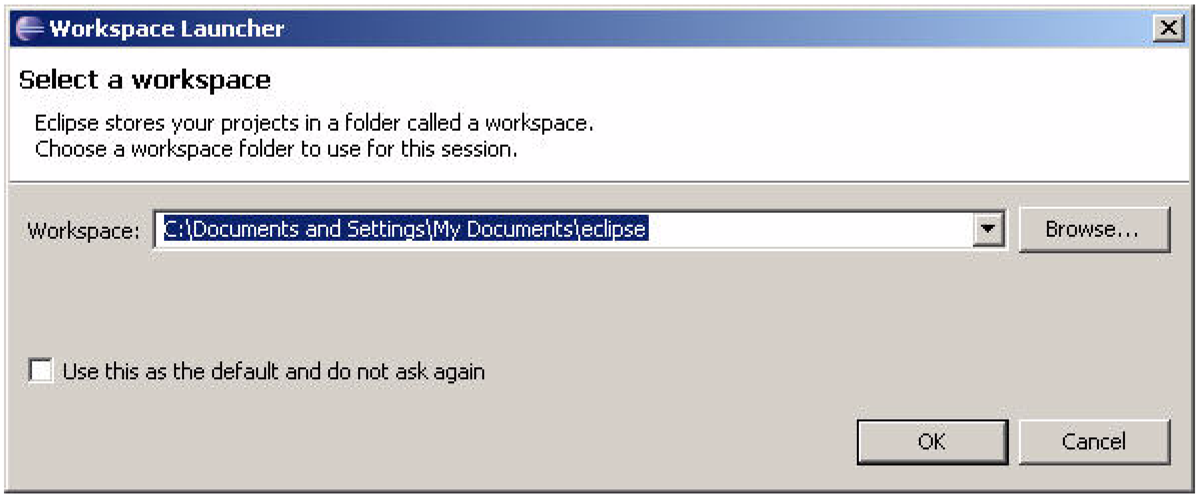
You can download Juno 4.2 here:

http://www.eclipse.org/downloads/packages/release/juno/sr1

Once you have downloaded the Eclipse Indigo 3.7 or Juno 4.2 installation module, do the following to install Eclipse on your system:

1. In the `C:\Program Files` directory, unzip the Eclipse distribution zip file.
2. In the `C:\Program Files\eclipse` directory, click `eclipse.exe` to launch Eclipse.
3. When the Select a Workspace dialog box appears, specify a directory location in which to store your Eclipse project files. Click OK.

Figure 1–2 The Eclipse Workspace Launcher



Select a workspace

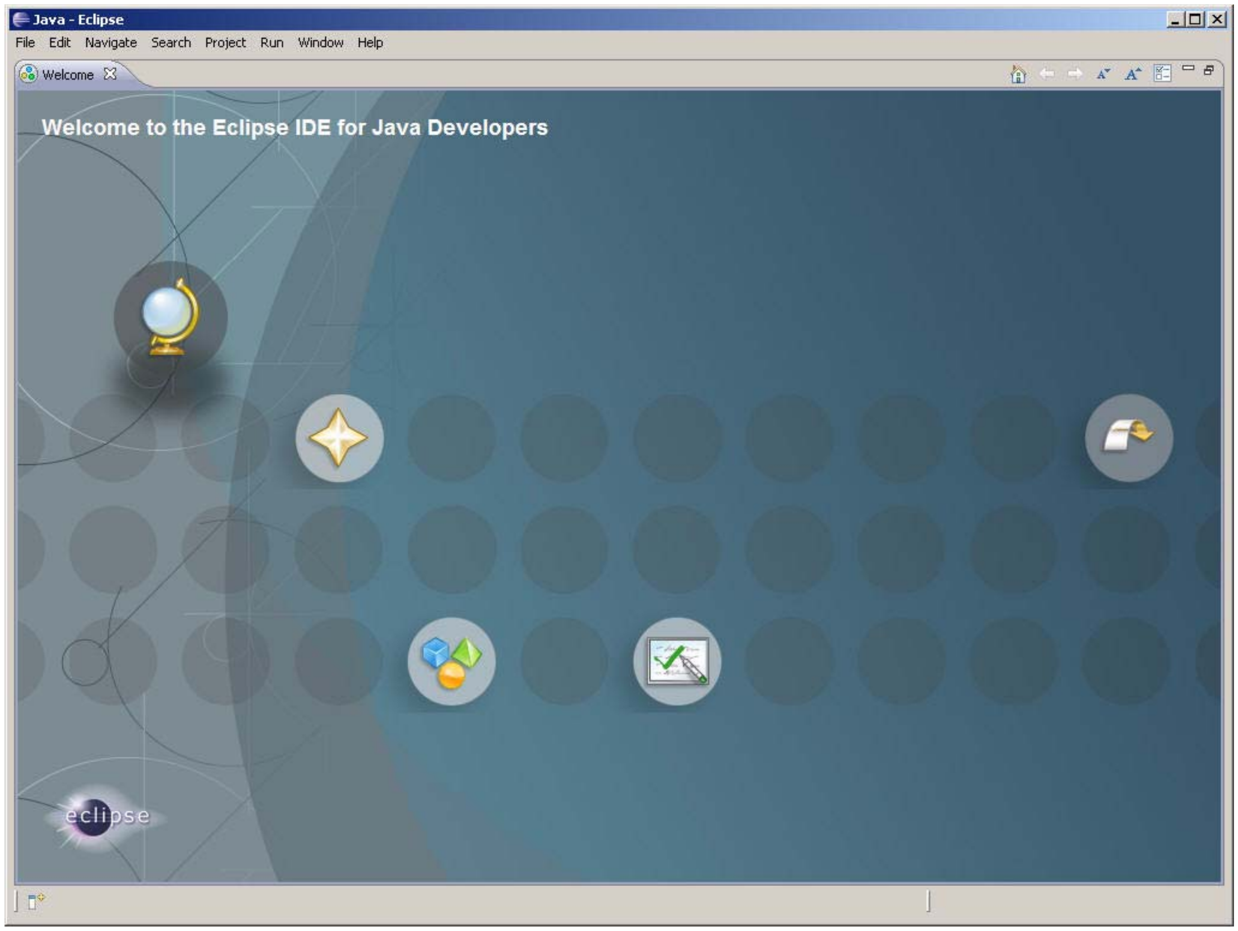
Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Documents and Settings\My Documents\eclipse

☐ Use this as the default and do not ask again

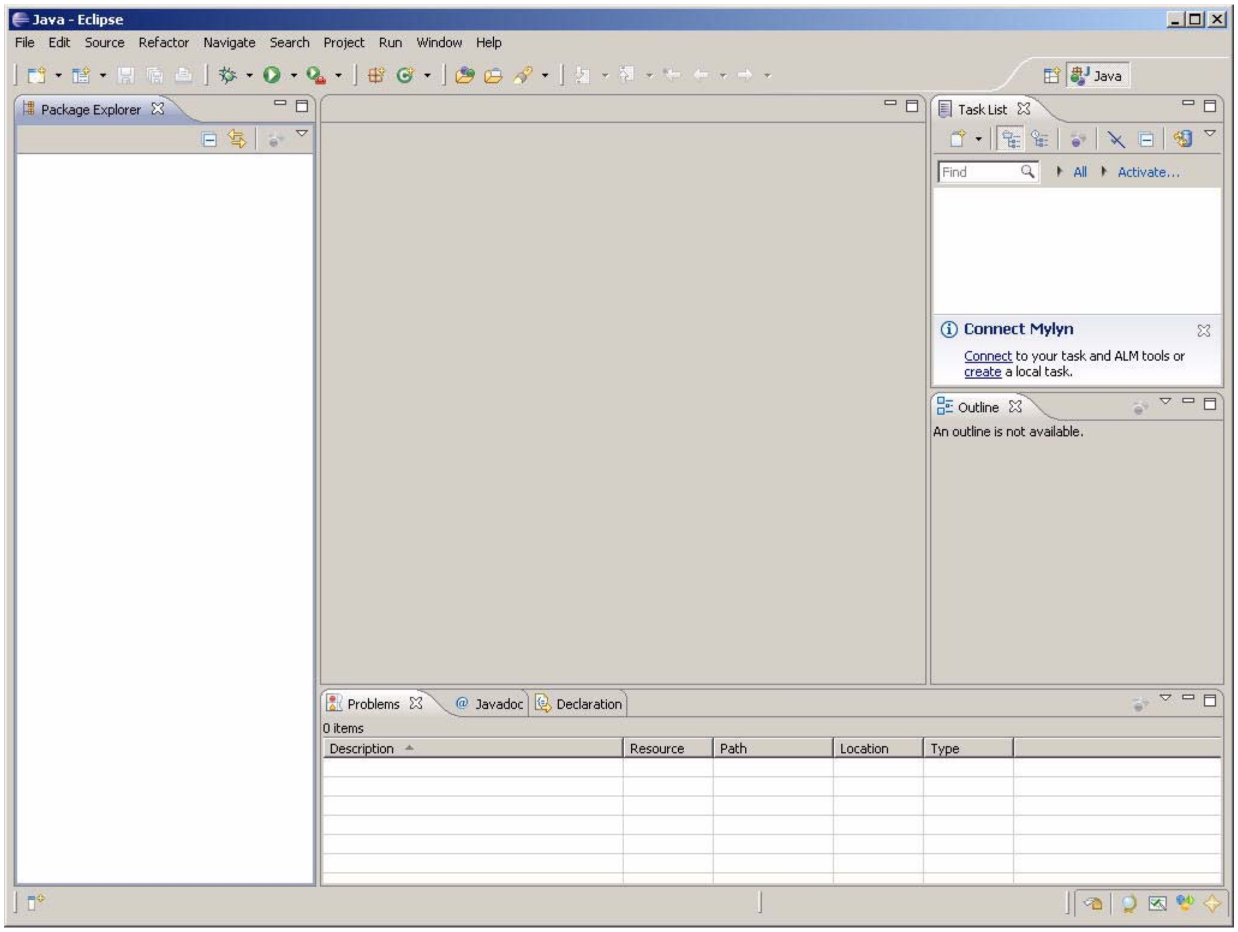
This opens the Eclipse main screen, as shown in Figure 1–3.

Figure 1–3 The Eclipse Start Screen


Java - Eclipse
File Edit Navigate Search Project Run Window Help
Welcome
Welcome to the Eclipse IDE for Java Developers
eclipse

4. Click the icon on the far right (it says "Go to the Workbench"). This displays an empty workbench screen, as shown in Figure 1–4.

Figure 1–4 The Eclipse Workbench Start Screen


Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Package Explorer
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
0 items
Description
Resource
Path
Location
Type

For more information on working with the Eclipse IDE, see "Using the Oracle Java ME SDK Software with Eclipse."

Verifying the UART Prefix

Lastly, verify that the `deviceaccess.uart.prefix` property in the `jwc_properties.ini` file contains a prefix that will make it easy to convert the `UARTConfig.portNumber` value to a platform-specific port name. This is used for COM port discovering.

Note: If you are planning to work *only* in emulation, you can skip this step.

In the Win32 environment, the property should be set to `COM` so that appending a port number maps correctly to the proper port name. Add the following line to the `jwc_properties.ini` file:

```
deviceaccess.uart.prefix=COM
```

The `jwc_properties.ini` file is found in the following location:

```
C:\Java_ME_platform_SDK\3.3\runtimes\impng\bin
```

2 Using the Oracle Java ME SDK Software with NetBeans

This chapter describes the NetBeans integrated development environment. NetBeans provides a rich, visual environment for developing embedded applications and numerous tools to improve the programming process.

Oracle Java ME SDK provides two plugins for working with NetBeans:

- Java ME SDK Tools plugin
- Java ME SDK Demos plugin

The Java ME SDK Demos plugin is optional, but recommended.

Downloading Oracle Java ME SDK Plugins

To download the Oracle Java ME SDK Plugins file for NetBeans (`oracle-jmesdk-3-3-rr-nb-plugins.zip`) go to the following location:

`http://www.oracle.com/technetwork/java/javame/javamobile/download/sdk`

Installing Oracle Java ME SDK Plugins

There are two ways to install the Oracle Java ME SDK Plugins:

- "Installing Plugins Using the Update Center"
- "Installing NetBeans Plugins Manually"

Note: Using the NetBeans Update Center to install the plugins is recommended. However, installing the plugins manually also works.

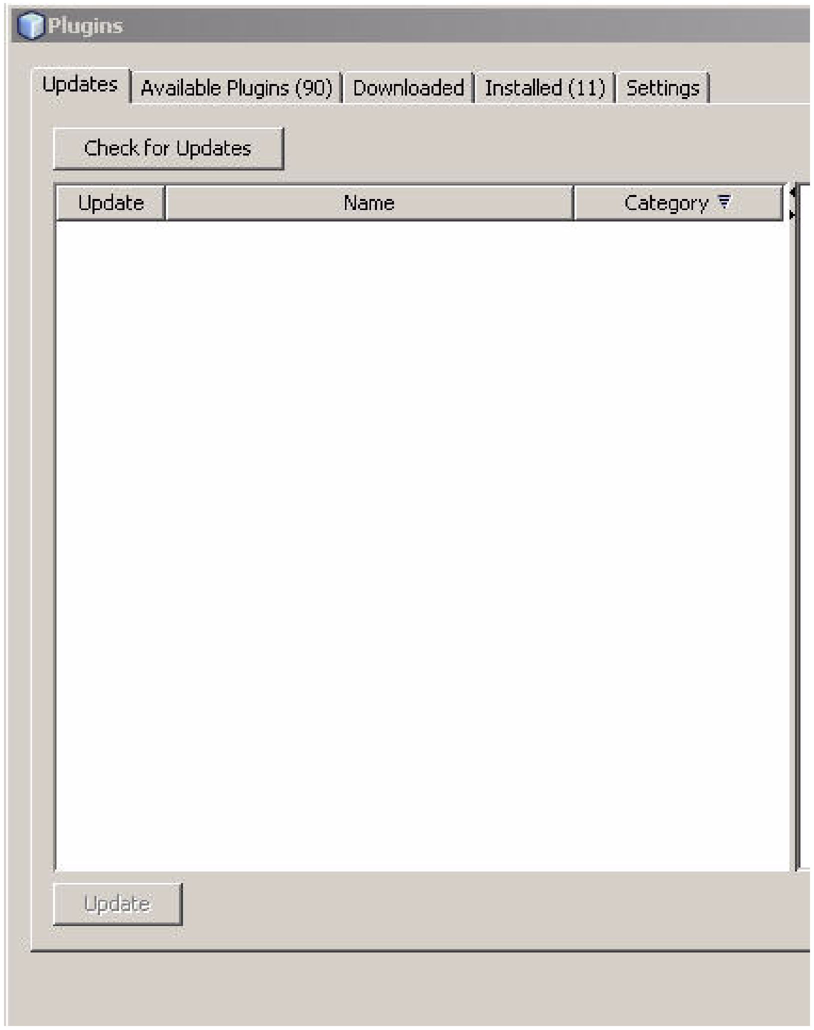
To get the plugins ready for installation, start NetBeans as described in Chapter 1

Installing Plugins Using the Update Center

To install the NetBeans Plugins using the NetBeans Update Center:

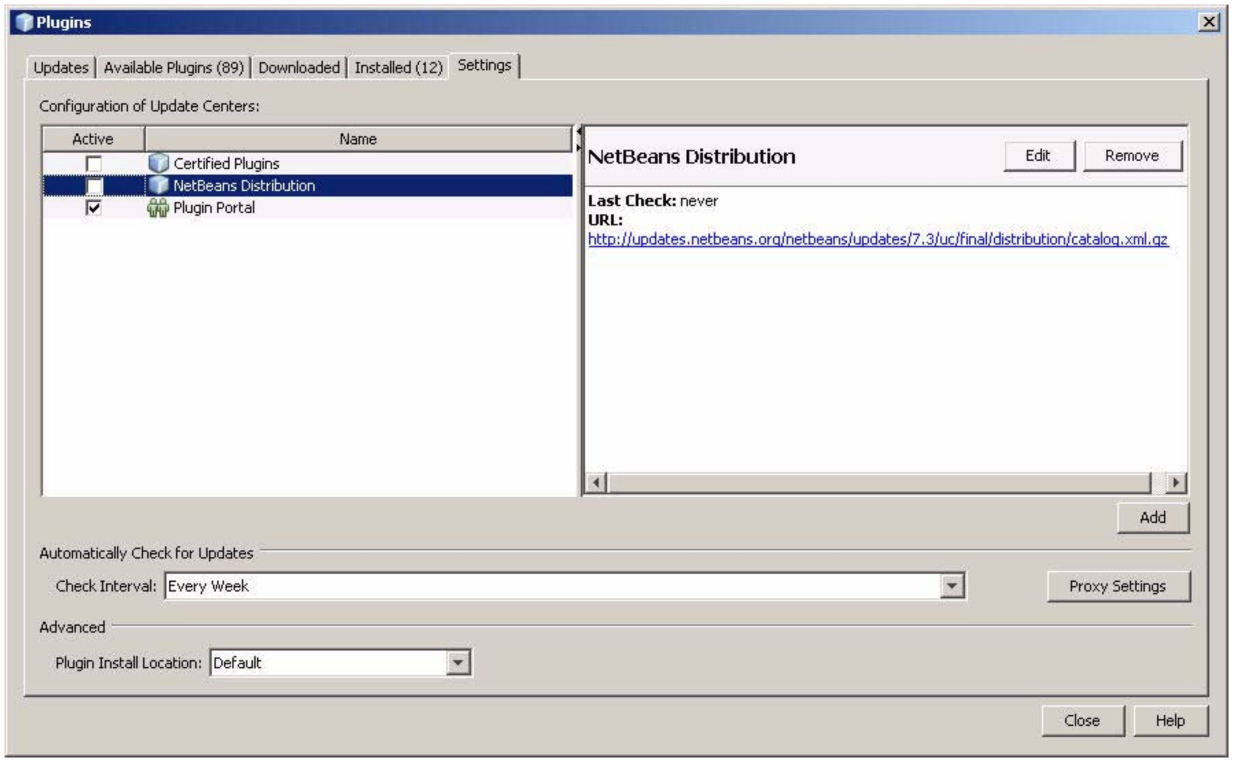
1. Select Tools > Plugins to open the NetBeans Plugins manager, as shown in Figure 2–1.

Figure 2–1 The NetBeans Plugin Manager



2. Click the Settings tab. This displays the available update centers, as shown in Figure 2–2.

Figure 2–2 Adding a Plugin Portal



Plugins

Updates | Available Plugins (89) | Downloaded | Installed (12) | Settings

Configuration of Update Centers:

Active	Name
☐	Certified Plugins
☐	NetBeans Distribution
☑	Plugin Portal

NetBeans Distribution

Last Check: never
URL:
http://updates.netbeans.org/netbeans/updates/7.3/uc/final/distribution/catalog.xml.gz

Automatically Check for Updates

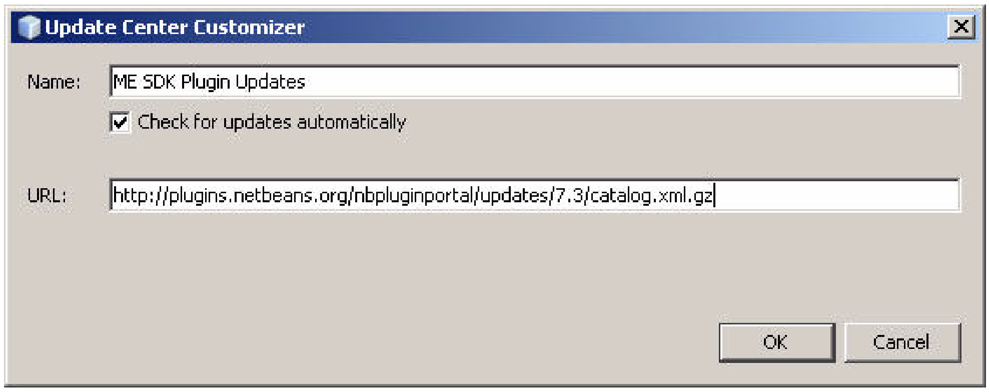
Check Interval: Every Week

Advanced

Plugin Install Location: Default

3. Click the Add button. In the new window, type a provider name and a URL for the location that contains the plugin center files, as shown in Figure 2–3.

Figure 2–3 Creating a New Update Center



Update Center Customizer

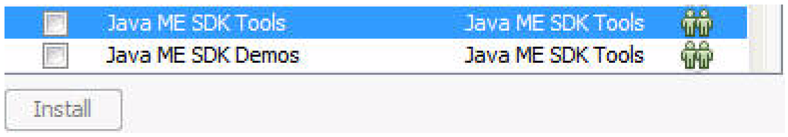
Name: ME SDK Plugin Updates

☑ Check for updates automatically

URL: http://plugins.netbeans.org/nbpluginportal/updates/7.3/catalog.xml.gz

4. When plugins are detected, they are displayed on the Available Plugins tab. If you do not see them, click the Check for Updates button.
   - Locate the Oracle Java ME SDK plugins.
   - In the Install column check the desired plugins, then click the Install button.

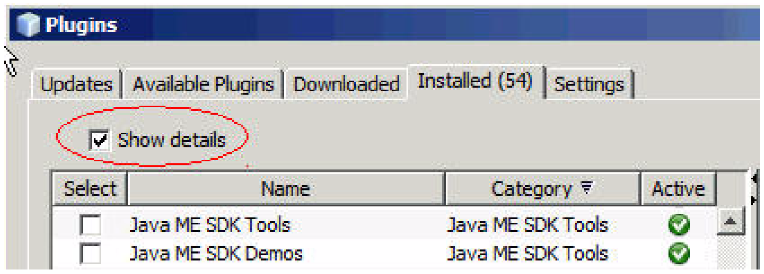
Figure 2–4 The Oracle Java ME SDK Plugins



☐	Java ME SDK Tools	Java ME SDK Tools
☐	Java ME SDK Demos	Java ME SDK Tools

5. Restart NetBeans.
6. In the Plugins Manager Settings tab, enable all available update centers.
7. In the Installed tab, check Show details (above the plugin list) and sort by category to easily find the Java ME SDK Tools plugins.
   - Make sure the plugins you installed are active (with a green check mark), as shown in Figure 2–5.
   - If the Oracle Java ME SDK plugins are not Active, check the Select boxes for the plugins and click Activate.

Figure 2–5 The Plugins Manager Installed Tab



Select	Name	Category	Active
☐	Java ME SDK Tools	Java ME SDK Tools	✓
☐	Java ME SDK Demos	Java ME SDK Tools	✓

The Oracle Java ME SDK is ready to use. For more information on how to verify your plugin installation, see "Verifying Your Installation."

Installing NetBeans Plugins Manually

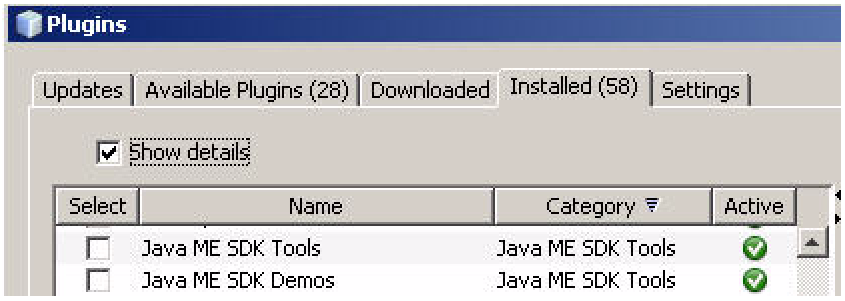
To install the NetBeans Plugins manually.

1. Extract the contents of the NetBeans Plugins file to a directory on your local machine. Make note of the location.
2. Open the NetBeans Plugins Manager. Select:

   Tools > Plugins
3. Uninstall previous plugins.
   - Go to the Installed tab and click Show details, as shown in Figure 2–6.
   - Check Java ME SDK Tools and Java ME SDK Demos.

   Note: If the previous plugins have already been uninstalled, go to Step 6.

Figure 2–6 NetBeans Plugins Manager Window

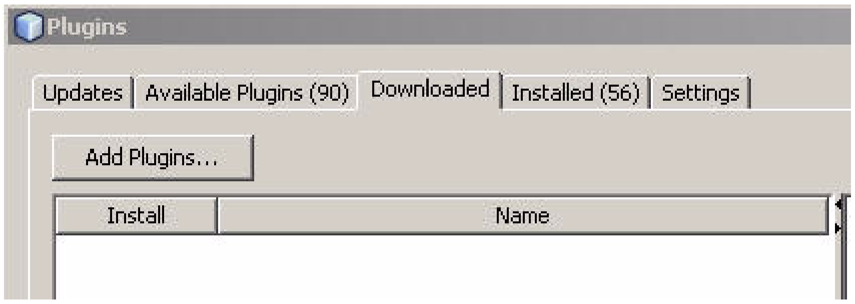


Select	Name	Category	Active
☐	Java ME SDK Tools	Java ME SDK Tools	✓
☐	Java ME SDK Demos	Java ME SDK Tools	✓

- Click Uninstall.
- Restart when requested.

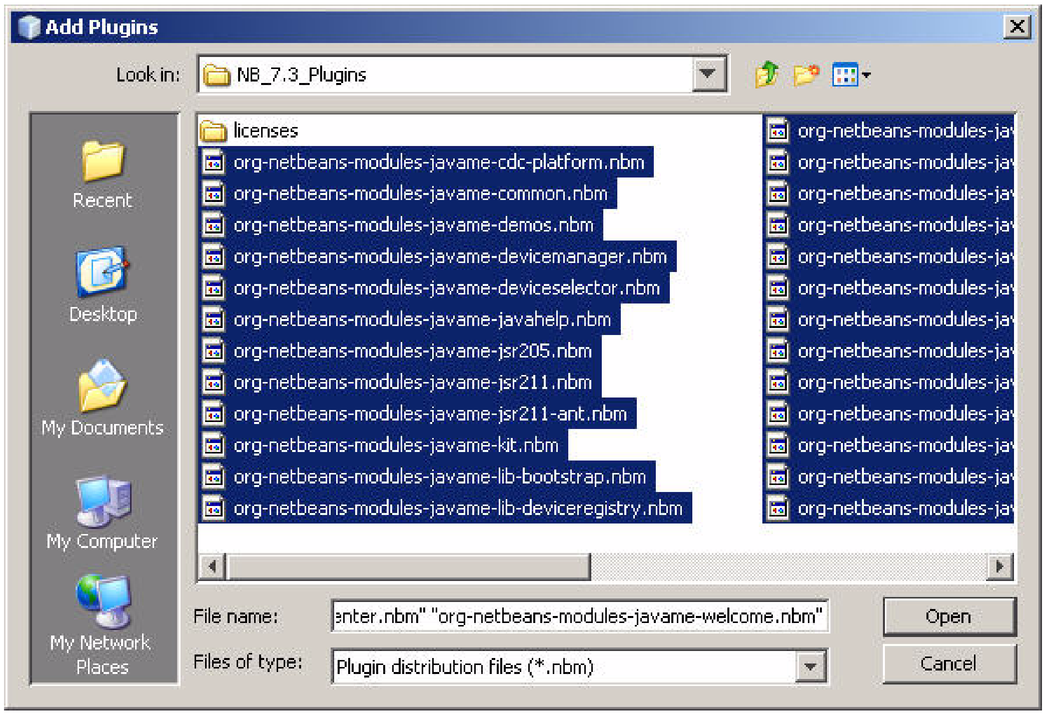
4. Re-open the Plugins Manager and click the Downloaded tab, as shown in Figure 2–7.
5. Click the Add Plugins button.

Figure 2–7 The Downloaded Tab and Add Plugins Button



6. In the file browser, go to the directory in which you have extracted the contents of the NetBeans Plugins.
7. Select all the `.nbm` files and click Open, as shown in Figure 2–8.

Figure 2–8 Adding Plugins

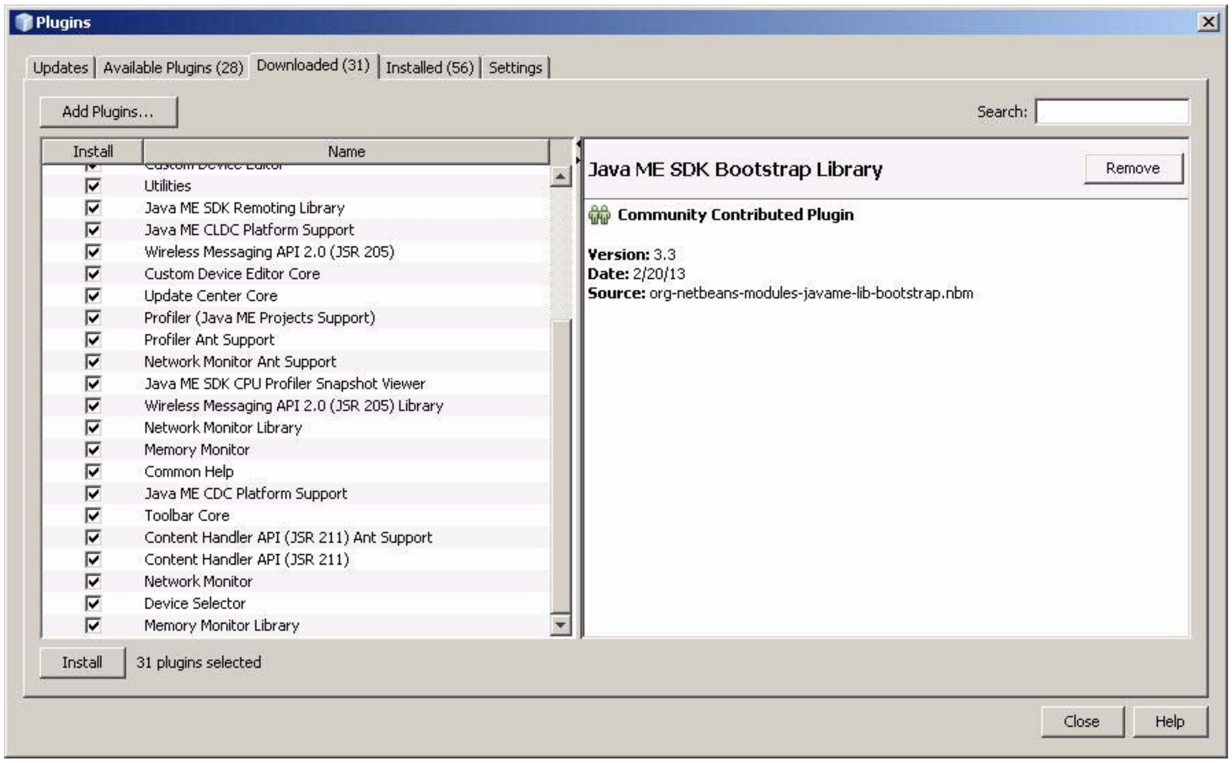


licenses
org-netbeans-modules-javame-cdc-platform.nbm
org-netbeans-modules-javame-common.nbm
org-netbeans-modules-javame-demos.nbm
org-netbeans-modules-javame-devicemanager.nbm
org-netbeans-modules-javame-deviceselector.nbm
org-netbeans-modules-javame-javahelp.nbm
org-netbeans-modules-javame-jsr205.nbm
org-netbeans-modules-javame-jsr211.nbm
org-netbeans-modules-javame-jsr211-ant.nbm
org-netbeans-modules-javame-kit.nbm
org-netbeans-modules-javame-lib-bootstrap.nbm
org-netbeans-modules-javame-lib-deviceregistry.nbm

File name: enter.nbm" "org-netbeans-modules-javame-welcome.nbm"
Files of type: Plugin distribution files (*.nbm)

8. Go to the Downloaded tab and select all downloaded plugins, as shown in Figure 2–9, and click Install.

Figure 2–9 Selecting Downloaded Plugins



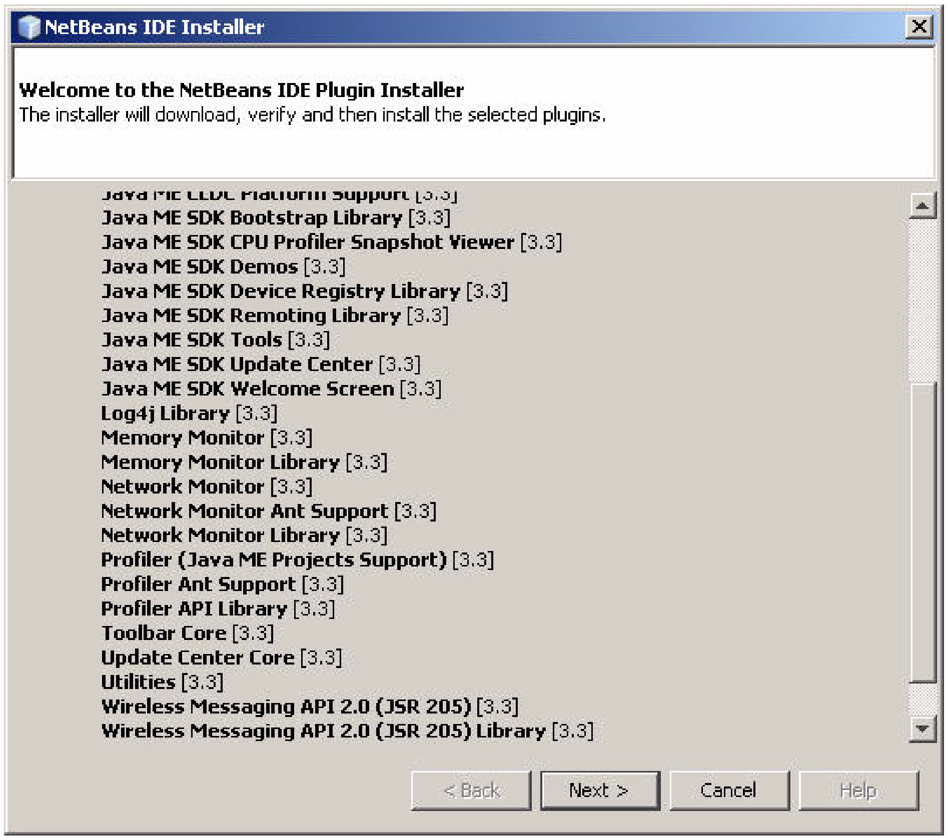
Install	Name
✓	Utilities
✓	Java ME SDK Remoting Library
✓	Java ME CLDC Platform Support
✓	Wireless Messaging API 2.0 (JSR 205)
✓	Custom Device Editor Core
✓	Update Center Core
✓	Profiler (Java ME Projects Support)
✓	Profiler Ant Support
✓	Network Monitor Ant Support
✓	Java ME SDK CPU Profiler Snapshot Viewer
✓	Wireless Messaging API 2.0 (JSR 205) Library
✓	Network Monitor Library
✓	Memory Monitor
✓	Common Help
✓	Java ME CDC Platform Support
✓	Toolbar Core
✓	Content Handler API (JSR 211) Ant Support
✓	Content Handler API (JSR 211)
✓	Network Monitor
✓	Device Selector
✓	Memory Monitor Library

31 plugins selected

Java ME SDK Bootstrap Library
Community Contributed Plugin
Version: 3.3
Date: 2/20/13
Source: org-netbeans-modules-javame-lib-bootstrap.nbm

9. When the NetBeans IDE Installer screen is displayed, as shown in Figure 2–10, click Next.
   - Accept the license terms and click Install.
   - If additional Validation screens appear, click Continue.

Figure 2–10 The NetBeans IDE Installer



Welcome to the NetBeans IDE Plugin Installer
The installer will download, verify and then install the selected plugins.

Java ME CLDC Platform Support [3.3]
Java ME SDK Bootstrap Library [3.3]
Java ME SDK CPU Profiler Snapshot Viewer [3.3]
Java ME SDK Demos [3.3]
Java ME SDK Device Registry Library [3.3]
Java ME SDK Remoting Library [3.3]
Java ME SDK Tools [3.3]
Java ME SDK Update Center [3.3]
Java ME SDK Welcome Screen [3.3]
Log4j Library [3.3]
Memory Monitor [3.3]
Memory Monitor Library [3.3]
Network Monitor [3.3]
Network Monitor Ant Support [3.3]
Network Monitor Library [3.3]
Profiler (Java ME Projects Support) [3.3]
Profiler Ant Support [3.3]
Profiler API Library [3.3]
Toolbar Core [3.3]
Update Center Core [3.3]
Utilities [3.3]
Wireless Messaging API 2.0 (JSR 205) [3.3]
Wireless Messaging API 2.0 (JSR 205) Library [3.3]

10. Click Finish to restart NetBeans.
11. Select Tools > Plugins to display the Plugins screen.
12. In the Installed tab, check Show details and click Category to sort the plugins.
    - Find the Java ME SDK Tools and Java ME SDK Demos plugins in the list.
    - Make sure the plugins you installed are Active (you should see a green check mark), as shown in Figure 2–6.
    - If the Oracle Java ME SDK plugins are not Active, check the Select boxes for the plugins and click Activate.
13. When your Oracle Java ME SDK plugins are Active, click Close.

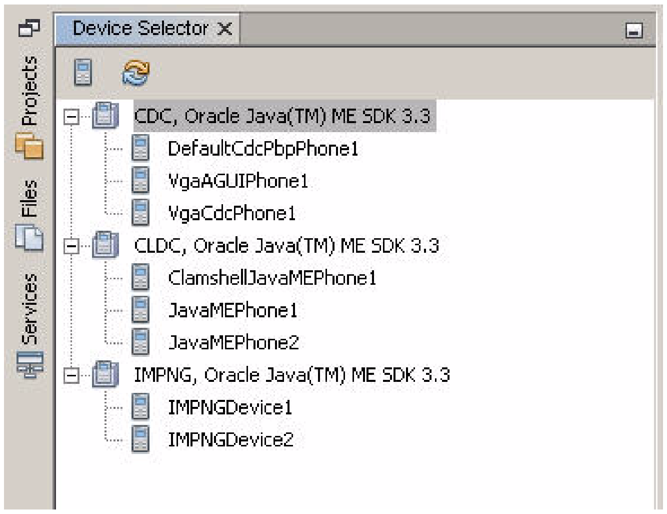
    The Oracle Java ME SDK is ready to use. For more information on how to verify your plugin installation, see "Verifying Your Installation."

Verifying Your Installation

Once you have installed the Oracle Java ME SDK Plugins into NetBeans (using the Update Center or by manual installation), the Oracle Java ME SDK platform is installed into the NetBeans IDE. To verify a successful installation, do the following:

1. To verify the Active Device Manager, select Tools > Java ME > Active Device Manager, and select the latest version (Oracle Java ME SDK 3.3).
2. To view available Oracle Java ME SDK 3.3 devices, select Tools > Java ME > Device Selector. The ME SDK devices are listed, as shown in Figure 2–11.

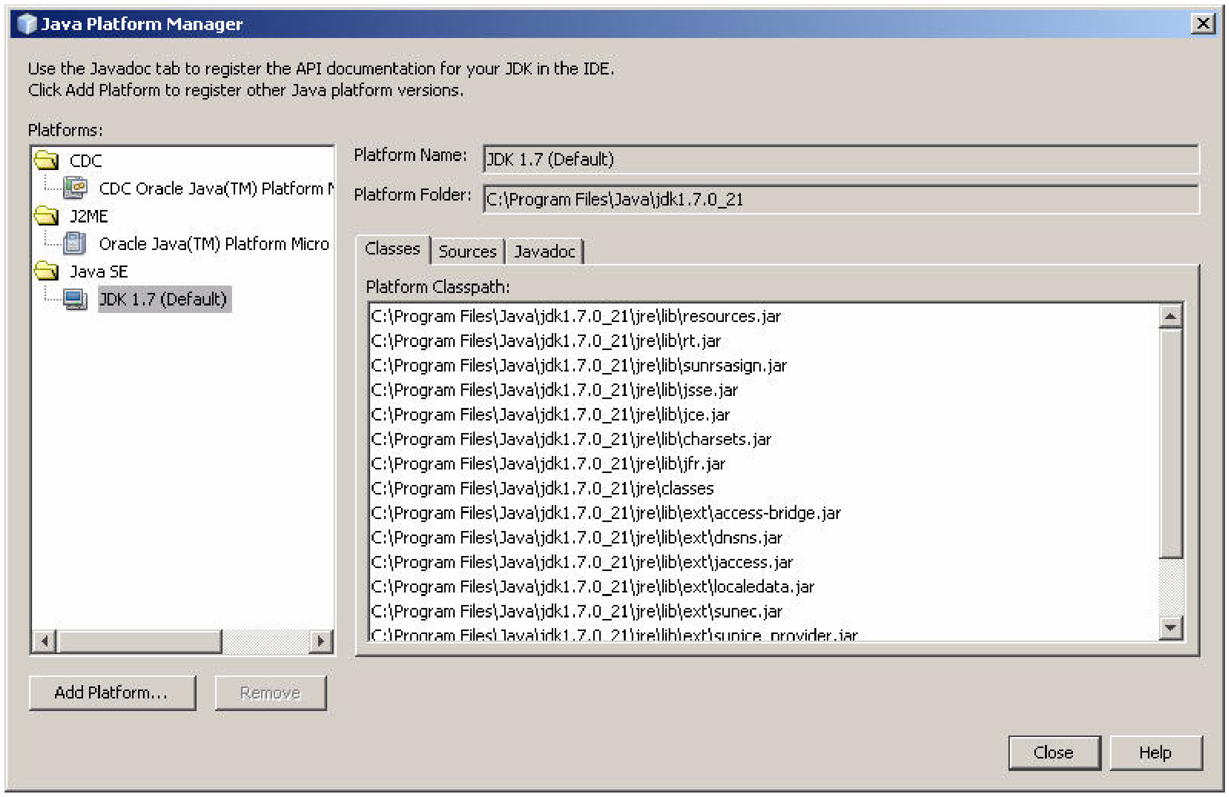
Figure 2–11 The Device Selector



```
CDC, Oracle Java(TM) ME SDK 3.3
    DefaultCdcPbpPhone1
    VgaAGUIPhone1
    VgaCdcPhone1
CLDC, Oracle Java(TM) ME SDK 3.3
    ClamshellJavaMEPhone1
    JavaMEPhone1
    JavaMEPhone2
IMPNG, Oracle Java(TM) ME SDK 3.3
    IMPNGDevice1
    IMPNGDevice2
```

3. To display the new Oracle Java ME SDK platform, choose Tools > Java Platforms. This displays the Java Platform Manager with the new platform, as shown in Figure 2–12.

Figure 2–12 The Java Platform Manager


Java Platform Manager
Use the Javadoc tab to register the API documentation for your JDK in the IDE.
Click Add Platform to register other Java platform versions.
Platforms:
CDC
CDC Oracle Java(TM) Platform M
J2ME
Oracle Java(TM) Platform Micro
Java SE
JDK 1.7 (Default)
Platform Name:
Platform Folder:
Classes
Sources
Javadoc
Platform Classpath:
Add Platform...
Remove
Close
Help

Platform Name: JDK 1.7 (Default)
Platform Folder: C:\Program Files\Java\jdk1.7.0_21

```
C:\Program Files\Java\jdk1.7.0_21\jre\lib\resources.jar
C:\Program Files\Java\jdk1.7.0_21\jre\lib\rt.jar
C:\Program Files\Java\jdk1.7.0_21\jre\lib\sunrsasign.jar
C:\Program Files\Java\jdk1.7.0_21\jre\lib\jsse.jar
C:\Program Files\Java\jdk1.7.0_21\jre\lib\jce.jar
C:\Program Files\Java\jdk1.7.0_21\jre\lib\charsets.jar
C:\Program Files\Java\jdk1.7.0_21\jre\lib\jfr.jar
C:\Program Files\Java\jdk1.7.0_21\jre\classes
C:\Program Files\Java\jdk1.7.0_21\jre\lib\ext\access-bridge.jar
C:\Program Files\Java\jdk1.7.0_21\jre\lib\ext\dnsns.jar
C:\Program Files\Java\jdk1.7.0_21\jre\lib\ext\jaccess.jar
C:\Program Files\Java\jdk1.7.0_21\jre\lib\ext\localedata.jar
C:\Program Files\Java\jdk1.7.0_21\jre\lib\ext\sunec.jar
C:\Program Files\Java\jdk1.7.0_21\jre\lib\ext\sunjce_provider.jar
```

The Oracle ME SDK Platform is now ready for you to work with, to create a new project and develop code, as shown in the following sections.

Creating a Sample IMlet File

In this section, you create a sample IMlet file, `IMletDemo.java`, from the code provided in Example 2–1. This IMlet file is used in the next section, "Creating a New Project".

1. Copy the code shown in Example 2–1 into a text file. Use Notepad rather than WordPad, to avoid any unneeded extra characters.
2. Name the file `IMletDemo.java` and Save. Set the file aside for now.

Example 2–1 Code for the Sample IMletDemo.java Project in NetBeans

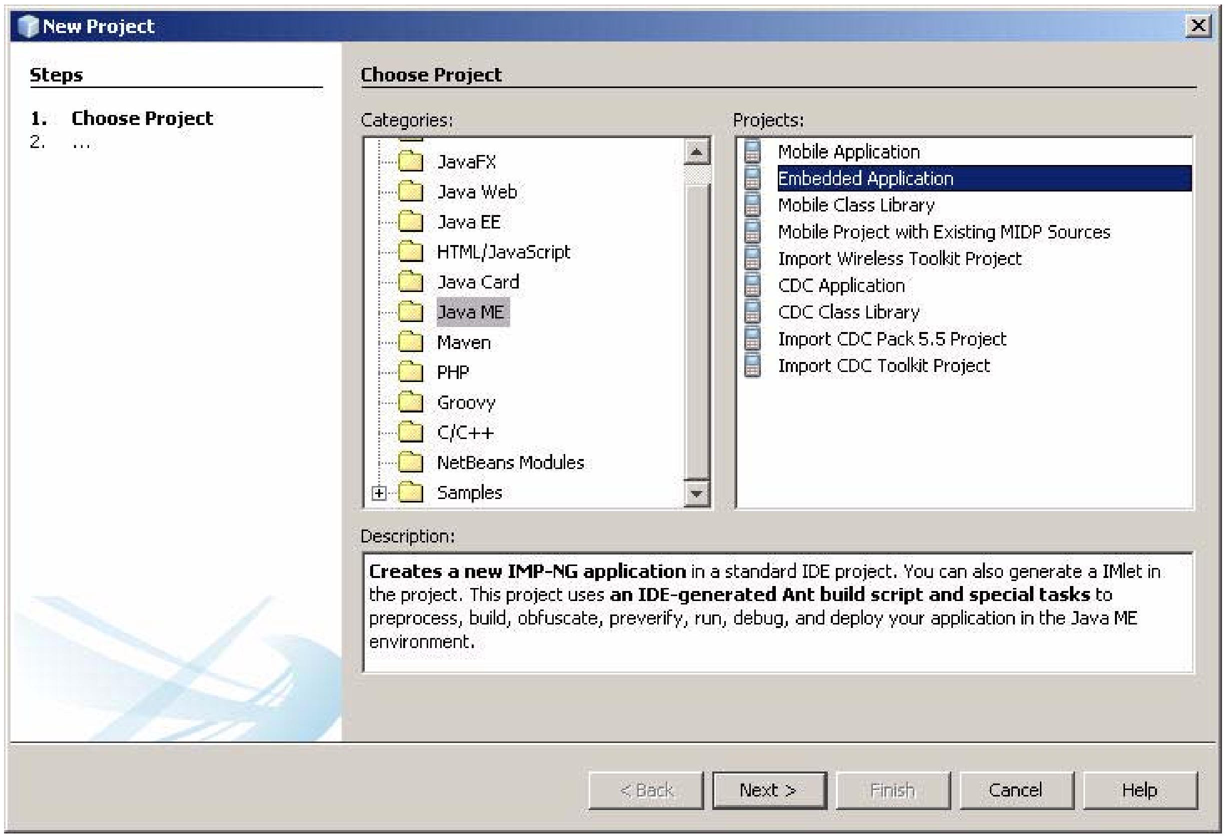
```
package imletdemo;
import javax.microedition.midlet.MIDlet;

public class IMletDemo extends MIDlet {

     boolean bFirst = false;
     boolean loopFlag = true;


     public void startApp() {

         if (bFirst = false) {

            try {
                // Perform startup operations
                } catch (Exception ex) {
                    ex.printStackTrace();
                    return;
                }

                bFirst = true;
            } else {
                System.out.println("IMlet Demo is already started...");
            }

            // Start program here

        }

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

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

           // Close all resources that have been opened

        }


}
```

Creating a New Project

This section walks you through creating a new embedded project using the Oracle Java ME SDK and NetBeans platforms.

1. Choose File > New Project. The New Project dialog box appears.
2. Choose Java ME from the Categories list and Embedded Application from the Projects list, as shown in Figure 2–13.

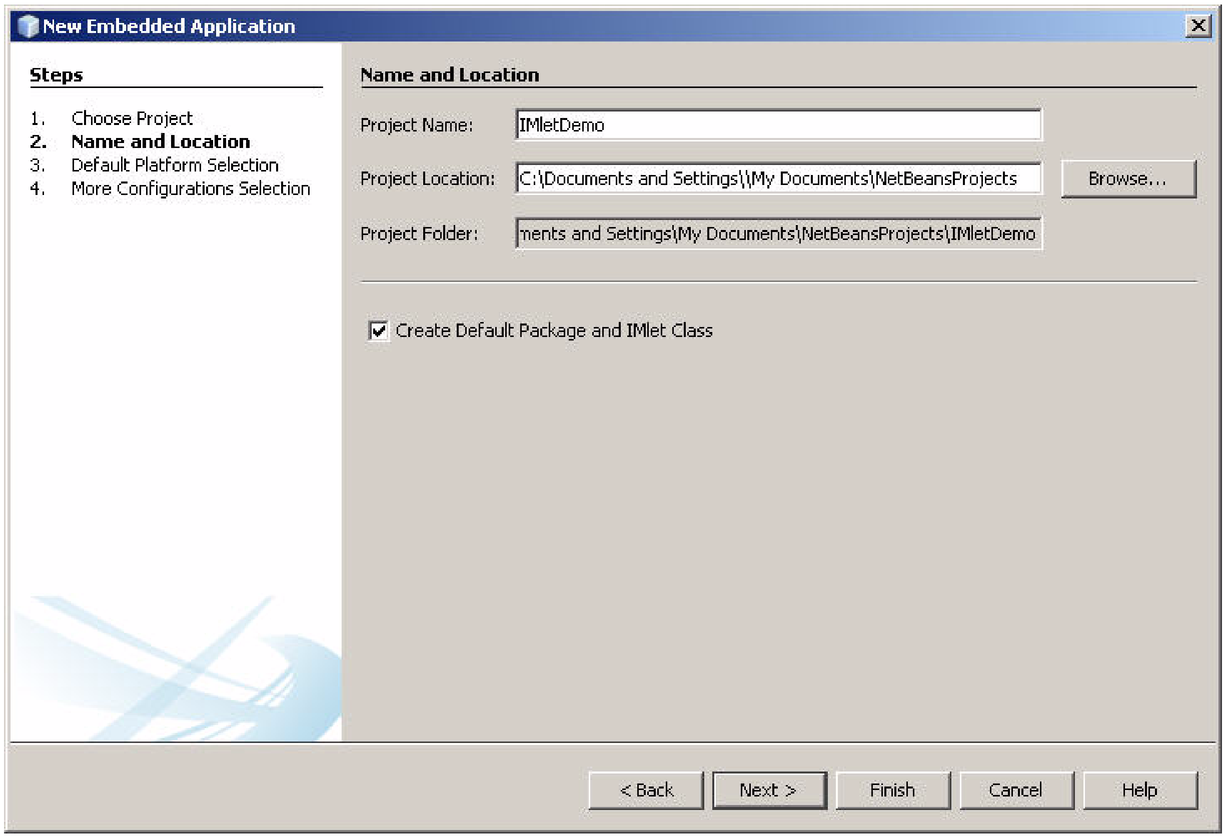
Figure 2–13 The New Project Screen



Creates a new IMP-NG application in a standard IDE project. You can also generate a IMlet in the project. This project uses **an IDE-generated Ant build script and special tasks** to preprocess, build, obfuscate, preverify, run, debug, and deploy your application in the Java ME environment.

3. Press the Next button. This shows the The New Embedded Application dialog box, with `EmbeddedApplication1` in the Name and Location panel.
4. Enter `IMletDemo` as the Project Name.
5. Accept the default Project Location or browse to select it. Ensure that the Create Default Package and IMlet Class option is checked, as shown in Figure 2–14. Click Next.

Figure 2–14 Creating a New Project



New Embedded Application

Steps

1. Choose Project
2. **Name and Location**
3. Default Platform Selection
4. More Configurations Selection

Name and Location

Project Name: IMletDemo

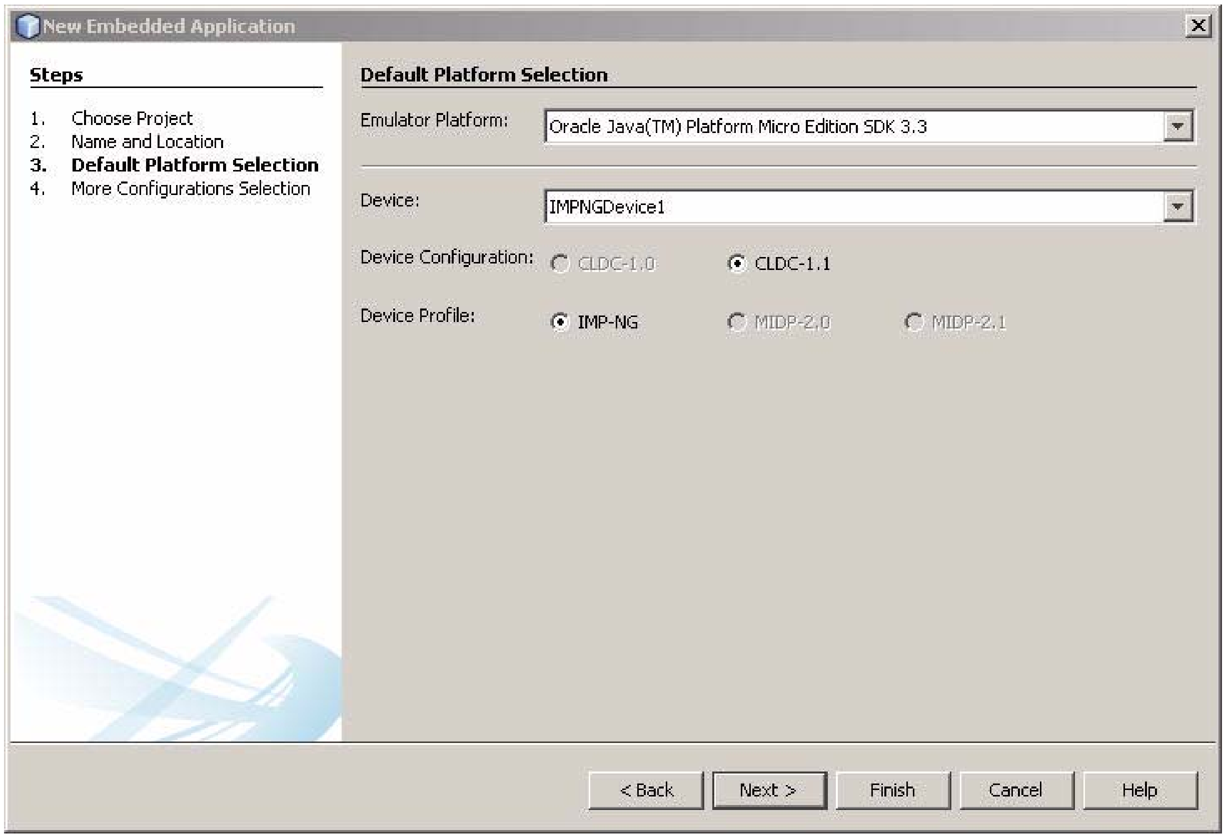
Project Location: C:\Documents and Settings\\My Documents\NetBeansProjects

Project Folder: ments and Settings\My Documents\NetBeansProjects\IMletDemo

[✓] Create Default Package and IMlet Class

6. The Default Platform Selection is displayed. In the Emulator Platform box, choose "Oracle Java(TM) Platform Micro Edition 3.3." Choose "IMPNGDevice1" for the Device, "CLDC 1.1" for the Device Configuration, and "IMP-NG" for the Device Profile, as shown in Figure 2–15. Press Finish.

Figure 2–15 Setting the New Default Platform



New Embedded Application

Steps

1. Choose Project
2. Name and Location
3. **Default Platform Selection**
4. More Configurations Selection

Default Platform Selection

Emulator Platform: Oracle Java(TM) Platform Micro Edition SDK 3.3

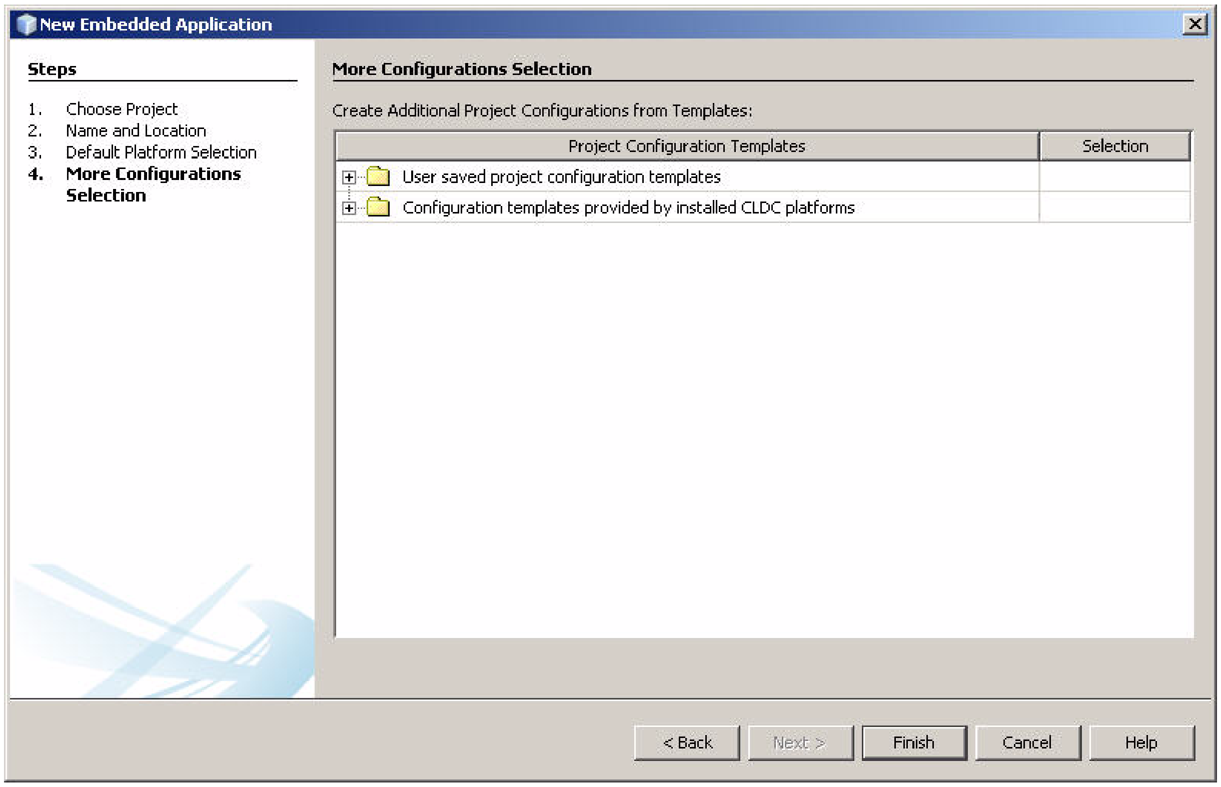
Device: IMPNGDevice1

Device Configuration: CLDC-1.0 [✓ CLDC-1.1]

Device Profile: [✓ IMP-NG] MIDP-2.0 MIDP-2.1

7. This More Configurations Selection box is displayed, as shown in Figure 2–16. Accept the defaults or add additional configurations. Press Finish.

Figure 2–16 Adding More Configurations

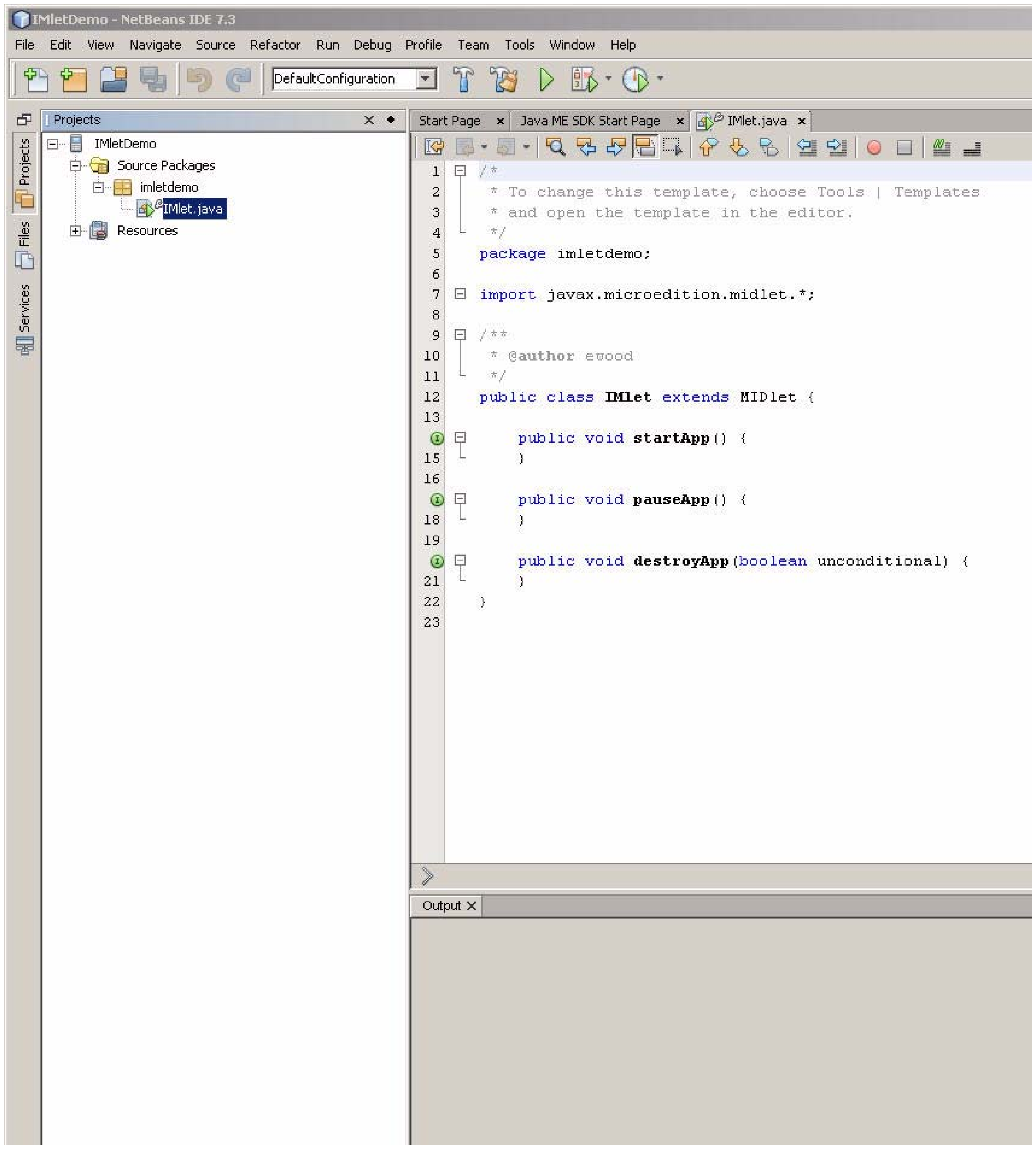


Create Additional Project Configurations from Templates:

Project Configuration Templates	Selection
User saved project configuration templates	
Configuration templates provided by installed CLDC platforms	

8. The NetBeans New Project screen is displayed for the project you have just created, containing default minimal sample code, as shown in Figure 2–17

Figure 2–17 The New IMletDemo Project

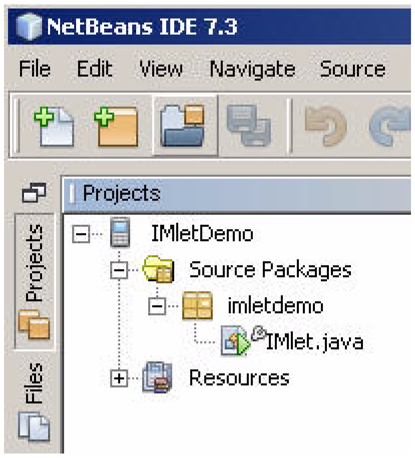


```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package imletdemo;

import javax.microedition.midlet.*;

/**
 * @author ewood
 */
public class IMlet extends MIDlet {

    public void startApp() {
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```

Including Your Sample IMlet Code

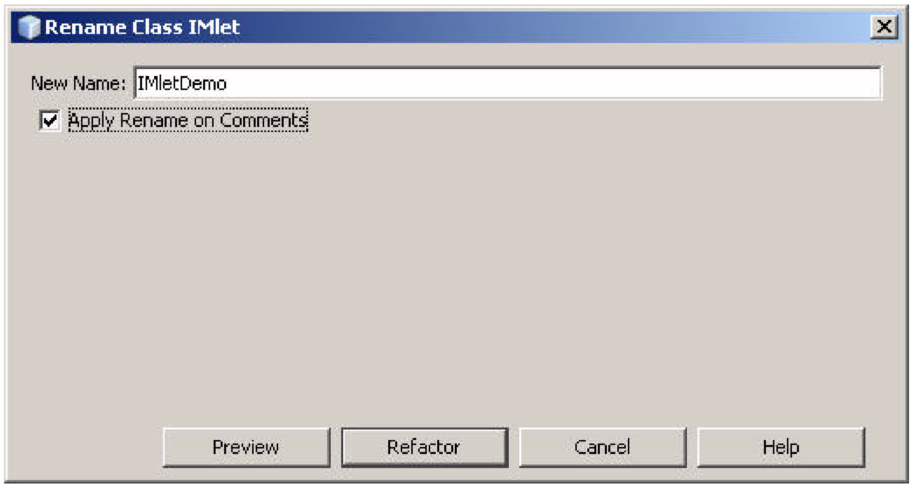
Now you can update the generic project with the sample code you created earlier in "Creating a Sample IMlet File." The NetBeans Projects window looks like Figure 2–18.

Figure 2–18   The IMletDemo Project Tree



1. Right click to select `IMlet.java` in the Projects window.
2. Select Refactor --> Rename to display the Rename Class IMlet dialog box.
3. In the Rename Class IMlet dialog box, enter `IMletDemo`. Ensure that Apply Rename on Comments is checked, as shown in Figure 2–19. Press Refactor.

Figure 2–19   The Rename Class IMlet Dialog Box

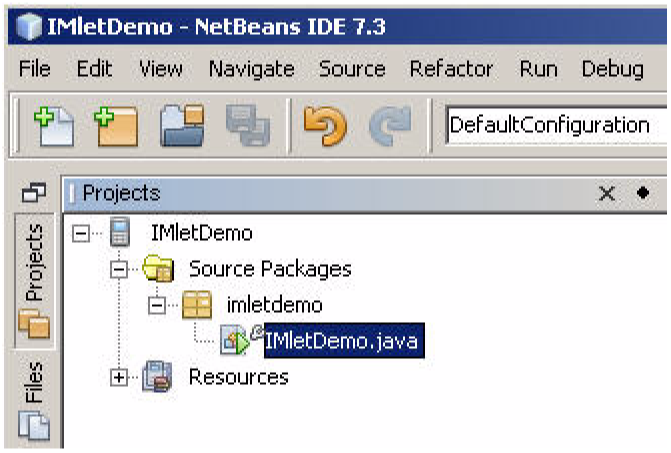


New Name: IMletDemo

[✓] Apply Rename on Comments

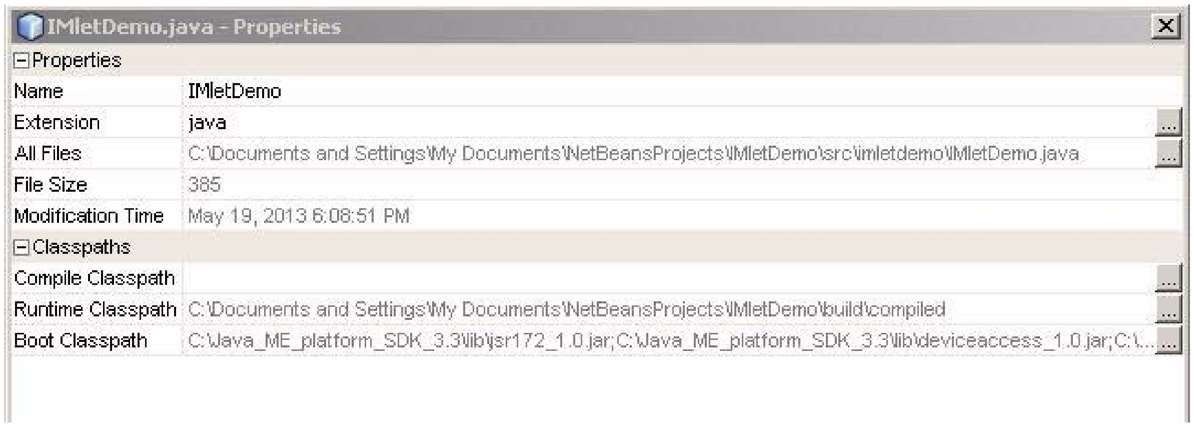
4. This changes the name in the Projects window to `IMletDemo.java`, as shown in Figure 2–20. Right click `IMletDemo.java` and select Properties.

Figure 2–20 The IMletDemo Projects Window



5. This displays the `IMletDemo.java` - Properties window. To see the path to the location of the newly-refactored `IMletDemo.java` file, look in the All Files line, as shown in Figure 2–21. Make a note of this path.

Figure 2–21 The IMletDemo Properties Window



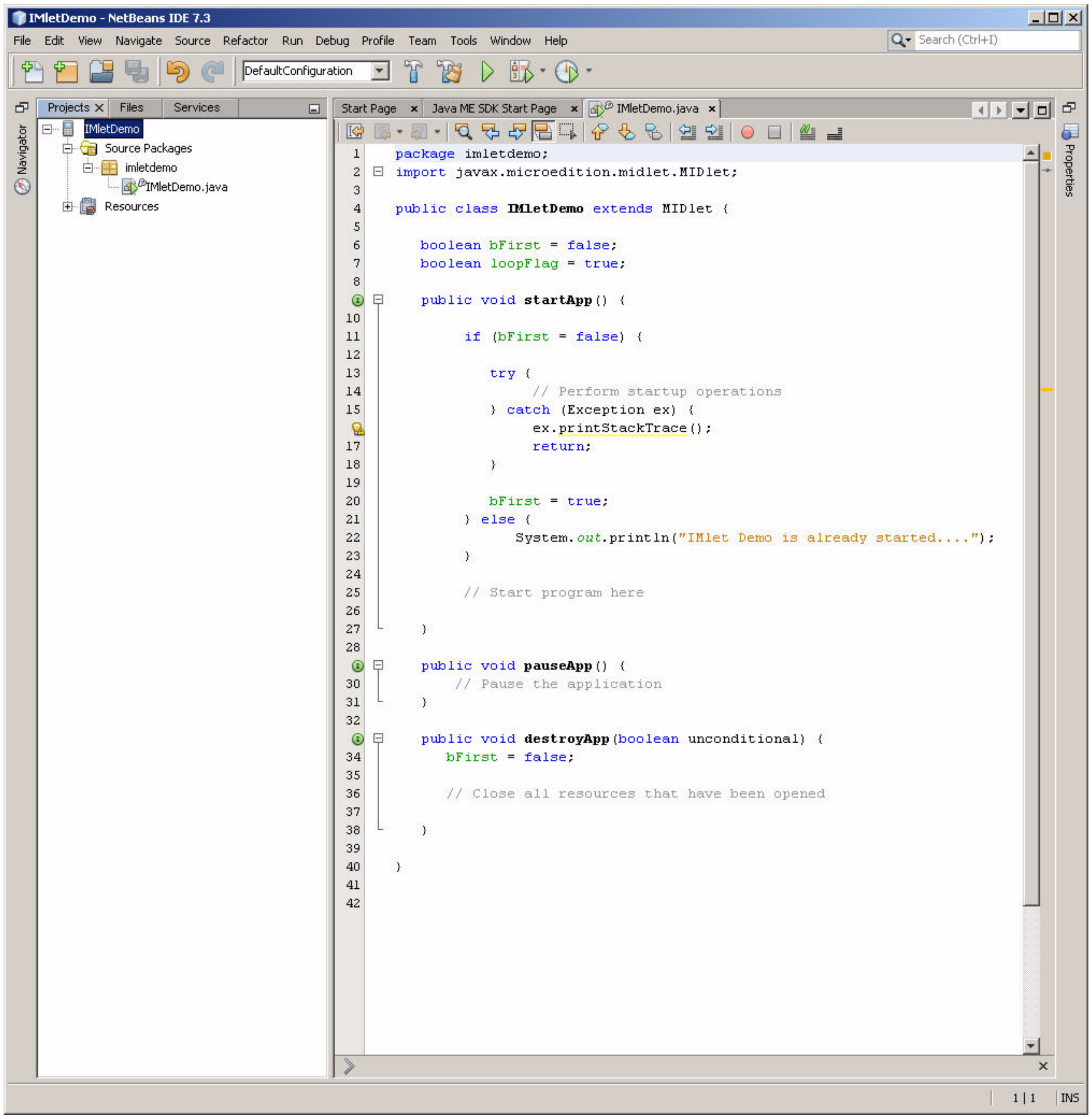
IMletDemo.java - Properties

Properties	
Name	IMletDemo
Extension	java
All Files	C:\Documents and Settings\My Documents\NetBeansProjects\IMletDemo\src\imletdemo\IMletDemo.java
File Size	385
Modification Time	May 19, 2013 6:08:51 PM
Classpaths	
Compile Classpath	
Runtime Classpath	C:\Documents and Settings\My Documents\NetBeansProjects\IMletDemo\build\compiled
Boot Classpath	C:\Java_ME_platform_SDK_3.3\lib\jsr172_1.0.jar;C:\Java_ME_platform_SDK_3.3\lib\deviceaccess_1.0.jar;C:\...

6. Take the `IMletDemo.java` file you created in "Creating a Sample IMlet File" and use it to overwrite the `IMletDemo.java` file in the path location you have just noted. When it asks you to confirm the File Replace, click Yes.

7. This changes the content of the NetBeans Project window to the content contained in the newly-added `IMletDemo.java` file, as shown in Figure 2–22. (If it does not double-click `IMletDemo.java` in the Projects window.

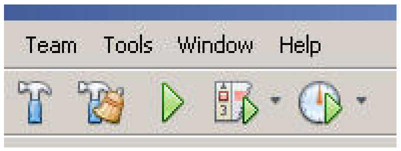
Figure 2–22 The NetBeans Project Window with the IMletDemo Project



```
package imletdemo;
import javax.microedition.midlet.MIDlet;

public class IMletDemo extends MIDlet {

    boolean bFirst = false;
    boolean loopFlag = true;

    public void startApp() {

        if (bFirst = false) {

            try {
                // Perform startup operations
            } catch (Exception ex) {
                ex.printStackTrace();
                return;
            }

            bFirst = true;
        } else {
            System.out.println("IMlet Demo is already started....");
        }

        // Start program here

    }

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

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

        // Close all resources that have been opened

    }

}
```

8. Clean and build the IMletDemo project by clicking on the hammer-and-broom icon in the NetBeans toolbar, as shown in Figure 2–23, or by selecting Run > Clean and Build Project (IMletDemo).

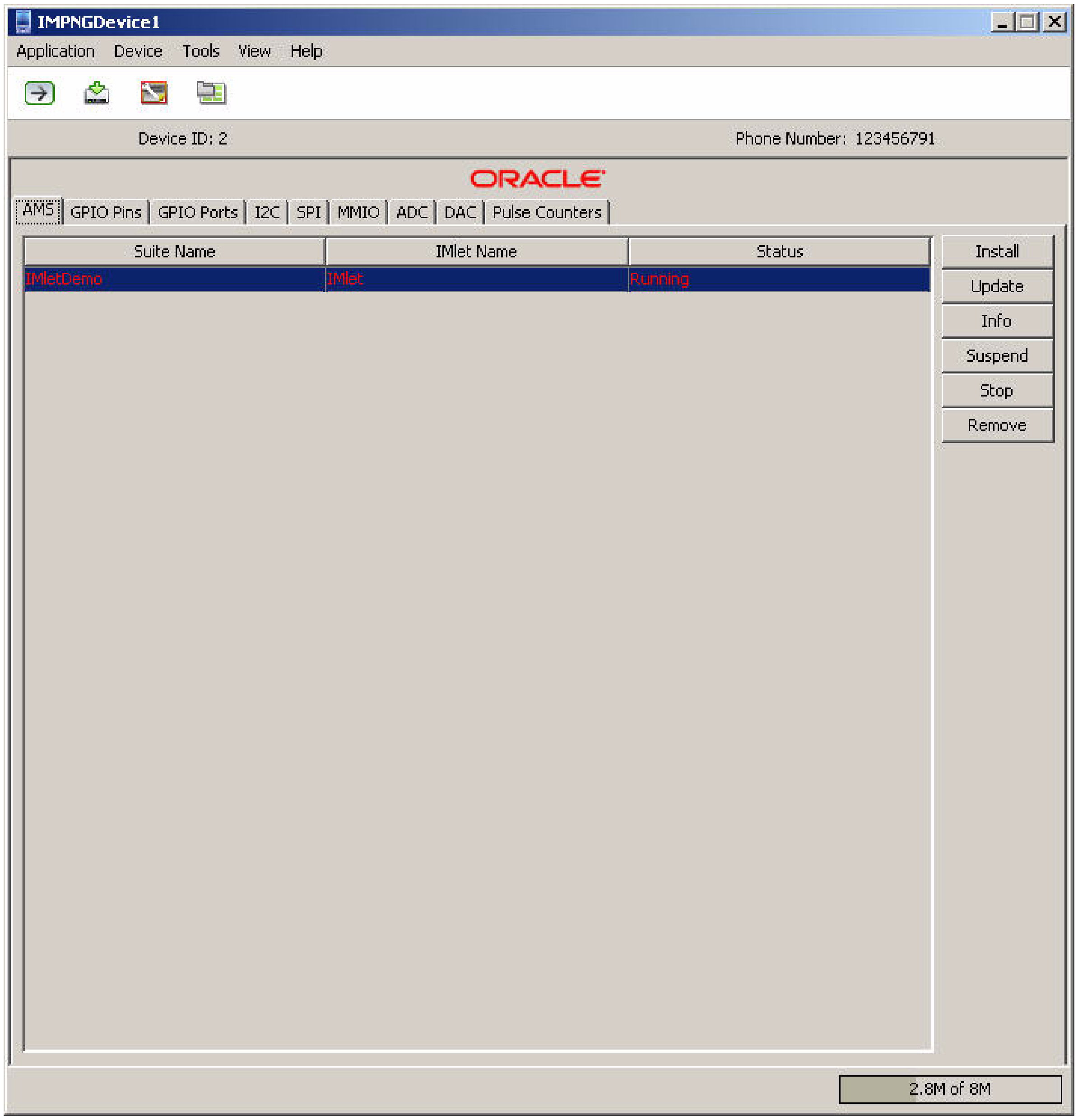
Figure 2–23 The NetBeans Toolbar and Menus



9. Run the newly cleaned and built IMletDemo project by selecting the green right-arrow icon in the NetBeans toolbar or by selecting Run > Run Project (IMletDemo).

   When the Run is successful, the IMPNGDevice1 emulator starts with the IMletDemo Suite running, as shown in Figure 2–24.

Figure 2–24 The NetBeans Emulator Window Running the IMletDemo Project



Suite Name	IMlet Name	Status
IMletDemo	IMlet	Running

For more information about the emulator, see Chapter 3, "Using the Emulator".

3 Using the Emulator

The Oracle Java ME SDK 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 continues from Chapter 2, "Using the Oracle Java ME SDK Software with NetBeans." For the Eclipse IDE, some small differences may appear. On the whole, the descriptions of the emulator for the NetBeans IDE map closely to the emulator for the Eclipse IDE.

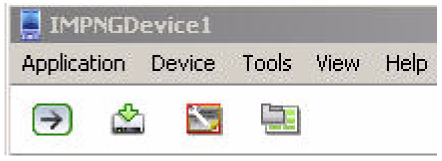
The Oracle Java ME SDK emulator can also be started from the Windows command line and can be run without starting either NetBeans or Eclipse. For more information on using the emulator from the command line, see Appendix B, "Using the Command Line Emulator."

The Emulator's Main Screen

The Oracle Java ME SDK emulator provides several ways to interact with an emulated device (for example, the IMPNGDevice1), as shown in Figure 3–1.

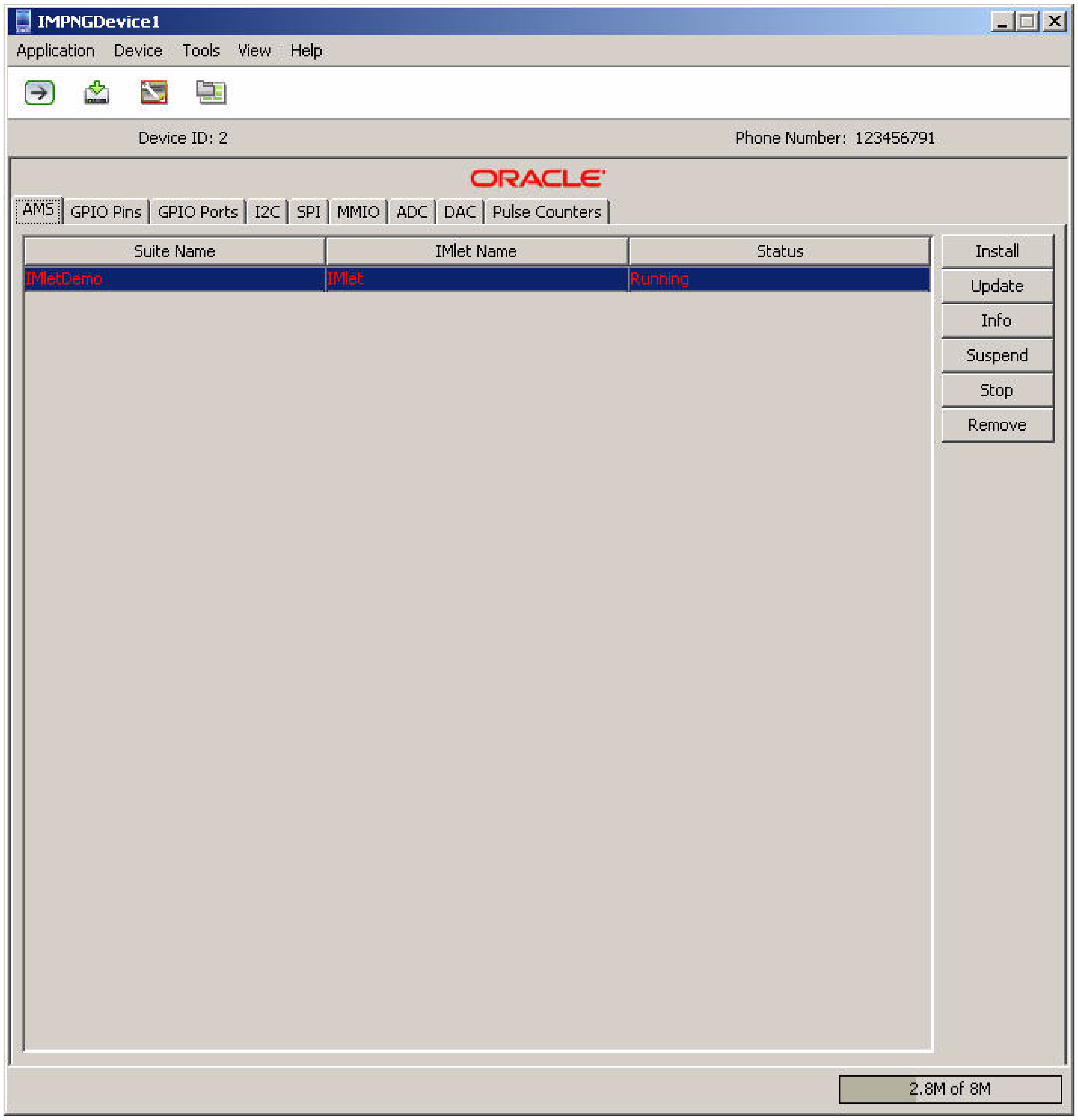
- The Application menu (and associated icons) allow you to install and run IMlet suites.
- The Tools menu allows you to manage a Landmark Store or file system.
- The External Events Generator icon (and Tools menu option) allows you to configure pins, ports, interfaces, and other customizable parts.
- The View menu allows you to display log information and other output in a console window.
- The Help menu allows you to display context-sensitive help to get more information about the emulator screen and functions.

Figure 3–1 The Oracle Java ME SDK Emulator Toolbar and Menu Items



When the Oracle Java ME SDK emulator starts, it appears with the Application Management System (AMS) tab selected as the default, as shown in Figure 3–2.

Figure 3–2 The Oracle Java ME SDK Emulator (AMS Tab) Default Screen



Suite Name	IMlet Name	Status
IMletDemo	IMlet	Running

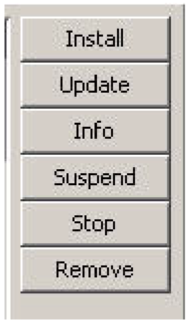
The AMS Tab

The first tab in the emulator, the AMS tab, displays which Oracle Java ME SDK applications are installed, including those that are currently running or stopped.

You can use the buttons in the AMS tab, as shown in Figure 3–3, to do the following:

- Install or update additional applications
- Obtain information about a currently-selected application
- Start, Stop, or Suspend an application
- Remove (uninstall) an application from the AMS

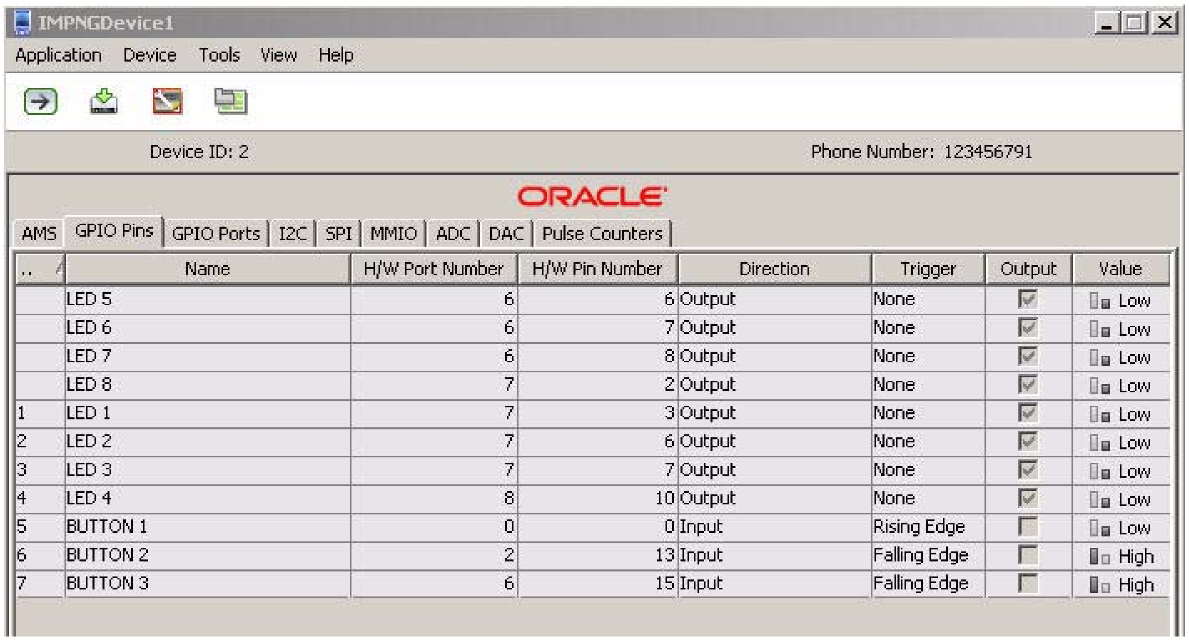
Figure 3–3 Buttons on the AMS Tab



The GPIO Pins Tab

The GPIO tab lists the emulator's current General Purpose I/O (GPIO) pins and their directional state (output). For GPIO pins, the current value is shown as *low* or *high*. The GPIO Pins tab is shown in Figure 3–4:

Figure 3–4 The Emulator GPIO Pins Tab



Device ID: 2

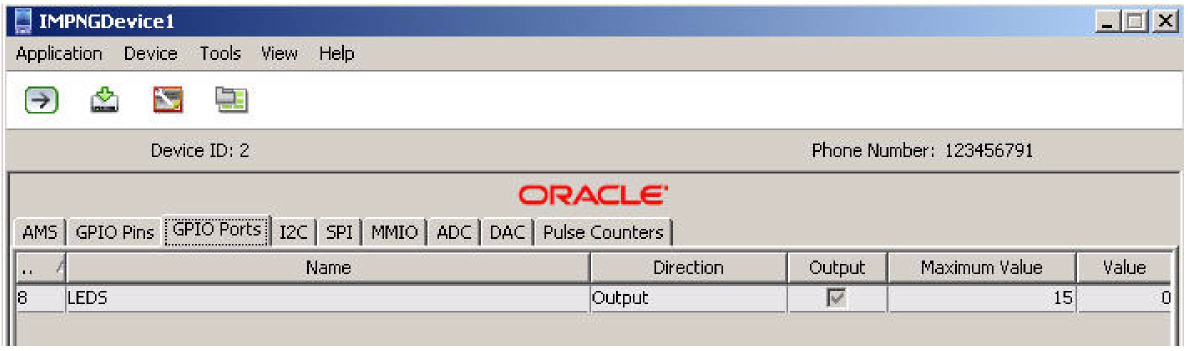
Phone Number: 123456791

..	Name	H/W Port Number	H/W Pin Number	Direction	Trigger	Output	Value
	LED 5	6	6	Output	None	✓	Low
	LED 6	6	7	Output	None	✓	Low
	LED 7	6	8	Output	None	✓	Low
	LED 8	7	2	Output	None	✓	Low
1	LED 1	7	3	Output	None	✓	Low
2	LED 2	7	6	Output	None	✓	Low
3	LED 3	7	7	Output	None	✓	Low
4	LED 4	8	10	Output	None	✓	Low
5	BUTTON 1	0	0	Input	Rising Edge	☐	Low
6	BUTTON 2	2	13	Input	Falling Edge	☐	High
7	BUTTON 3	6	15	Input	Falling Edge	☐	High

The GPIO Ports Tab

For GPIO ports, the port ID and port Name is shown, along with the port's Maximum Value and current Value., as shown in Figure 3–5.

Figure 3–5 The Emulator GPIO Ports Tab



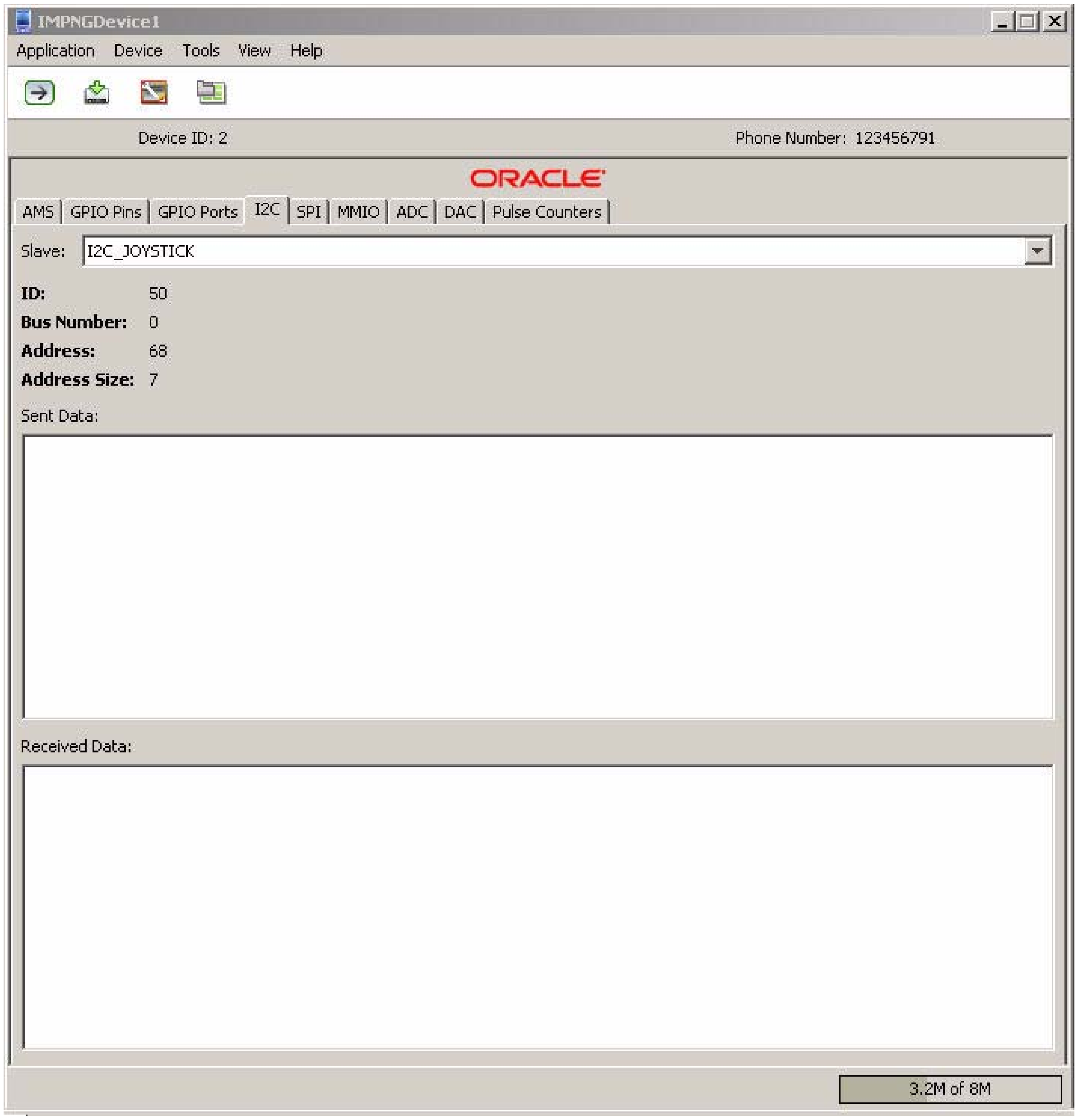
Device ID: 2 Phone Number: 123456791

..	Name	Direction	Output	Maximum Value	Value
8	LEDS	Output	✓	15	0

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 3–6:

Figure 3–6 The Emulator I2C Tab



Device ID: 2

Phone Number: 123456791

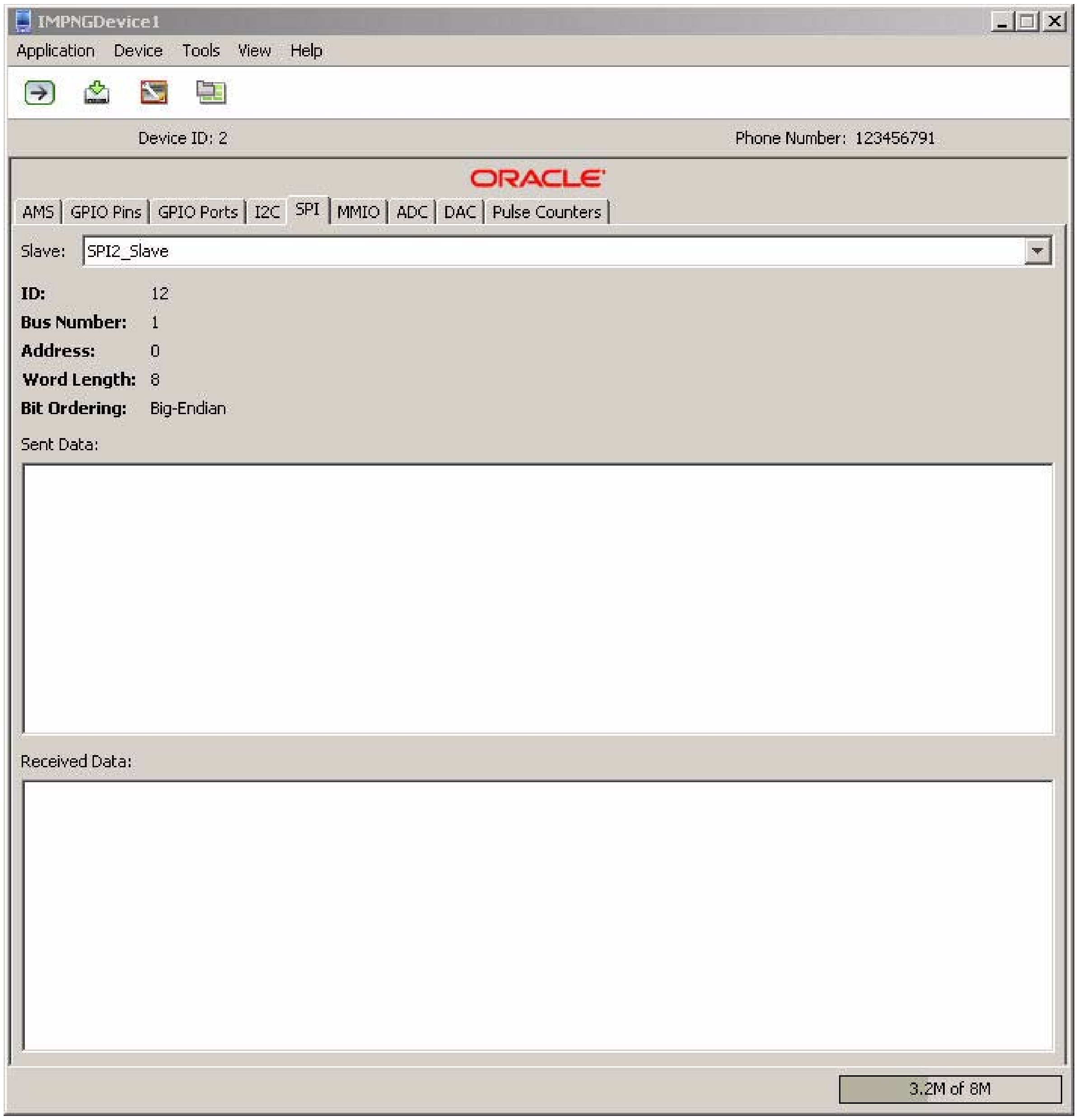
Slave: I2C_JOYSTICK

ID: 50
Bus Number: 0
Address: 68
Address Size: 7

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 3–7:

Figure 3–7 The Emulator SPI Tab


IMPNGDevice1
Application Device Tools View Help
ORACLE
AMS GPIO Pins GPIO Ports I2C SPI MMIO ADC DAC Pulse Counters

Device ID: 2

Phone Number: 123456791

Slave: SPI2_Slave

ID: 12

Bus Number: 1

Address: 0

Word Length: 8

Bit Ordering: Big-Endian

Sent Data:

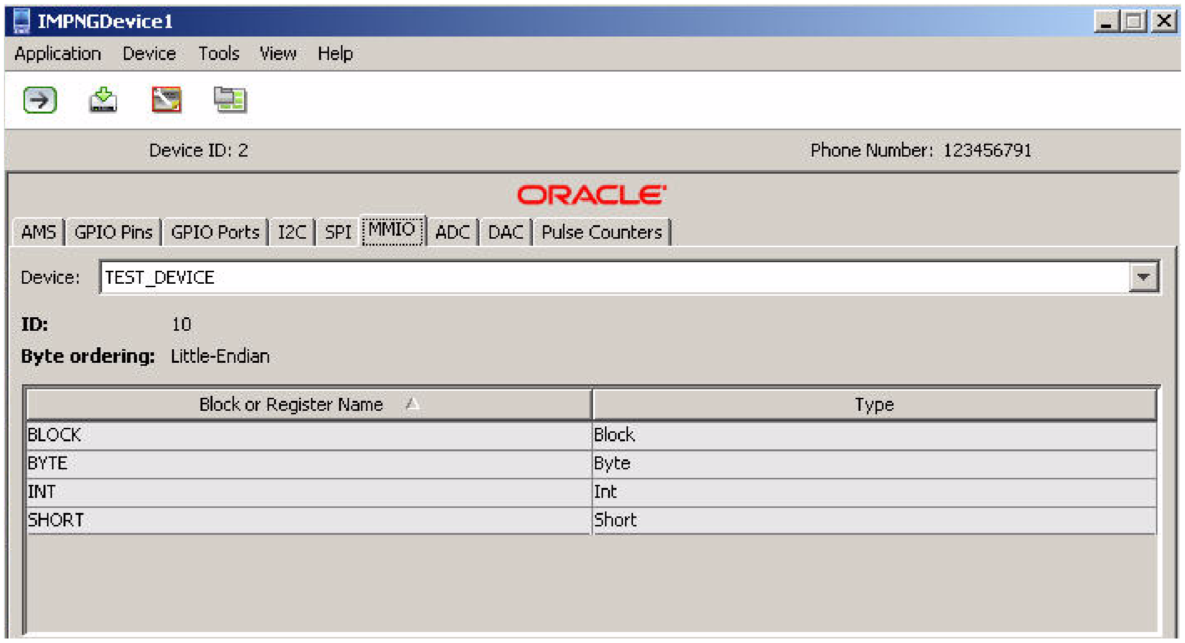
Received Data:

3.2M of 8M

The MMIO Tab

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, WDOGLOG, RTC, and BIG_ENDIAN_DEVICE. Each type of device displays its appropriate block configuration in an information table, as shown in Figure 3–8:

Figure 3–8 The Emulator MMIO Tab


IMPNGDevice1
Application Device Tools View Help
ORACLE
AMS GPIO Pins GPIO Ports I2C SPI MMIO ADC DAC Pulse Counters

Device ID: 2 Phone Number: 123456791

Device: TEST_DEVICE

ID: 10

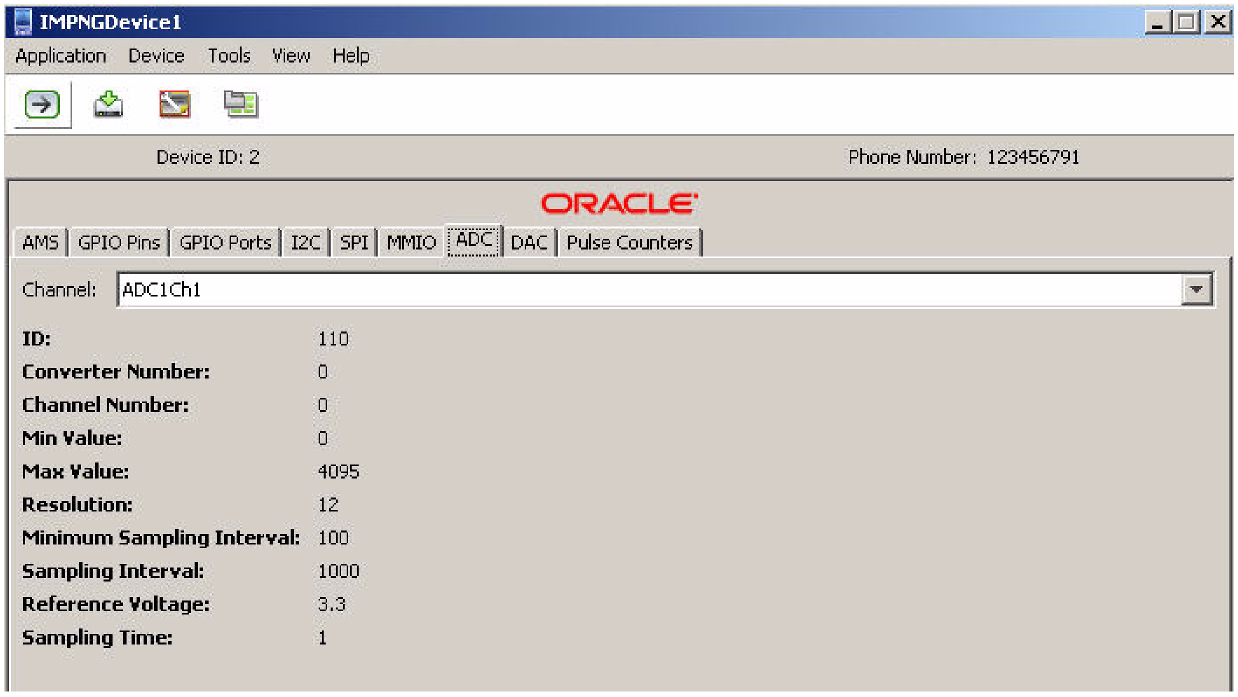
Byte ordering: Little-Endian

Block or Register Name	Type
BLOCK	Block
BYTE	Byte
INT	Int
SHORT	Short

The ADC Tab

The Audio-to-Digital Converter (ADC) tab displays current Channel information, such as ID, Converter Number, Minimum and Maximum Values, and Sampling Intervals, as shown in Figure 3–9.

Figure 3–9 The Emulator ADC Tab


IMPNGDevice1
Application Device Tools View Help
ORACLE
AMS GPIO Pins GPIO Ports I2C SPI MMIO ADC DAC Pulse Counters

Device ID: 2 Phone Number: 123456791

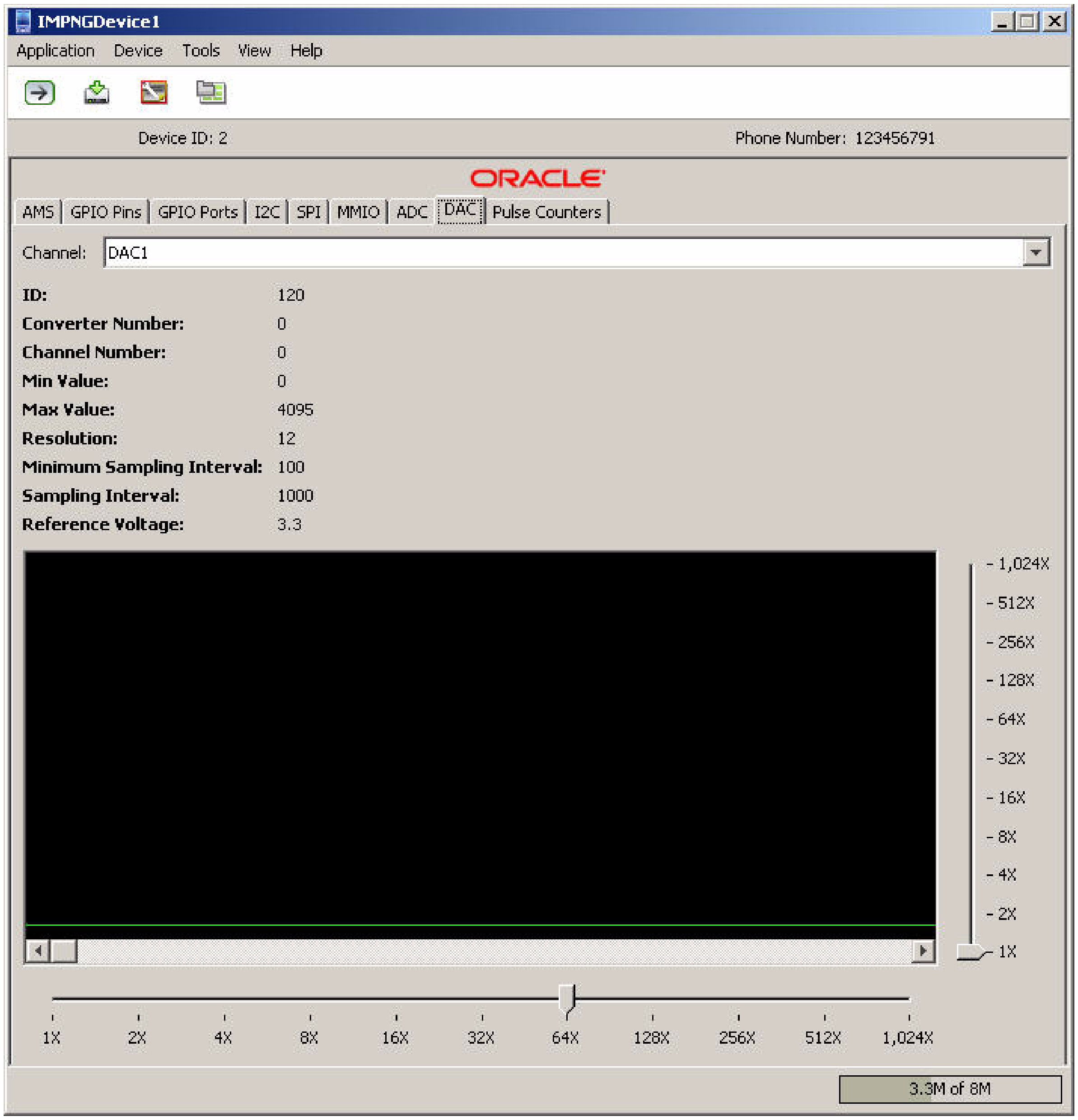
Channel: ADC1Ch1

ID:	110
Converter Number:	0
Channel Number:	0
Min Value:	0
Max Value:	4095
Resolution:	12
Minimum Sampling Interval:	100
Sampling Interval:	1000
Reference Voltage:	3.3
Sampling Time:	1

The DAC Tab

The Digital-to-Audio Converter (DAC) tab displays current Channel information, such as Minimum and Maximum Values, Sampling Interval, and Reference Voltage, as shown in Figure 3–10.

Figure 3–10 The Emulator DAC Tab


IMPNGDevice1
Application Device Tools View Help
Device ID: 2
Phone Number: 123456791
ORACLE
AMS GPIO Pins GPIO Ports I2C SPI MMIO ADC DAC Pulse Counters
Channel: DAC1

ID: 120
Converter Number: 0
Channel Number: 0
Min Value: 0
Max Value: 4095
Resolution: 12
Minimum Sampling Interval: 100
Sampling Interval: 1000
Reference Voltage: 3.3

1,024X
512X
256X
128X
64X
32X
16X
8X
4X
2X
1X
1X 2X 4X 8X 16X 32X 64X 128X 256X 512X 1,024X
3.3M of 8M

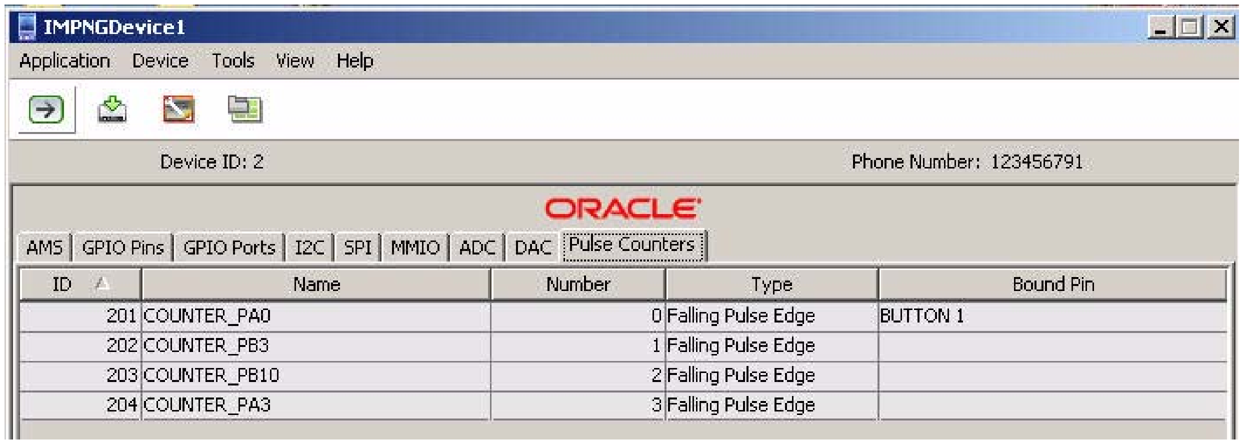
The Pulse Counters Tab

The Pulse Counters tab allows you to send pulse information to as many as four counters. as shown in Figure 3–11. There are four kinds of pulses:

- **Falling Pulse Edge** - Can only be bound to input pins with the Falling Edge trigger.
- **Rising Pulse Edge** - Can only be bound to input pins with the Rising Edge trigger.

- **Negative Pulse Edge** - Can only be bound to input pins with the Both Edges trigger.
- **Positive Pulse Edge** - Can only be bound to input pins with the Both Edges trigger.

Figure 3–11 The Emulator Pulse Counters Tab



Device ID: 2   Phone Number: 123456791

ID	Name	Number	Type	Bound Pin
201	COUNTER_PA0	0	Falling Pulse Edge	BUTTON 1
202	COUNTER_PB3	1	Falling Pulse Edge	
203	COUNTER_PB10	2	Falling Pulse Edge	
204	COUNTER_PA3	3	Falling Pulse Edge	

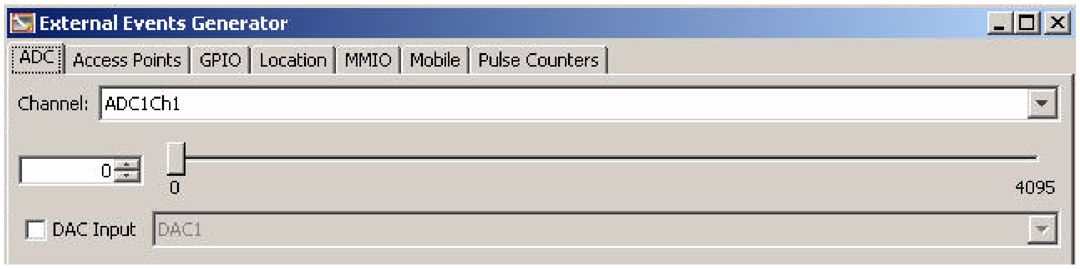
The External Events Generator

The External Events Generator allows you to test the capabilities of your device by simulating events on that device. For example, you can send pulses to a pulse counter or define an access point for network connectivity.

The External Events Generator is started by clicking on the External Events Generator tab, as shown in Figure 3–1, or by selecting Tools --> External Events Generator.

When the External Events Generator is started the default tab is the Audio-to-Digital Converter tab, as shown in Figure 3–12.

Figure 3–12 The External Events Generator ADC Tab



Channel: ADC1Ch1

0

DAC Input DAC1

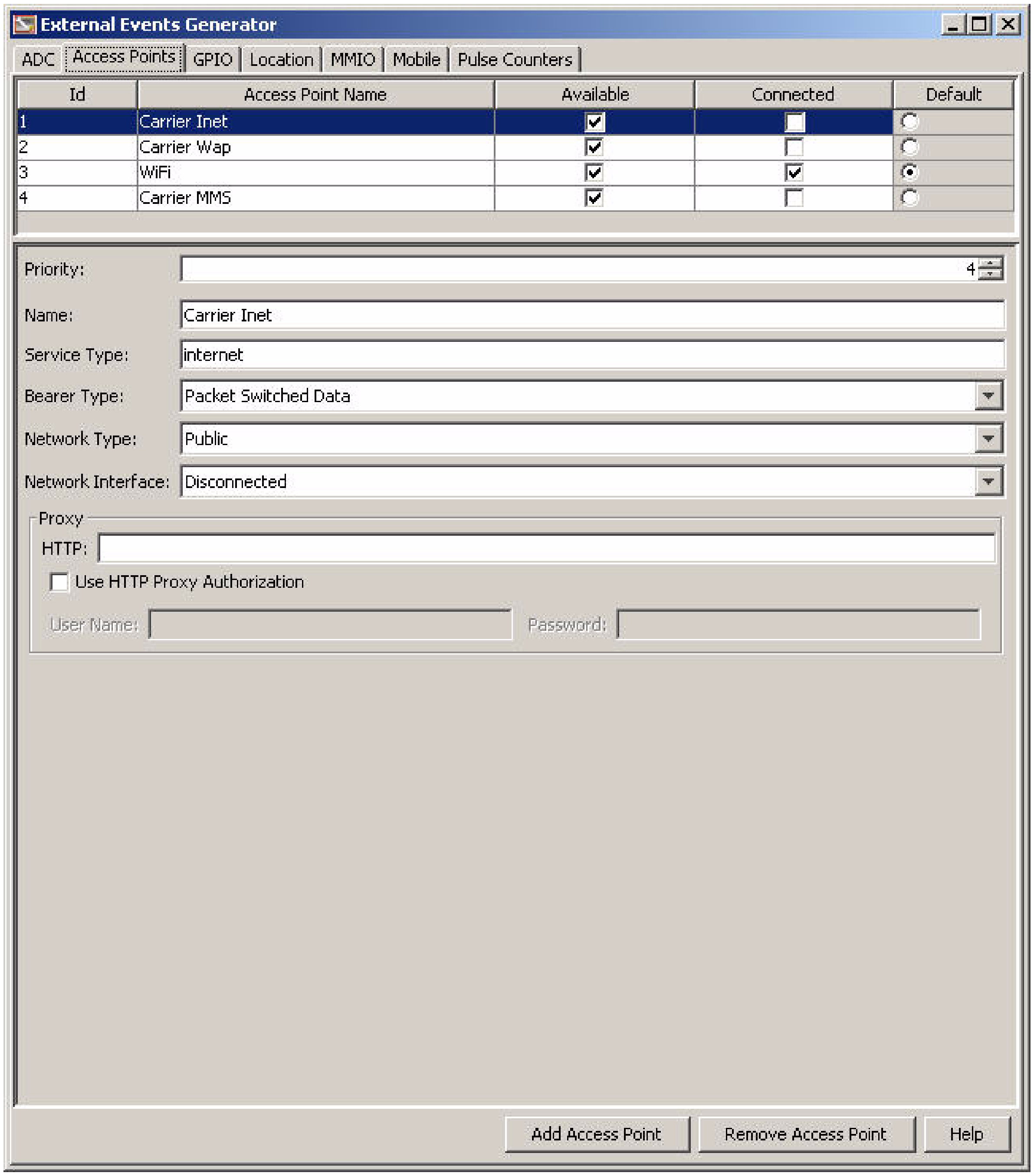
The ADC Tab

The Audio-to-Digital Converter (ADC) tab allows you to test audio input, as shown in Figure 3–12, including Channel, DAC Input, and minimum and maximum values.

The Access Points Tab

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, as shown in Figure 3–13.

Figure 3–13 The External Events Generator Access Points Tab



Id	Access Point Name	Available	Connected	Default
1	Carrier Inet	✓		○
2	Carrier Wap	✓		○
3	WiFi	✓	✓	●
4	Carrier MMS	✓		○

Priority:	4
Name:	Carrier Inet
Service Type:	internet
Bearer Type:	Packet Switched Data
Network Type:	Public
Network Interface:	Disconnected

Proxy

HTTP:

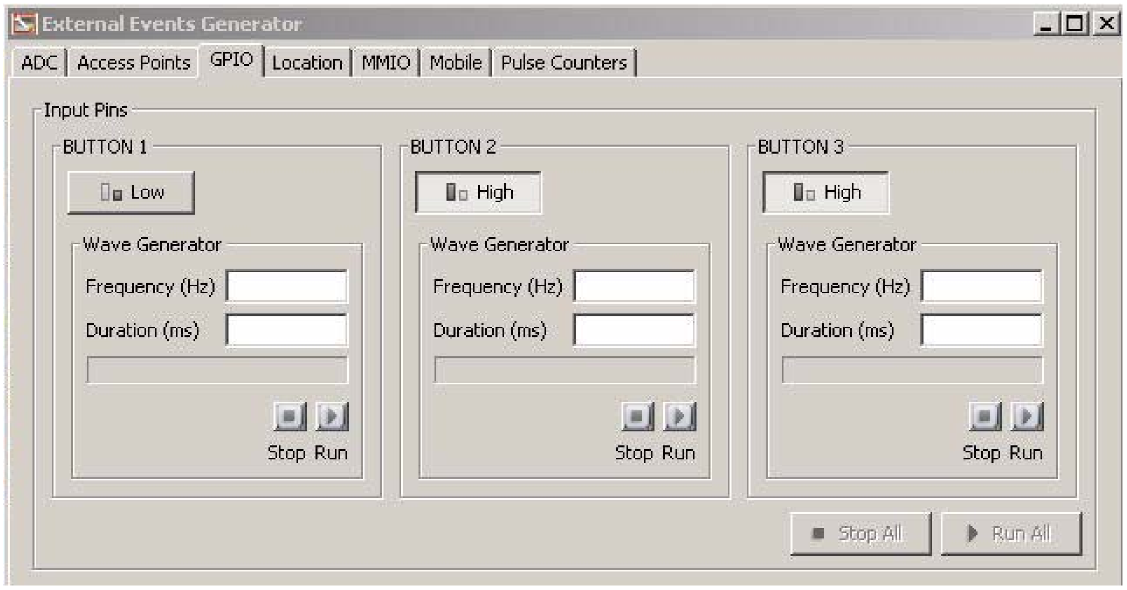
☐ Use HTTP Proxy Authorization

User Name: Password:

The GPIO Tab

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 3–14.

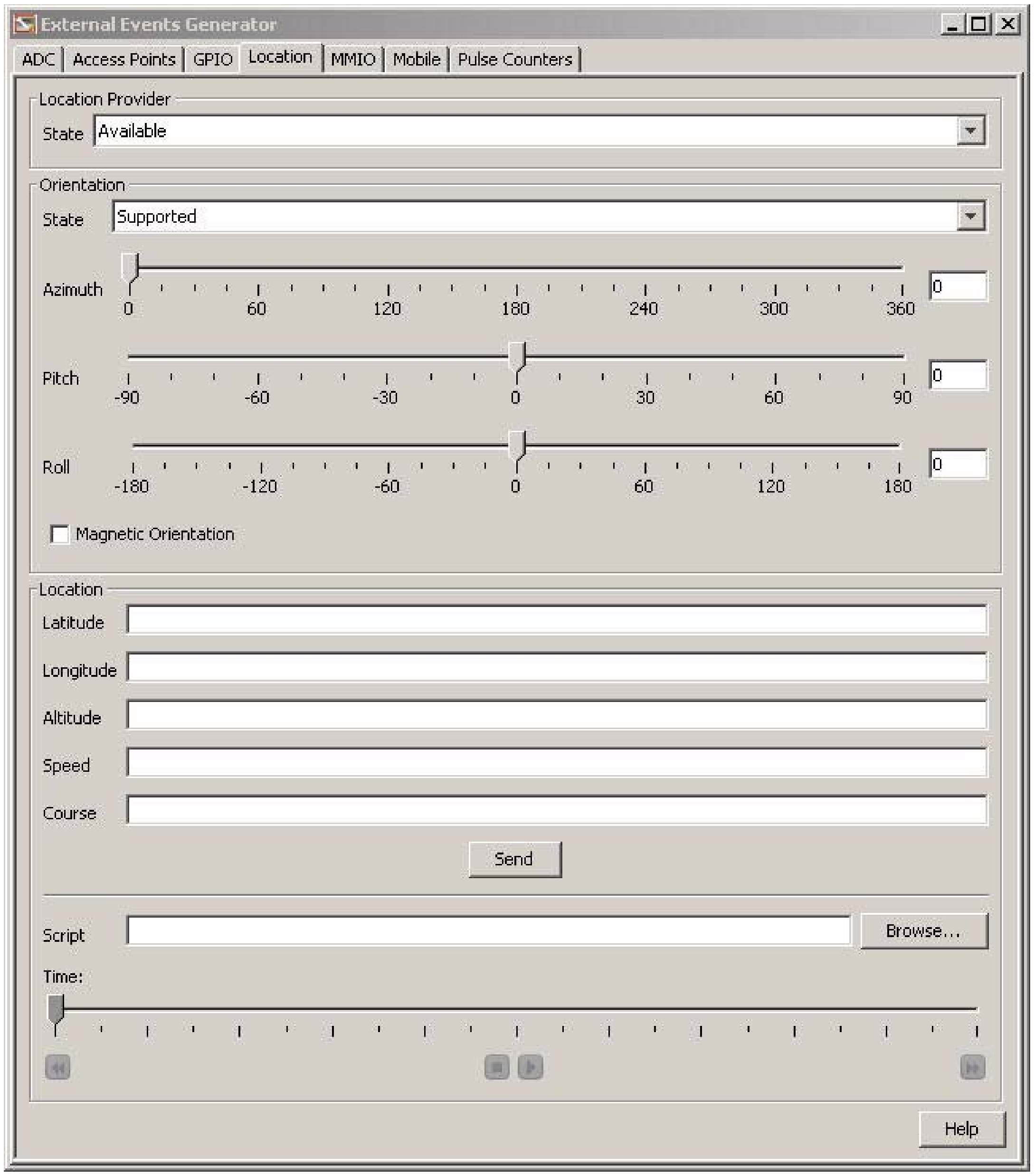
Figure 3–14 The External Events Generator GPIO Tab



The Location Tab

The Location tab allows the device to specify the simulated Location Provider, Orientation, and other settings, such as Latitude, Longitude, Altitude, Speed, and Course, as shown in Figure 3–15.

Figure 3–15 The External Events Generator Location Tab


External Events Generator

ADC | Access Points | GPIO | Location | MMIO | Mobile | Pulse Counters

Location Provider
State: Available

Orientation
State: Supported
Azimuth: 0
Pitch: 0
Roll: 0
Magnetic Orientation

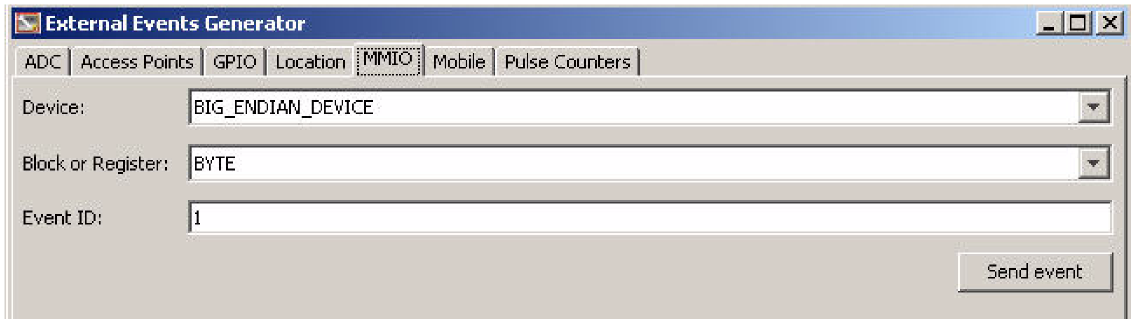
Location
Latitude
Longitude
Altitude
Speed
Course

Script
Time:

The MMIO Tab

The MMIO tab of the External Events Generator allows the user to simulate sending input event IDs to one of the four different peripheral device types. This tab is shown in Figure 3–16.

Figure 3–16 The External Events Generator MMIO Tab



Device: BIG_ENDIAN_DEVICE

Block or Register: BYTE

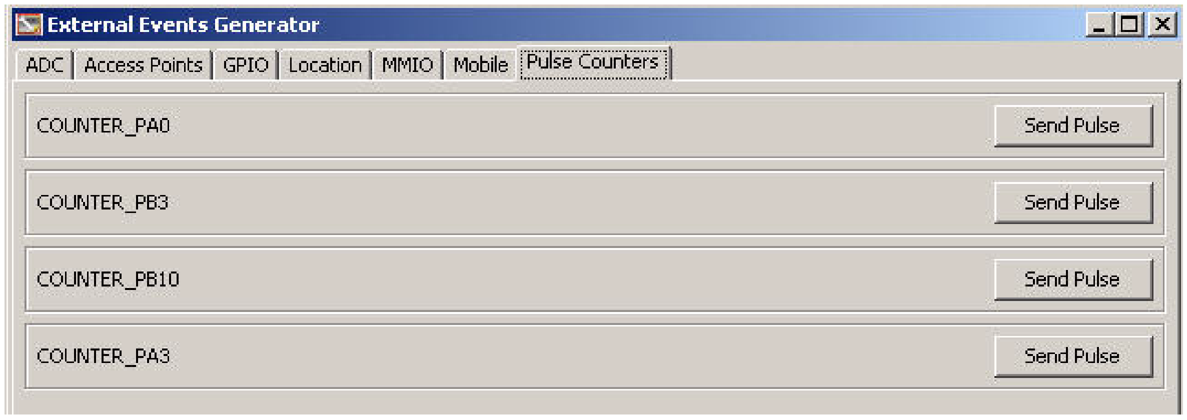
Event ID: 1

Note: The External Events Generator Mobile tab is not used by Embedded Applications so it is not documented here.

The Pulse Counter Tab

The Pulse Counter tab allows you to send a a sequence of pulses to as many as four pulse counters, as shown in Figure 3–17.

Figure 3–17 The External Events Generator Pulse Counters Tab

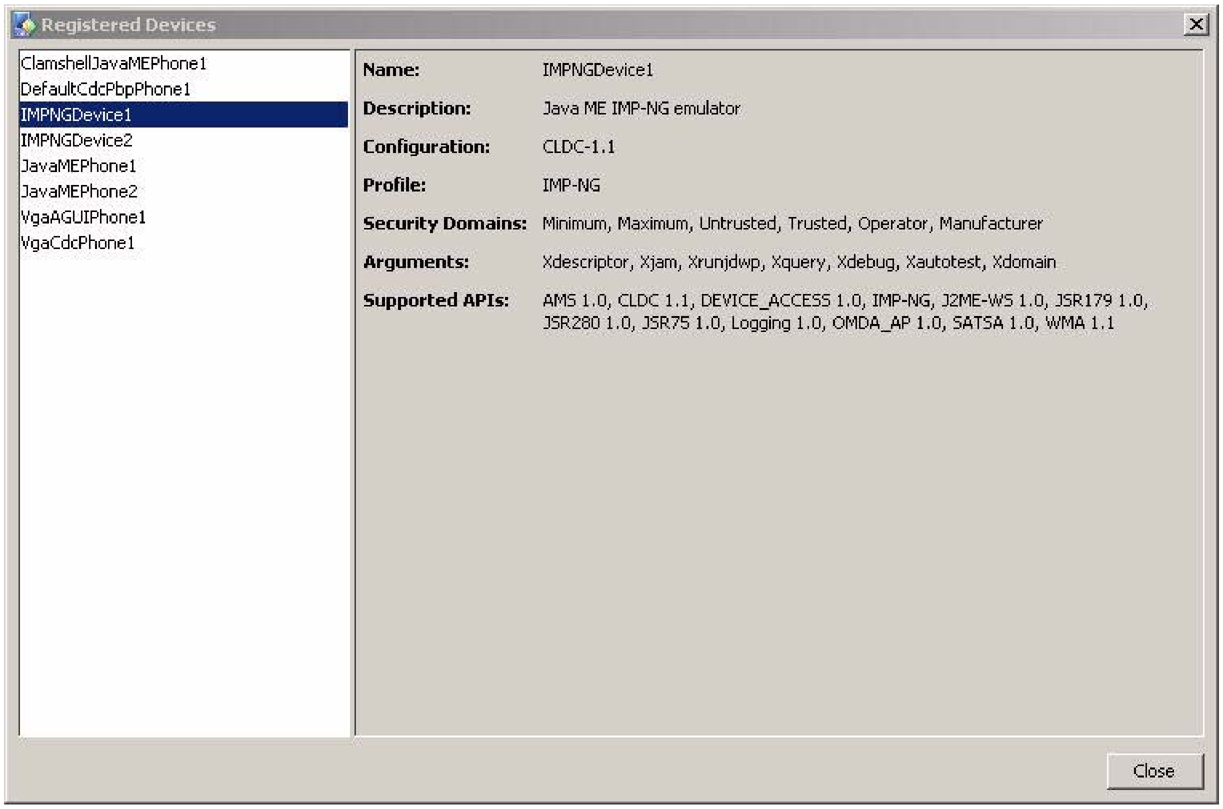


Using the Device Manager

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. The list of registered devices is shown in Figure 3–18.

Note: The Oracle Java ME SDK supports many device types, but only two are used by Embedded Applications: IMPNGDevice1 and IMPNGDevice2.

Figure 3–18 A List of Registered Devices in the Device Manager

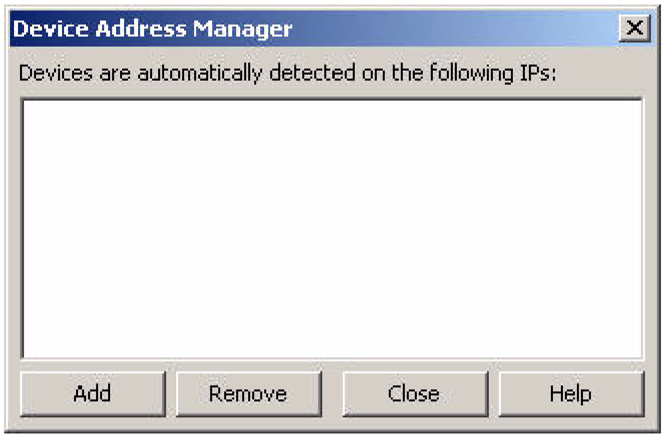


Name:	IMPNGDevice1
Description:	Java ME IMP-NG emulator
Configuration:	CLDC-1.1
Profile:	IMP-NG
Security Domains:	Minimum, Maximum, Untrusted, Trusted, Operator, Manufacturer
Arguments:	Xdescriptor, Xjam, Xrunjdwp, Xquery, Xdebug, Xautotest, Xdomain
Supported APIs:	AMS 1.0, CLDC 1.1, DEVICE_ACCESS 1.0, IMP-NG, J2ME-WS 1.0, JSR179 1.0, JSR280 1.0, JSR75 1.0, Logging 1.0, OMDA_AP 1.0, SATSA 1.0, WMA 1.1

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.

The Device Manager also allows you to add the IP address of one or more additional devices using the Device Address Manager, as shown in Figure 3–19.

Figure 3–19 The Device Address Manager



Devices are automatically detected on the following IPs:

4 Using the Oracle Java ME SDK Software with Eclipse

Using the Eclipse integrated development environment, you can create and test embedded applications using a graphical development environment.

Note: The examples in this chapter use the Eclipse Indigo 3.7 platform.

Installing the Oracle Java ME SDK Eclipse Plugin

The Oracle Java ME SDK Plugins are the same for both the Indigo 3.7 and Juno 4.2 platforms.

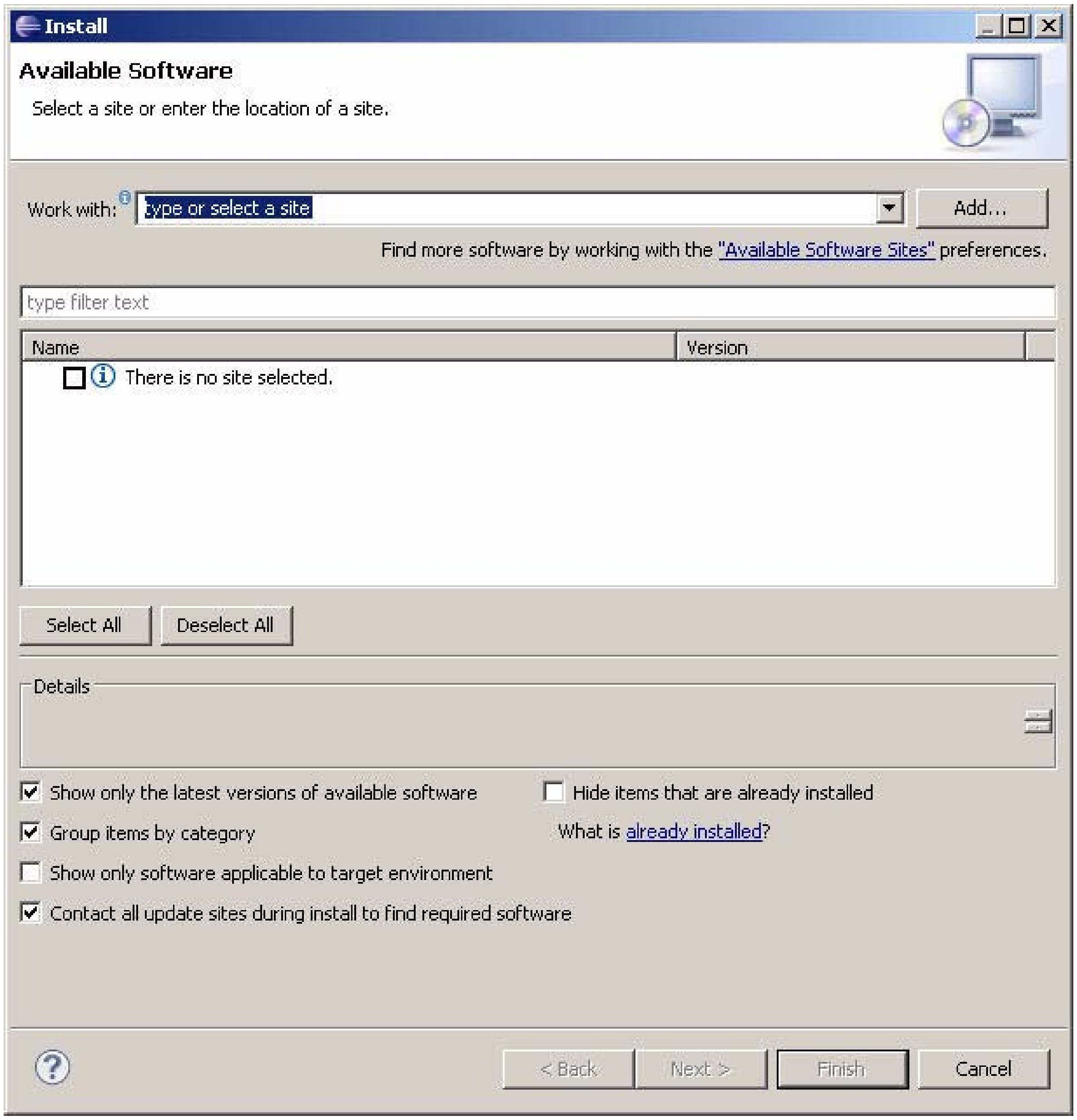
To download the Oracle Java ME SDK Plugins file for Eclipse (`oracle-jmesdk-3-3-rr-eclipse-plugins.zip`), go to the following location:

`http://www.oracle.com/technetwork/java/javame/javamobile/download/sdk`

Installing Plugins on Eclipse Indigo 3.7

1. In Eclipse, go to Help > Install New Software. This displays the Available Software screen, as shown in Figure 4–1.

Figure 4–1 The Eclipse Available Software Screen



Install

Available Software

Select a site or enter the location of a site.

Work with: type or select a site

Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
☐ There is no site selected.	

Details

[✓] Show only the latest versions of available software

[✓] Group items by category

[ ] Show only software applicable to target environment

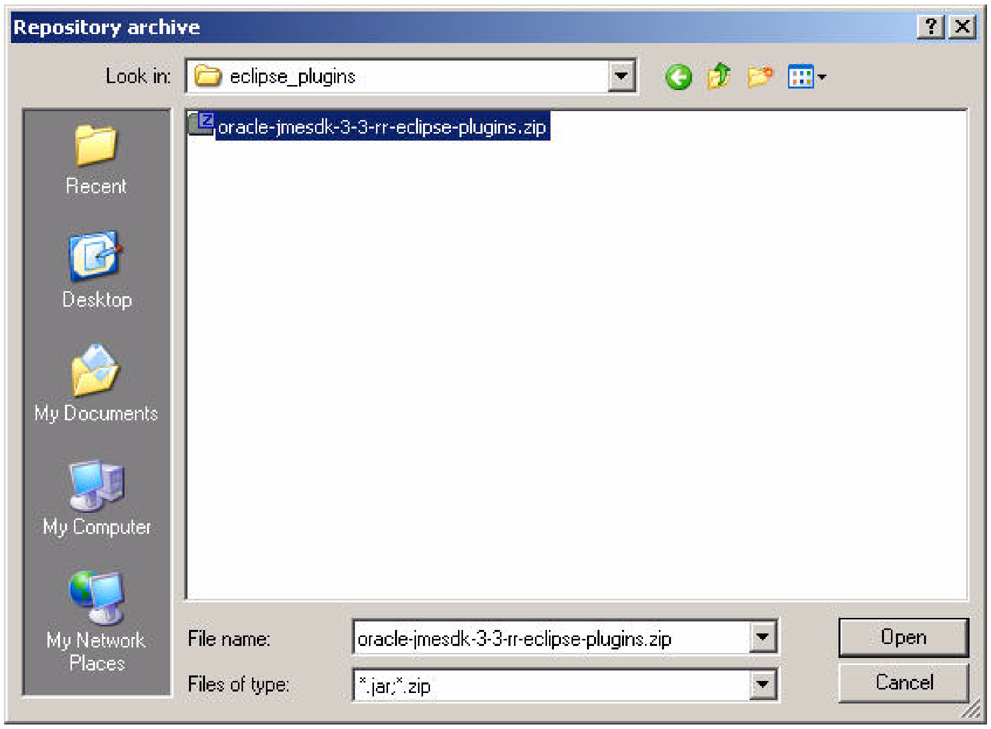
[✓] Contact all update sites during install to find required software

[ ] Hide items that are already installed

What is already installed?

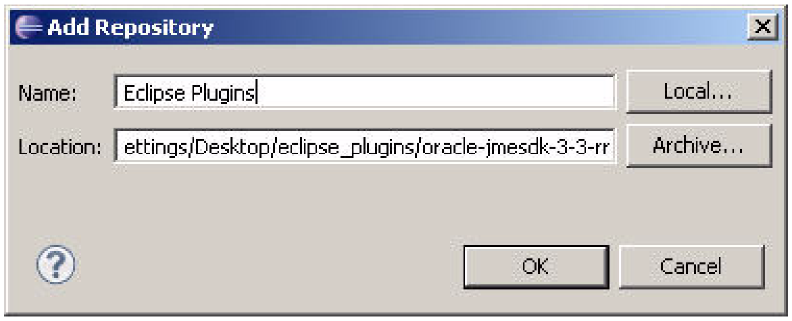
2. Press the Add button. This displays the Add Repository window.
3. In the Add Repository window, press the Archive button to open a file browser.
4. In the file browser, select the Eclipse plugin file that you downloaded, as shown in Figure 4–2. Click Open to return to the previous dialog.

Figure 4–2 Selecting Eclipse Plugins


Repository archive
Look in: eclipse_plugins
oracle-jmesdk-3-3-rr-eclipse-plugins.zip
Recent
Desktop
My Documents
My Computer
My Network Places
File name: oracle-jmesdk-3-3-rr-eclipse-plugins.zip
Files of type: *.jar;*.zip
Open
Cancel

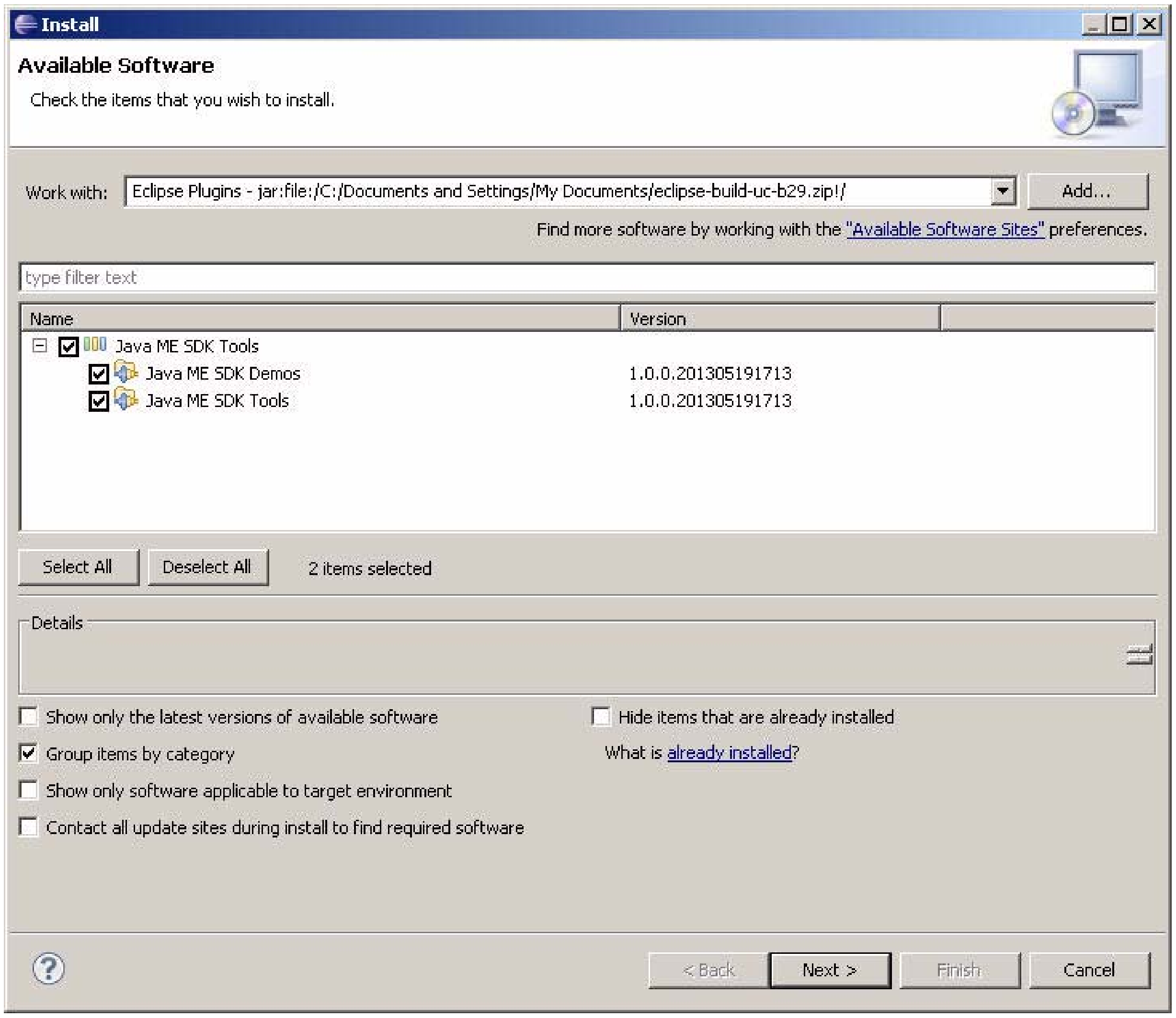
5. The Add Repository window appears as shown in Figure 4–3. Press OK.

Figure 4–3 The Add Repository Window with Eclipse Plugins


Add Repository
Name: Eclipse Plugins
Location: ettings/Desktop/eclipse_plugins/oracle-jmesdk-3-3-rr
Local...
Archive...
OK
Cancel

6. The category Java ME SDK Tools appears in the Name area. Check the box in front of Java ME SDK Tools and ME SDK Demos as shown in Figure 4–4. Click Next.

Figure 4–4 Selecting Oracle Java ME SDK Plugins



Install

Available Software

Check the items that you wish to install.

Work with: Eclipse Plugins - jar:file:/C:/Documents and Settings/My Documents/eclipse-build-uc-b29.zip!/ [Add...]

Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
☑ Java ME SDK Tools	
☑ Java ME SDK Demos	1.0.0.201305191713
☑ Java ME SDK Tools	1.0.0.201305191713

[Select All] [Deselect All] 2 items selected

Details

☐ Show only the latest versions of available software

☑ Group items by category

☐ Show only software applicable to target environment

☐ Contact all update sites during install to find required software

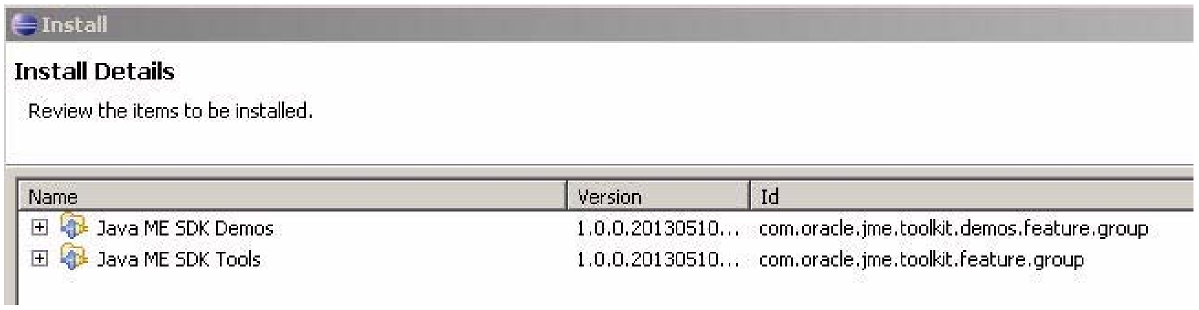
☐ Hide items that are already installed

What is already installed?

[< Back] [Next >] [Finish] [Cancel]

7. The installation details are displayed. Press Next. The items you checked are listed, as shown in Figure 4–5. Press Next again.

Figure 4–5 Eclipse Plugins Installation Details



Install

Install Details

Review the items to be installed.

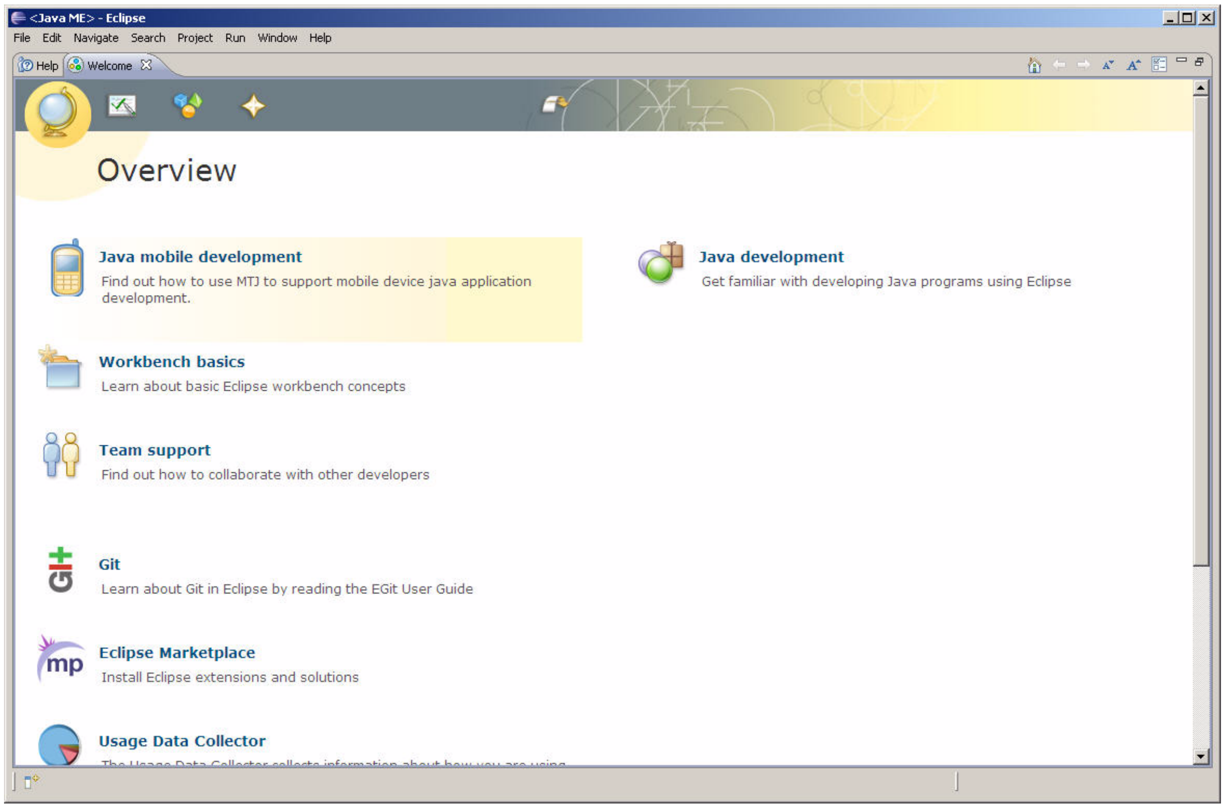
Name	Version	Id
Java ME SDK Demos	1.0.0.20130510...	com.oracle.jme.toolkit.demos.feature.group
Java ME SDK Tools	1.0.0.20130510...	com.oracle.jme.toolkit.feature.group

8. Accept the terms of the license agreement and click Finish.

9. Check the Java ME SDK Tools Category and its subcategories and press the Next button to start the Add wizard.

10. Accept the terms of the license agreement and click Finish. The installation process starts.
11. When the installation process completes, restart Eclipse. The Eclipse Welcome Screen is displayed, as shown in Figure 4–6.

Figure 4–6 The Eclipse Welcome Screen



Overview

Java mobile development
Find out how to use MTJ to support mobile device java application development.

Java development
Get familiar with developing Java programs using Eclipse

Workbench basics
Learn about basic Eclipse workbench concepts

Team support
Find out how to collaborate with other developers

Git
Learn about Git in Eclipse by reading the EGit User Guide

Eclipse Marketplace
Install Eclipse extensions and solutions

Usage Data Collector

Installing Plugins on Eclipse Juno 4.2

On Eclipse Juno 4.2, installing the Oracle Java ME SDK plugins requires the Eclipse Mobile Tools for Java (MTJ) toolkit. The MTJ is not bundled with Juno and must be installed manually, before installing the Eclipse plugins.

To install the Eclipse Mobile Tools for Java toolkit:

1. Go to Help > Install New Software.
2. In the “Work with” field, enter this URL:
   `http://download.eclipse.org/releases/indigo`
3. Click Add.
4. When the plugins are discovered, open the Mobile and Device Development Tools hierarchy and check Mobile Tools for Java Examples and Mobile Tools for Java SDK.
5. Uncheck the "Contact all update sites...." checkbox and click Next.
6. On the installation screen select all plugins and click Finish. Then click OK.

A Juno plugin named `org.eclipse.jetty.server_<version>.jar` conflicts with the MTJ libraries when the plugin version is higher than 6.

If the `org.eclipse.jetty.server` version is higher than 6 you must prevent it from loading by changing its name so the JAR file is not parsed. Locate the file in the `\plugins` directory of your Juno installation and change the extension from `.jar` to something else.

For example, change it to: `org.eclipse.jetty.server_<version>`**.old.**

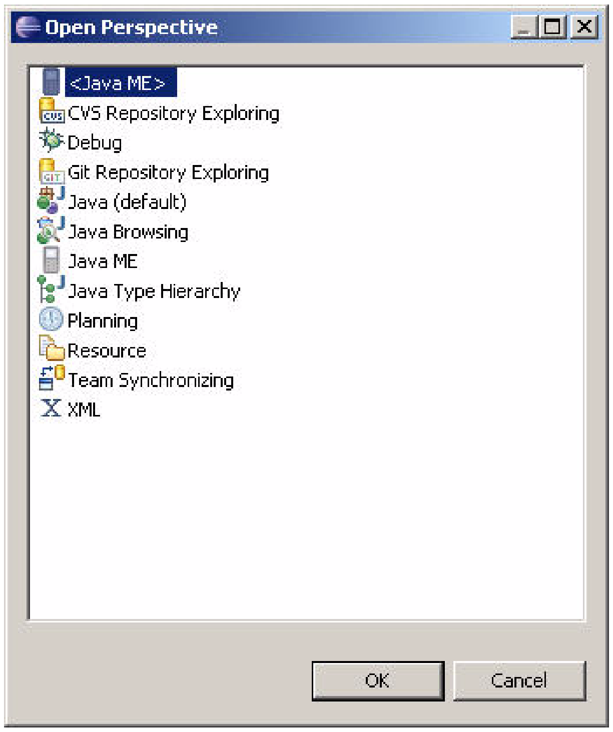
Note: Once you have installed the MTJ toolkit for Eclipse Juno 4.2, the procedure for installing the Oracle Java ME SDK plugins is the same as for Eclipse Indigo 3.7. See "Installing Plugins on Eclipse Indigo 3.7".

Configuring Eclipse

To configure Eclipse, follow these steps:

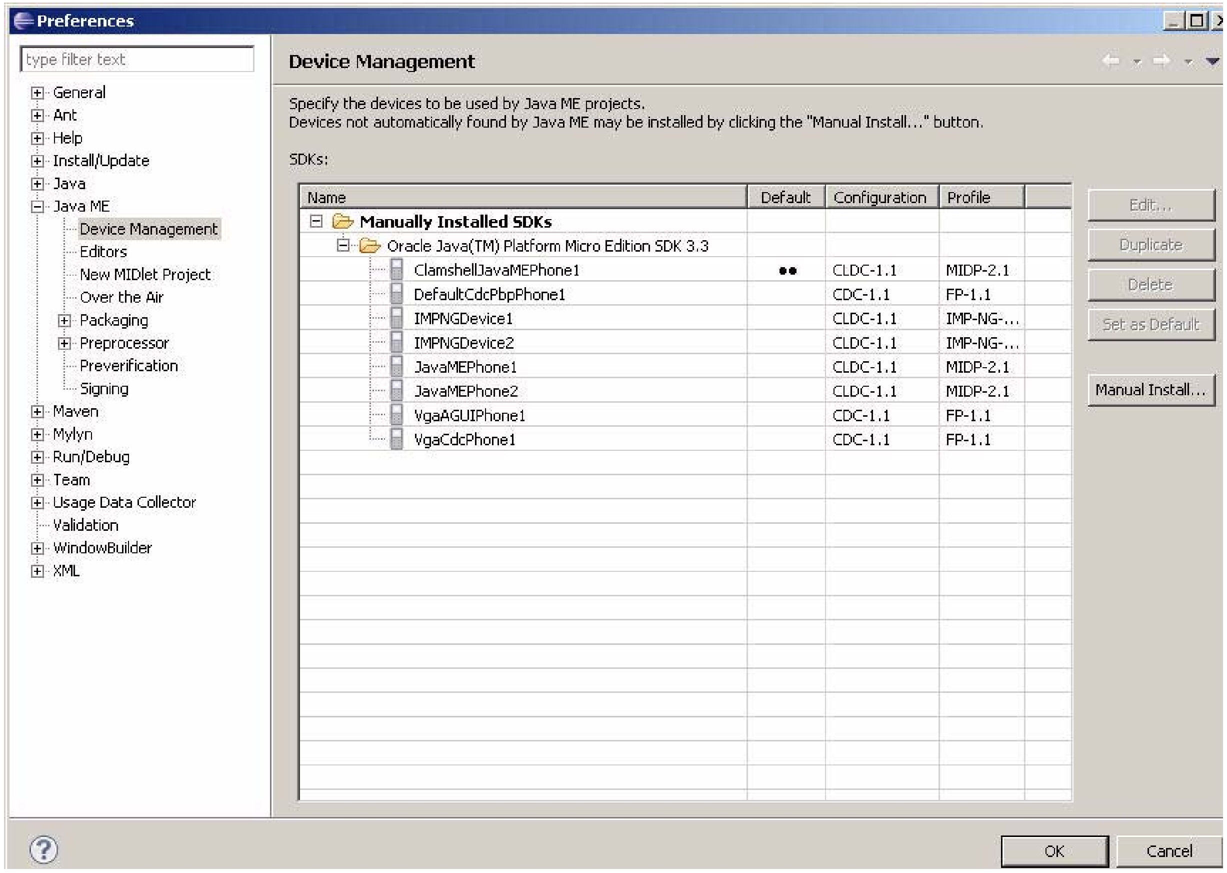
1. Select Window > Open Perspective > Other and choose Java ME, as shown in Figure 4–7. Click OK.

Figure 4–7 Selecting Java ME for Configuration



2. Select Window > Preferences > and expand the Java ME node.
3. Select Device Management. The Device Management page displays a list of devices used by Oracle Java ME SDK projects, as shown in Figure 4–8.

Figure 4–8 The Device Management Screen



Preferences

type filter text

- General
- Ant
- Help
- Install/Update
- Java
- Java ME
  - Device Management
  - Editors
  - New MIDlet Project
  - Over the Air
  - Packaging
  - Preprocessor
  - Preverification
  - Signing
- Maven
- Mylyn
- Run/Debug
- Team
- Usage Data Collector
- Validation
- WindowBuilder
- XML

Device Management

Specify the devices to be used by Java ME projects.
Devices not automatically found by Java ME may be installed by clicking the "Manual Install..." button.

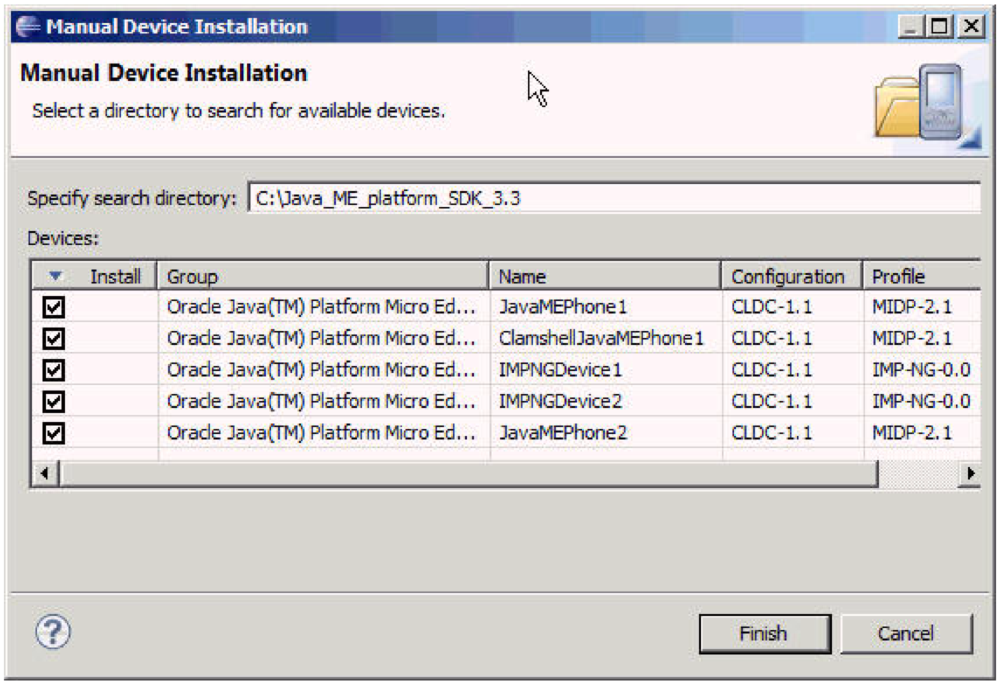
SDKs:

Name	Default	Configuration	Profile
Manually Installed SDKs			
Oracle Java(TM) Platform Micro Edition SDK 3.3			
ClamshellJavaMEPhone1	●●	CLDC-1.1	MIDP-2.1
DefaultCdcPbpPhone1		CDC-1.1	FP-1.1
IMPNGDevice1		CLDC-1.1	IMP-NG-...
IMPNGDevice2		CLDC-1.1	IMP-NG-...
JavaMEPhone1		CLDC-1.1	MIDP-2.1
JavaMEPhone2		CLDC-1.1	MIDP-2.1
VgaAGUIPhone1		CDC-1.1	FP-1.1
VgaCdcPhone1		CDC-1.1	FP-1.1

4. Click the Manual Install... button, to the right of the Device Management screen.
5. In the Specify Search Directory field, enter or browse to the location of the Oracle Java ME SDK platform installation. For example:

   C:\Java_ME_platform_SDK_3.3
6. Click OK. When the five default emulators are detected, make sure all the devices are checked, as shown in Figure 4–9, and click Finish.

Figure 4–9 Manual Device Installation


Manual Device Installation
Manual Device Installation
Select a directory to search for available devices.
Finish
Cancel

Specify search directory: C:\Java_ME_platform_SDK_3.3

Devices:

Install	Group	Name	Configuration	Profile
✓	Oracle Java(TM) Platform Micro Ed...	JavaMEPhone1	CLDC-1.1	MIDP-2.1
✓	Oracle Java(TM) Platform Micro Ed...	ClamshellJavaMEPhone1	CLDC-1.1	MIDP-2.1
✓	Oracle Java(TM) Platform Micro Ed...	IMPNGDevice1	CLDC-1.1	IMP-NG-0.0
✓	Oracle Java(TM) Platform Micro Ed...	IMPNGDevice2	CLDC-1.1	IMP-NG-0.0
✓	Oracle Java(TM) Platform Micro Ed...	JavaMEPhone2	CLDC-1.1	MIDP-2.1

7. Click OK to close the installation window.

   This returns you to the Eclipse main window. The Oracle Java ME SDK is now ready to use.

Creating a Sample IMlet File

In this section, you create a sample IMlet file, `IMletDemo.java`, from the code provided in Example 4–1. This IMlet file is used in the next section, "Creating a New Oracle Java ME SDK Project in Eclipse."

1. Copy the code shown in Example 4–1 into a text file. Use Notepad rather than WordPad, to avoid any unneeded extra characters.
2. Name the file `IMletDemo.java` and Save. Set the file aside for now.

Example 4–1 Code for the Sample IMletDemo.java Project in NetBeans

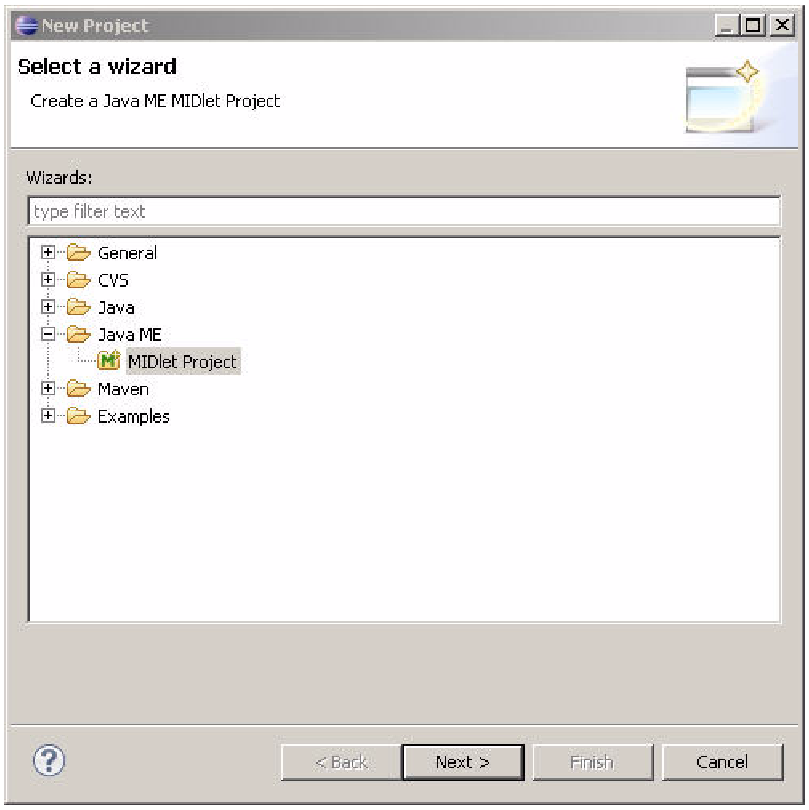
```
import javax.microedition.midlet.MIDlet;

public class IMletDemo extends MIDlet {

     boolean bFirst = false;
     boolean loopFlag = true;


     public void startApp() {

         if (bFirst = false) {

            try {
                // Perform startup operations
            } catch (Exception ex) {
                ex.printStackTrace();
                return;
            }

            bFirst = true;
         } else {
             System.out.println("IMlet Demo is already started...");
         }

         // Start program here

    }

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

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

       // Close all resources that have been opened

    }


}
```

Creating a New Oracle Java ME SDK Project in Eclipse

Follow the steps below to create a new Oracle Java ME SDK project in the Eclipse IDE:

1. Choose the File >New >Project > Java ME > Midlet Project menu item and click Next, as shown in Figure 4–10.

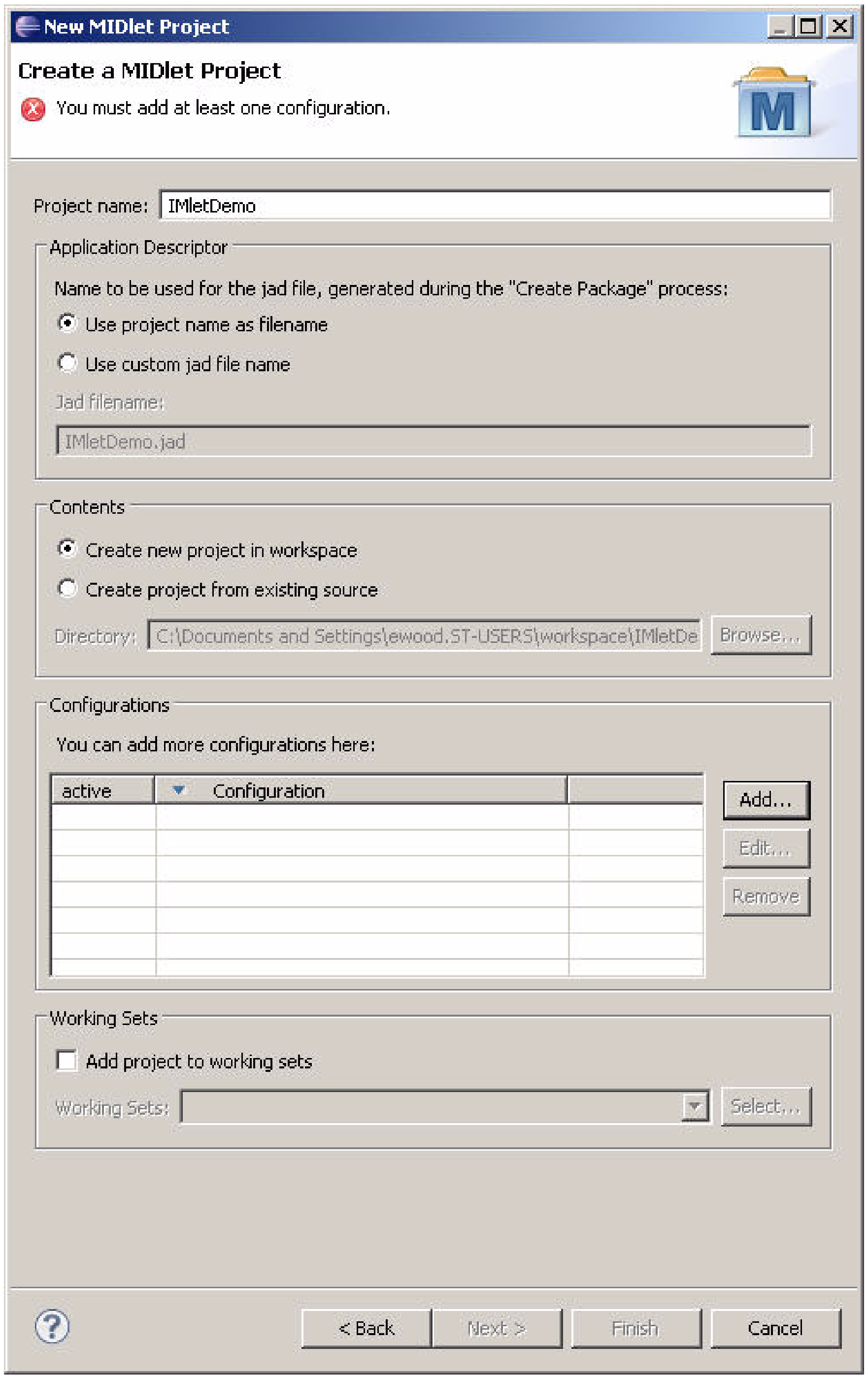
Figure 4–10 Selecting a Project



2. When the Create a MIDlet Project window appears, enter a Project Name (for example, `IMletDemo`).

3. To the right of the Configurations window, press the Add... button to add a configuration. This displays the Add Configuration dialog box, with no configurations, as shown in Figure 4–11.

Figure 4–11 Creating a New MIDlet Project


New MIDlet Project

Create a MIDlet Project

You must add at least one configuration.

Project name: IMletDemo

Application Descriptor

Name to be used for the jad file, generated during the "Create Package" process:

[✓ Use project name as filename]

Use custom jad file name

Jad filename: IMletDemo.jad

Contents

[✓ Create new project in workspace]

Create project from existing source

Directory: C:\Documents and Settings\ewood.ST-USERS\workspace\IMletDe

Configurations

You can add more configurations here:

active	Configuration	

Working Sets

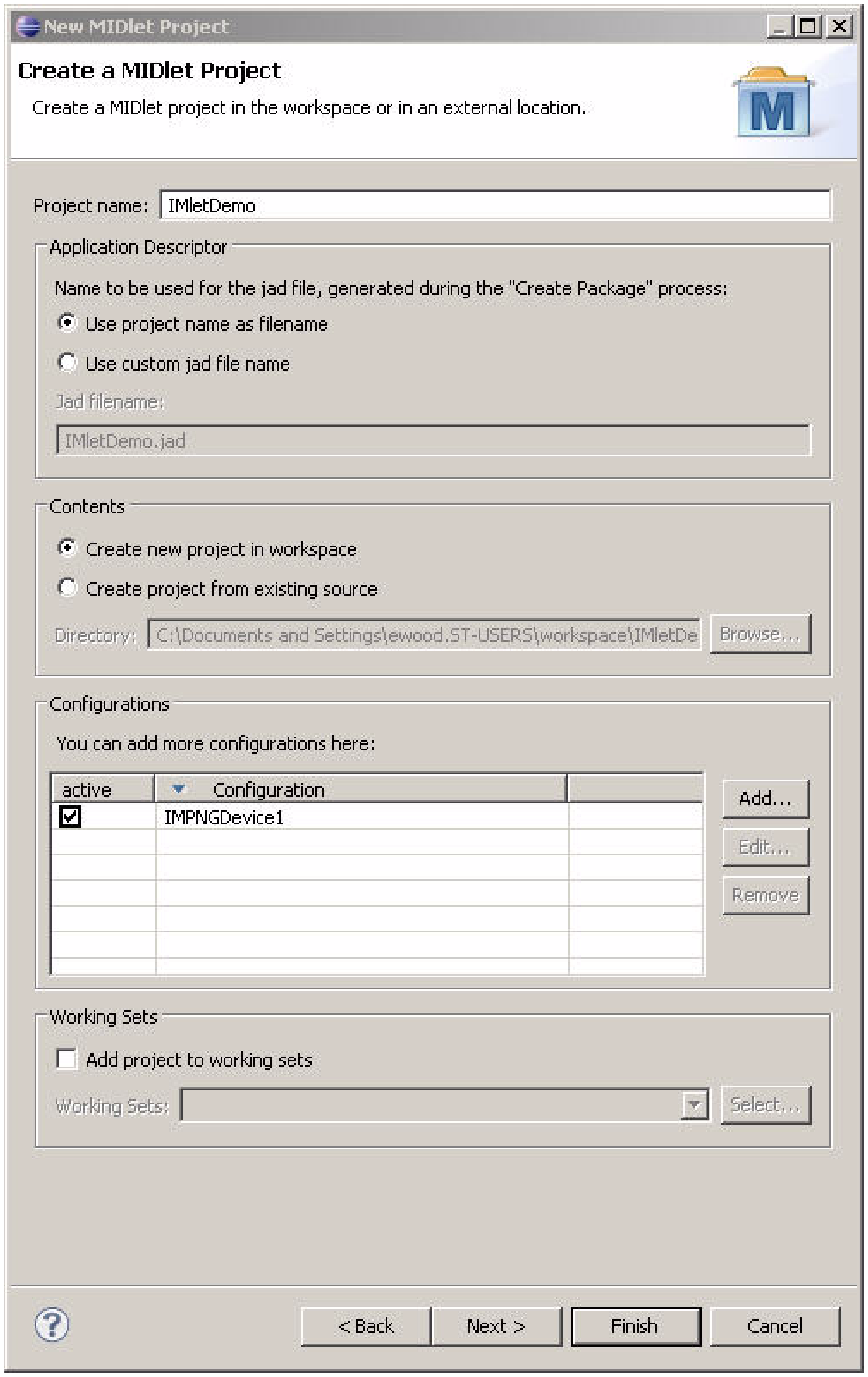
Add project to working sets

Working Sets:

4. In the Add Configuration dialog box, select "Oracle Java(TM) Platform Micro Edition SDK 3.3" from the drop-down menu in the SDK field. In the Device field, select "IMPNGDevice1," as shown in Figure 4–12.

5. Press Finish. This displays the New MIDlet Project window again, with `IMPNGDevice1` selected as the Active Configuration, as shown in Figure 4–12.

Figure 4–12 Creating a New Active Configuration


New MIDlet Project

Create a MIDlet Project

Create a MIDlet project in the workspace or in an external location.

Project name: IMletDemo

Application Descriptor

Name to be used for the jad file, generated during the "Create Package" process:

(●) Use project name as filename

( ) Use custom jad file name

Jad filename: IMletDemo.jad

Contents

(●) Create new project in workspace

( ) Create project from existing source

Directory: C:\Documents and Settings\ewood.ST-USERS\workspace\IMletDe

Configurations

You can add more configurations here:

active	Configuration	
✓	IMPNGDevice1	

Working Sets

☐ Add project to working sets

Working Sets:

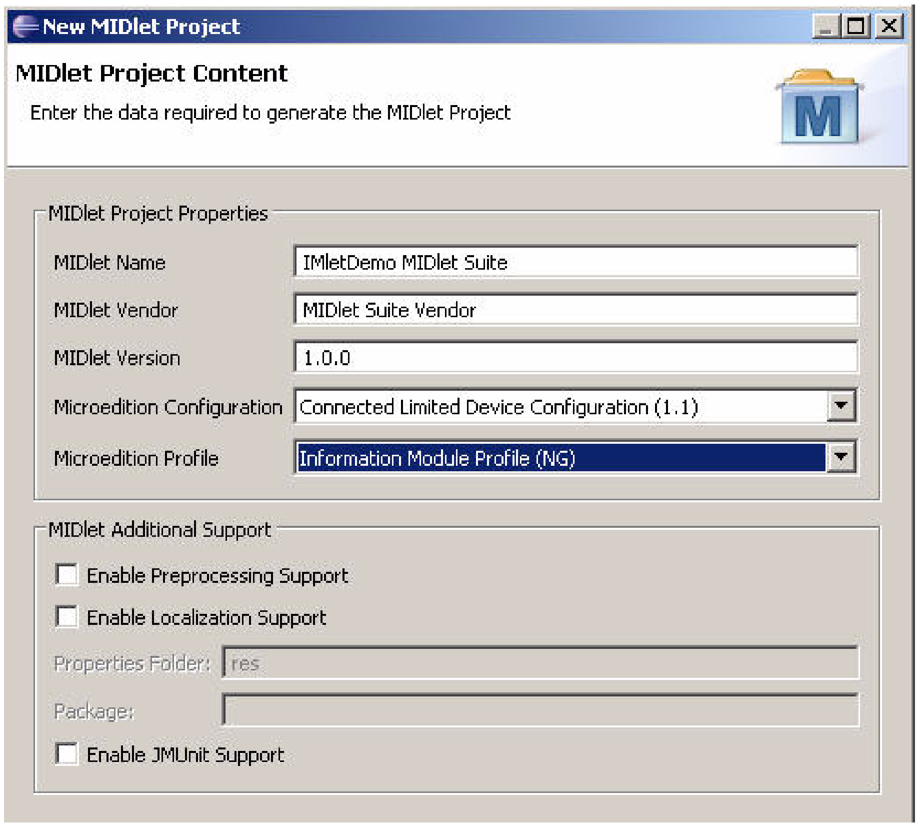
6. Press Next. This displays the MIDlet Project Content dialog box, as shown in Figure 4–13.

   The MIDlet Project Content dialog box allows you to create an Application Descriptor for your new project.

7. In the MIDlet Project Content dialog box, set the following fields as shown here:

   - **Microedition Configuration**: Connected Limited Device Configuration (1.1)
   - **Microedition Profile**: Information Module Profile (NG)

Figure 4–13 Creating A New Application Descriptor


New MIDlet Project

MIDlet Project Content

Enter the data required to generate the MIDlet Project

MIDlet Project Properties

MIDlet Name	IMletDemo MIDlet Suite
MIDlet Vendor	MIDlet Suite Vendor
MIDlet Version	1.0.0
Microedition Configuration	Connected Limited Device Configuration (1.1)
Microedition Profile	Information Module Profile (NG)

MIDlet Additional Support

- [ ] Enable Preprocessing Support
- [ ] Enable Localization Support

Properties Folder: res

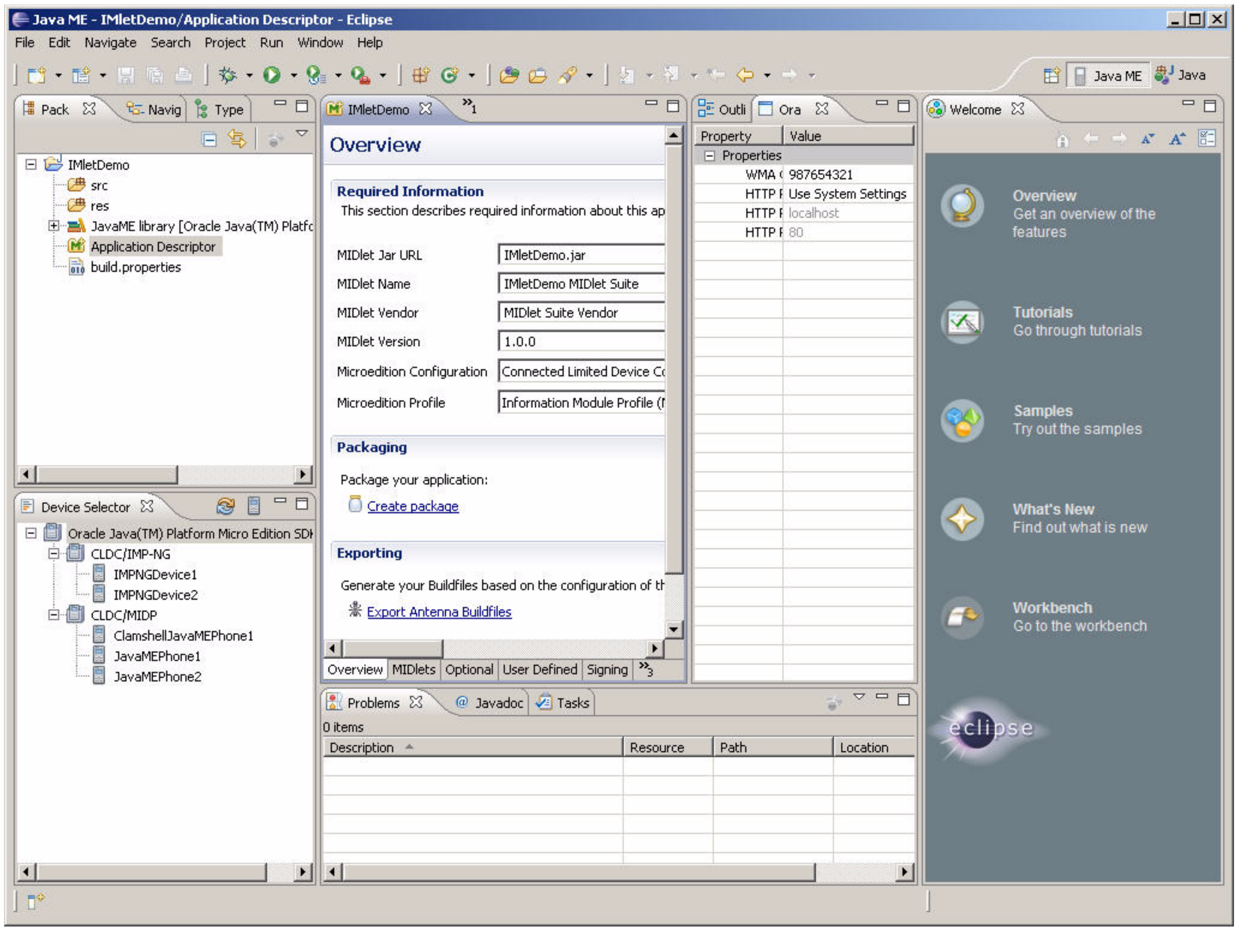
Package:

- [ ] Enable JMUnit Support

8. Press Finish to create the new `IMletDemo` project.

   Your new Eclipse project is ready to be run, as shown in Figure 4–14.

Figure 4–14 The New IMletDemo Project Overview



Overview

Required Information

This section describes required information about this ap

MIDlet Jar URL	IMletDemo.jar
MIDlet Name	IMletDemo MIDlet Suite
MIDlet Vendor	MIDlet Suite Vendor
MIDlet Version	1.0.0
Microedition Configuration	Connected Limited Device Co
Microedition Profile	Information Module Profile (I

Packaging

Package your application:

Create package

Exporting

Generate your Buildfiles based on the configuration of th

Export Antenna Buildfiles

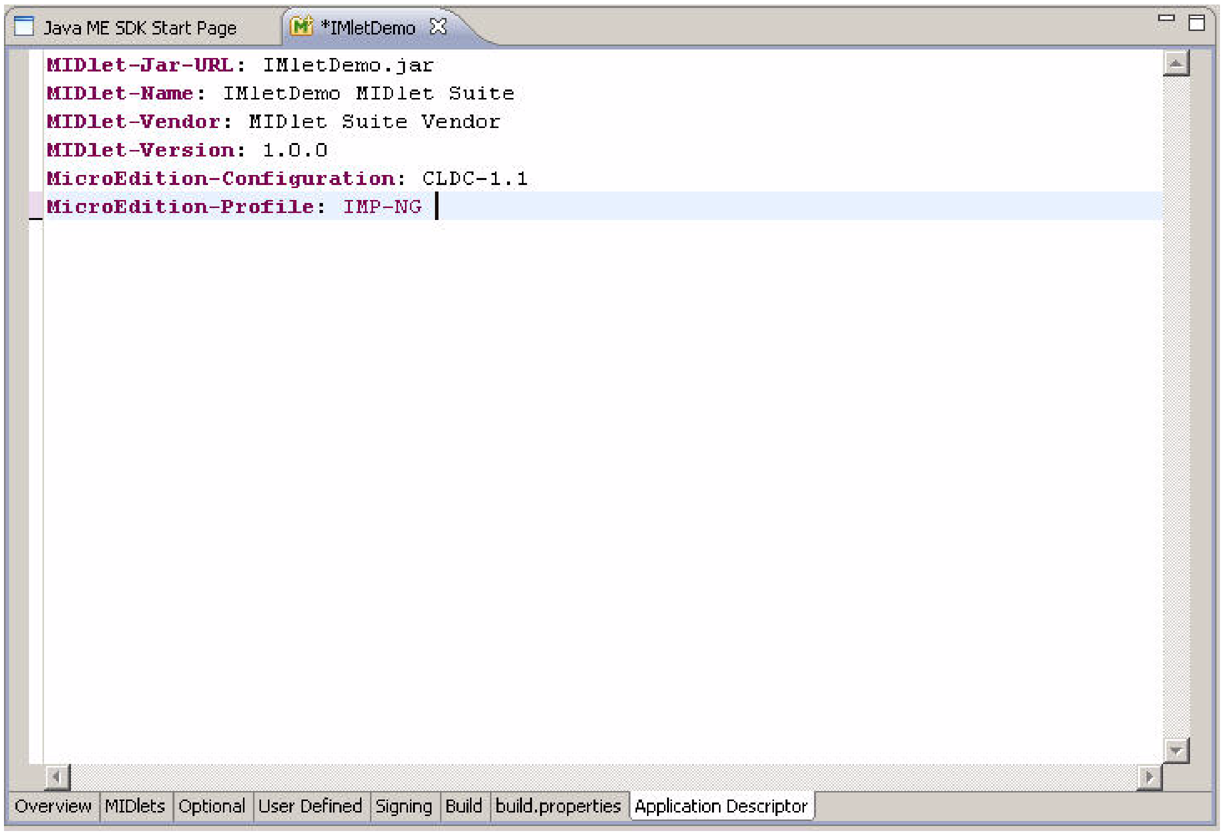
Property	Value
Properties	
WMA (	987654321
HTTP I	Use System Settings
HTTP I	localhost
HTTP I	80

Configuring the Application Descriptor

With the `IMletDemo` project displayed in the Overview window of the Eclipse workscreen, do the following:

1. Click the Application Descriptor tab to display the contents of the Application Descriptor.
2. Change the `MicroEdition-Profile` property to `IMP-NG`, as shown in Figure 4–15.

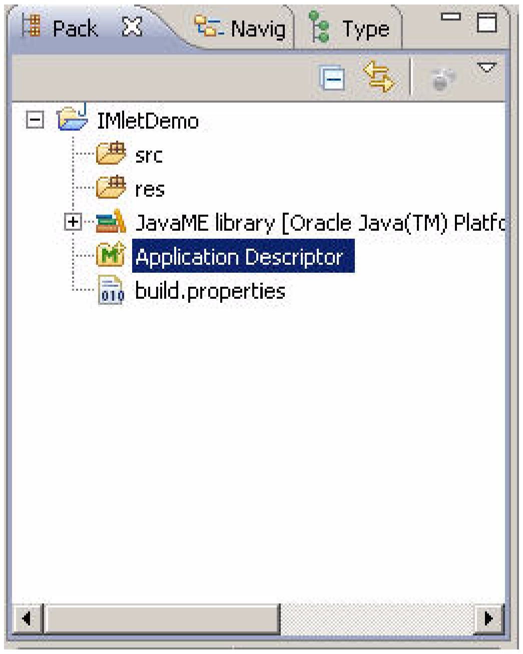
Figure 4–15 Configuring the Application Descriptor



```
MIDlet-Jar-URL: IMletDemo.jar
MIDlet-Name: IMletDemo MIDlet Suite
MIDlet-Vendor: MIDlet Suite Vendor
MIDlet-Version: 1.0.0
MicroEdition-Configuration: CLDC-1.1
MicroEdition-Profile: IMP-NG
```

3. Click the Application Descriptor in the Pack window, as shown in Figure 4–16.
4. Press Control-S to save your changes.

Figure 4–16 Saving Changes to the Application Descriptor

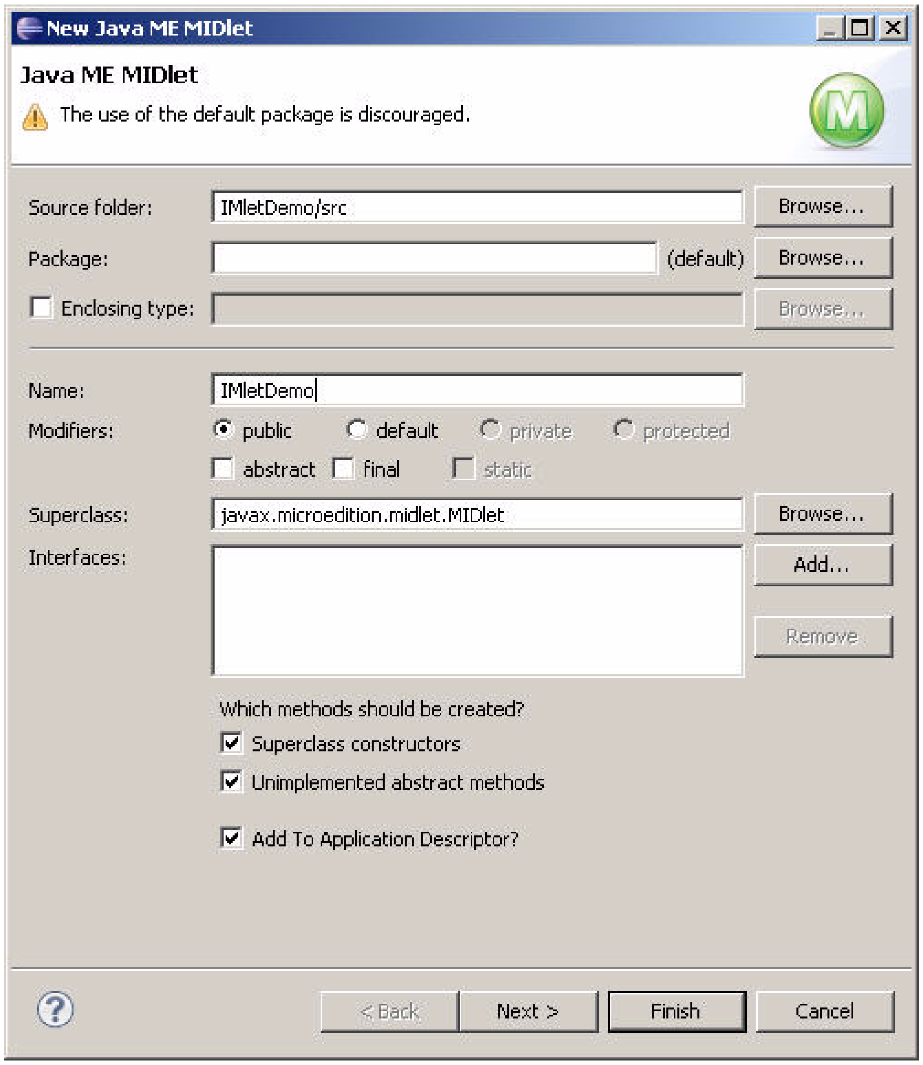


Adding a Test MIDlet to the Project

Adding a Test MIDlet to Eclipse allows to you set up sample code, from which you can easily launch your own project. To add a test MIDlet to Eclipse:

1. Select File > New > Java ME MIDlet.
2. Enter the name and optional package of the MIDlet, as shown in Figure 4–17.

Figure 4–17 Creating a New Java MIDlet



Java ME MIDlet

The use of the default package is discouraged.

Source folder: IMletDemo/src

Package: (default)

[ ] Enclosing type:

Name: IMletDemo

Modifiers: [✓ public] default private protected

[ ] abstract [ ] final [ ] static

Superclass: javax.microedition.midlet.MIDlet

Interfaces:

Which methods should be created?

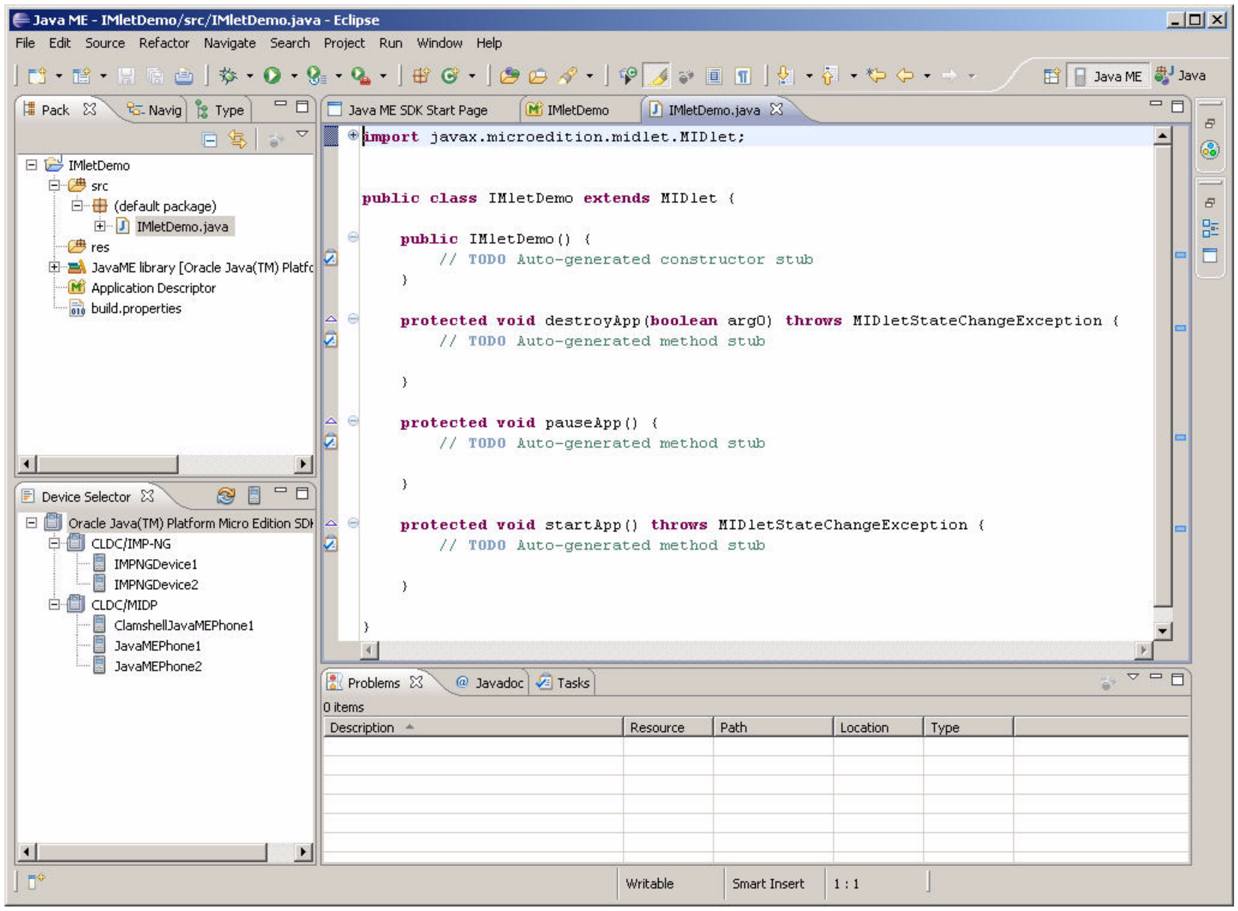
[✓] Superclass constructors

[✓] Unimplemented abstract methods

[✓] Add To Application Descriptor?

3. Press Finish. This creates the `IMletDemo` test MIDlet and displays Eclipse default code, as shown in Figure 4–18.

Figure 4–18 Displaying the IMletDemo Test MIDlet Sample Code



```java
import javax.microedition.midlet.MIDlet;

public class IMletDemo extends MIDlet {

    public IMletDemo() {
        // TODO Auto-generated constructor stub
    }

    protected void destroyApp(boolean arg0) throws MIDletStateChangeException {
        // TODO Auto-generated method stub

    }

    protected void pauseApp() {
        // TODO Auto-generated method stub

    }

    protected void startApp() throws MIDletStateChangeException {
        // TODO Auto-generated method stub

    }

}
```

Adding Sample Code to a New Project

You have just created a new Oracle Java ME Embedded project as a test MIDlet in Eclipse. At this point, you can use the skeleton code provided in "Creating a Sample IMlet File" to start building an embedded project.

1. Select and delete the default code displayed in your new Eclipse project window.
2. Copy and paste the sample code you created in "Creating a Sample IMlet File" into the Eclipse project window.
3. Select `IMletDemo.java` in the Package Explorer window, as shown in Figure 4–19.
4. Press Control-S to save the new `IMletDemo.java` code in the Eclipse workscreen.

Figure 4–19 Saving IMletDemo Java Code

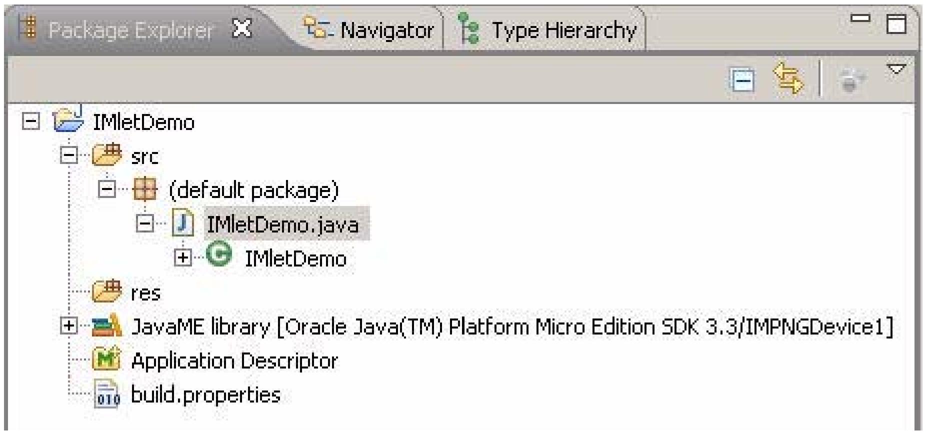
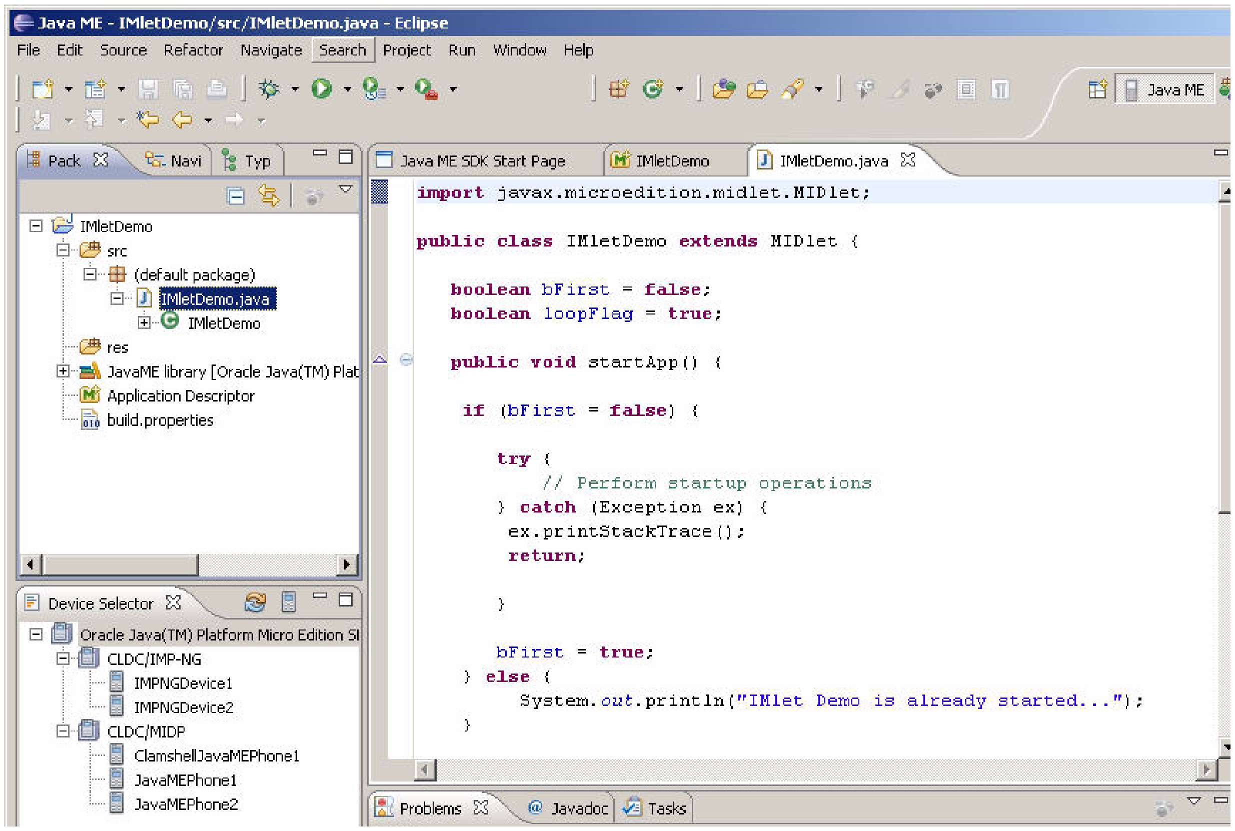


Figure 4–20 shows the Eclipse IDE window with the new project sample code incorporated.

Figure 4–20 The IMletDemo Sample Project Code



```
import javax.microedition.midlet.MIDlet;

public class IMletDemo extends MIDlet {

   boolean bFirst = false;
   boolean loopFlag = true;

   public void startApp() {

    if (bFirst = false) {

       try {
          // Perform startup operations
       } catch (Exception ex) {
        ex.printStackTrace();
        return;

       }

       bFirst = true;
    } else {
         System.out.println("IMlet Demo is already started...");
    }
```

Running the IMletDemo Sample Project

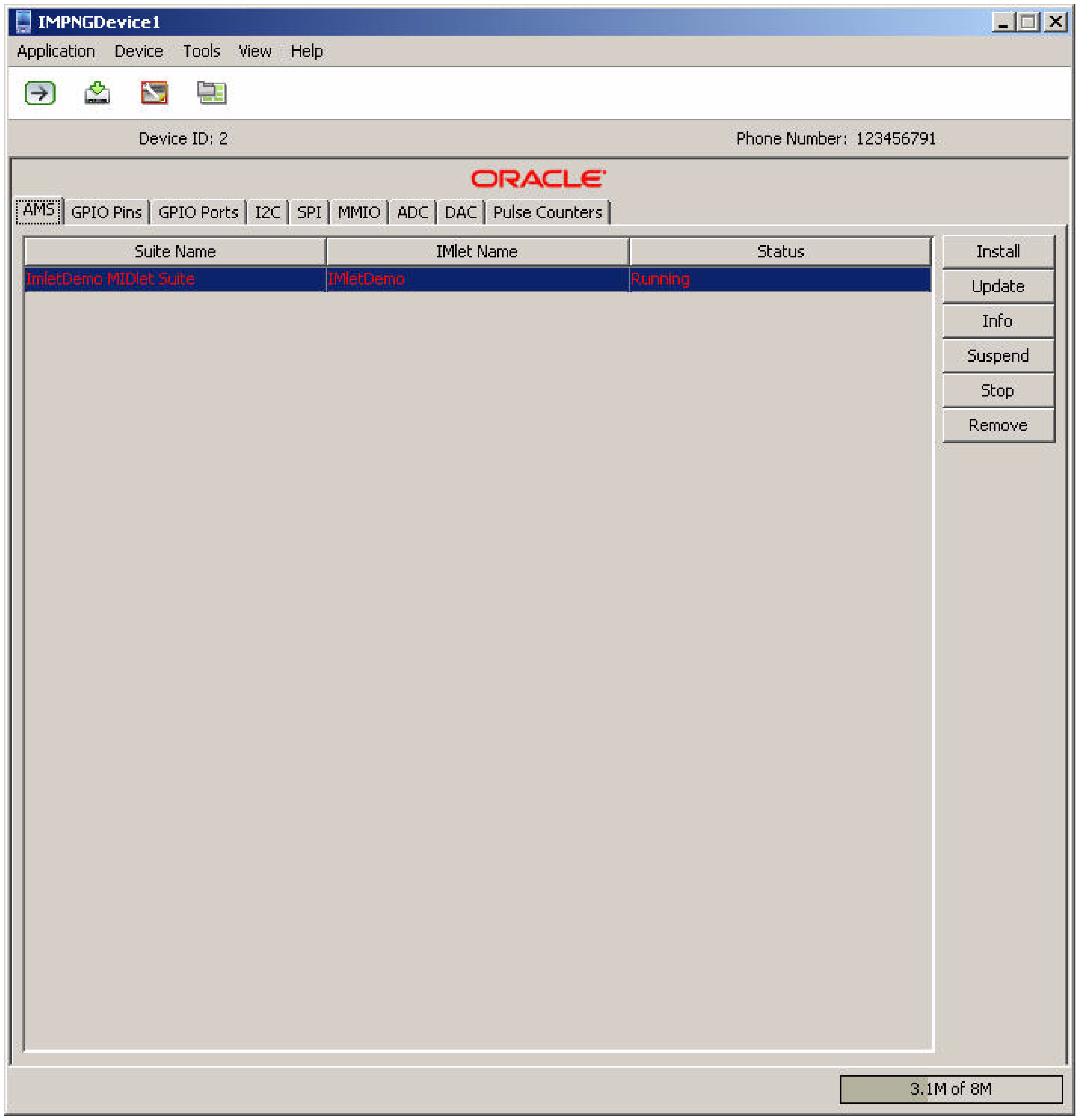
To run the `IMletDemo` sample code:

1. With `IMletDemo.java` selected, as shown in Figure 4–19, press Control-F11.

   This runs the sample code and launches the IMPNGDevice1 emulator, with the `IMletDemo` MIDlet Suite running, as shown in Figure 4–21.

For more information on working with the Oracle Java ME SDK emulator, see Chapter 3.

Figure 4–21 The IMPNGDevice1 Emulator with the IMLetDemo Running



Suite Name	IMlet Name	Status
ImletDemo MIDlet Suite	IMletDemo	Running

A Running Sample Applications

This section describes how to use demos in the Oracle Java ME SDK platform. Because IMP-NG is headless, you can observe the running status of an application in the following ways:

- In the emulator's External Events Generator
- In the Windows command line console

Sample applications are available in two ways:

- As part of the Oracle Java ME SDK Plugins for each IDE (that is, for NetBeans and Eclipse)
- By downloading from the Oracle Java ME SDK Update Center

Note: Before using the Oracle Java ME SDK demo applications, see "Installation and Runtime Security Guidelines" in Appendix C. Some demos use network access and open ports, and do not include protections against malicious intrusion. If you choose to run the sample projects, you should ensure your environment is secure.

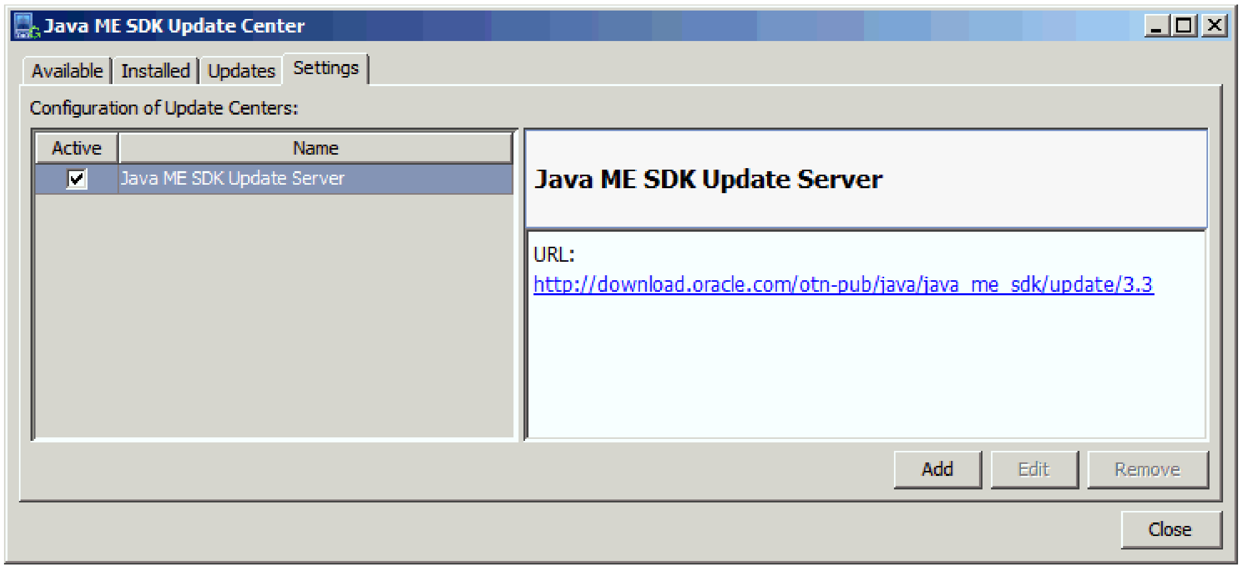
Using the Oracle Java ME SDK Update Center to Get Sample Applications

The Oracle Java ME SDK Update Center is a part of the Oracle Java ME SDK product distribution and is used only for updating Oracle Java ME SDK components.

To run the Oracle Java ME SDK Update Center:

1. Go to the location where you have installed the Oracle Java ME SDK distribution. For example, `C:\Java_ME_platform_SDK_3.3`.
2. Change to the `\bin` directory and double click on `update-center.exe` to launch the Oracle Java ME SDK Update Center, as shown in Figure A–1.

Figure A–1 The Java ME SDK Update Center Window



Configuration of Update Centers:

Active	Name
✓	Java ME SDK Update Server

Java ME SDK Update Server

URL:
http://download.oracle.com/otn-pub/java/java_me_sdk/update/3.3

3. Click on the Available and Updates tabs to find Samples (or to find other Oracle Java ME SDK software updates, such as new plugins.)
4. Select the Samples or updates you want to install and click Add.

Using Sample Applications in the Emulator

The Oracle Java ME SDK platform comes with a number of sample applications. The following four demos best illustrate embedded features when run in Oracle Java ME SDK:

- GPIODemo
- I2CDemo
- NetworkDemoIMPNG
- PDAPDemoIMPNG

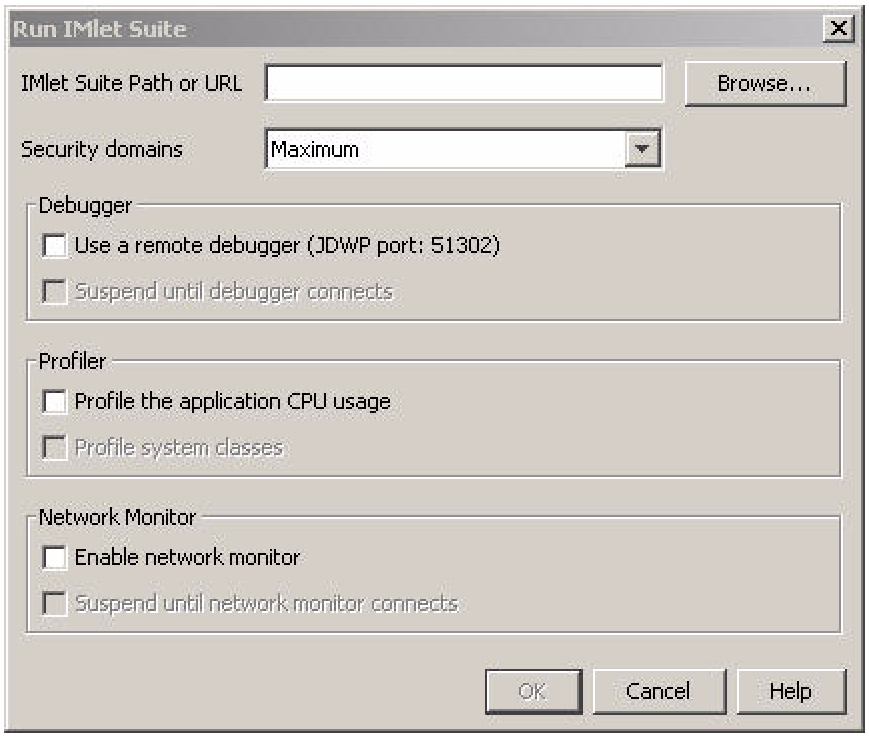
Note: With the exception of the I2CDemo, the sample projects provided with the Oracle Java ME SDK release can be run on the emulator or on a real device.

Starting Sample Applications

There are two ways to start sample applications in Oracle Java ME SDK:

- From the IMPNGDevice1 emulator:
  1. Select Application --> Run IMlet Suite
  2. Fill in the Run IMlet Suite dialog box and provide the Path or URL for the IMlet Suite, as shown in Figure A–2.

Figure A–2 The Run IMlet Suite Dialog Box



Run IMlet Suite

IMlet Suite Path or URL

Security domains: Maximum

Debugger
- [ ] Use a remote debugger (JDWP port: 51302)
- [ ] Suspend until debugger connects

Profiler
- [ ] Profile the application CPU usage
- [ ] Profile system classes

Network Monitor
- [ ] Enable network monitor
- [ ] Suspend until network monitor connects

- From the Windows command line. Enter the following:

  emulator.exe -Xdevice:IMPNGDevice1 -Xdescriptor:C:\Java_ME_platform_SDK_3.3\apps\sample\<*sample_demo_folder*>\<*sample_demo_name*>**.jad**

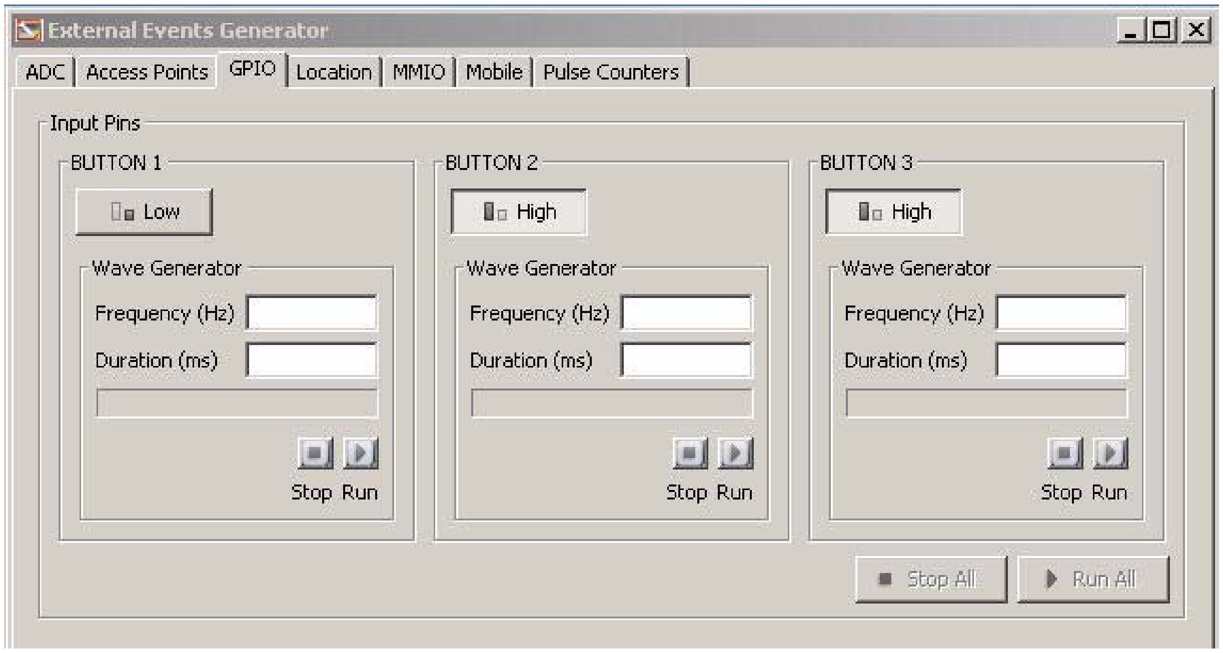
For the four demos described in this chapter, all but GPIODemo can be started using both of the ways shown here.

The GPIODemo Sample Application

The GPIODemo can be run on the Oracle Java ME SDK emulator, IMPNGDevice1:

1. Launch the External Events Generator by selecting Tools > External Events Generator or clicking on the External Events Generator icon.
2. Click the GPIO tab. This displays the GPIO screen, as shown in Figure A–3.

Figure A–3 The External Events Generator GPIO Tab

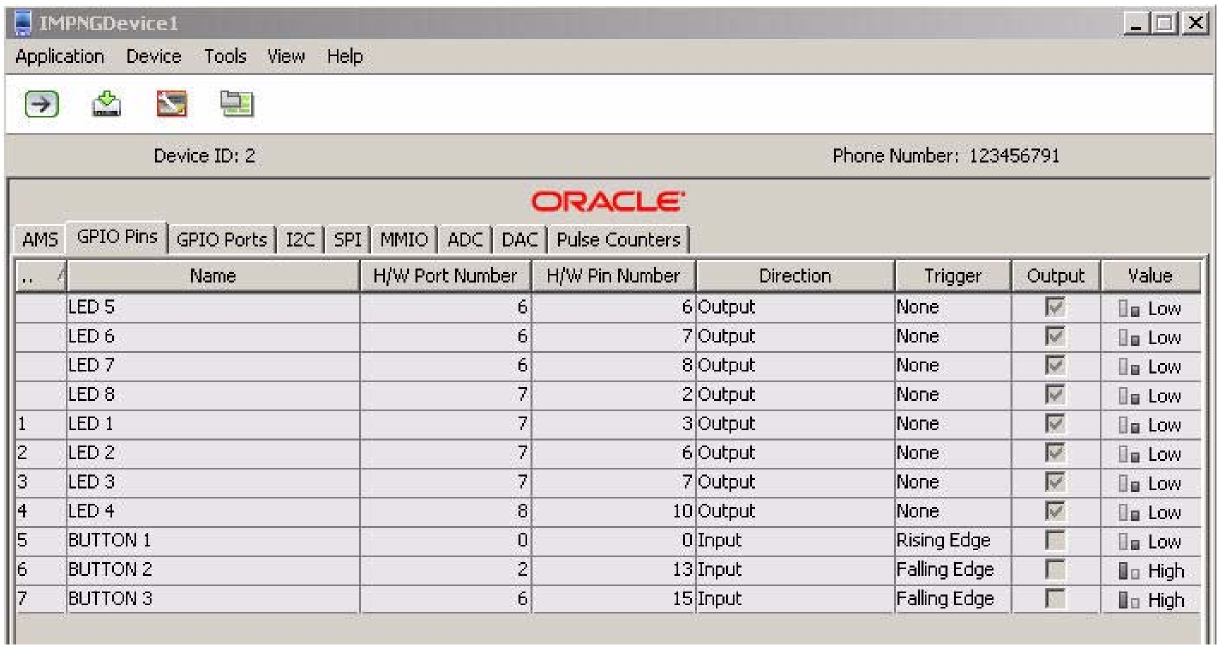


The External Events Generator GPIO tab approximates actions on a device. Each button is a toggle that turns an LED on and off, changing the value of the corresponding Pin from High to Low, or Low to High.

When a button is clicked, information is written to the IMPNGDevice1 GPIO Pins tab. Changes in the value of a Button can be seen in the Output and Value columns for Buttons 1, 2, and 3, as shown in Figure A–4.

3. Click each button to change the value in the Output column of the GPIO Pins tab, in the IMPNGDevice1.

Figure A–4 The GPIO Pins Tab



Device ID: 2

Phone Number: 123456791

..	Name	H/W Port Number	H/W Pin Number	Direction	Trigger	Output	Value
	LED 5	6	6	Output	None	✓	Low
	LED 6	6	7	Output	None	✓	Low
	LED 7	6	8	Output	None	✓	Low
	LED 8	7	2	Output	None	✓	Low
1	LED 1	7	3	Output	None	✓	Low
2	LED 2	7	6	Output	None	✓	Low
3	LED 3	7	7	Output	None	✓	Low
4	LED 4	8	10	Output	None	✓	Low
5	BUTTON 1	0	0	Input	Rising Edge	☐	Low
6	BUTTON 2	2	13	Input	Falling Edge	☐	High
7	BUTTON 3	6	15	Input	Falling Edge	☐	High

Notice the following about Buttons 1, 2, and 3 in the IMPNGDevice1 GPIO Pins tab:

- Button 1 corresponds to Hardware Port Number `0` and Hardware Pin Number `0`. Button 1 has a Rising Edge Trigger.
- Button 2 corresponds to Hardware Port Number `2` and Hardware Pin Number `13`. Button 2 has a Falling Edge Trigger.
- Button 3 corresponds to Hardware Port Number `6` and Hardware Pin Number `15`. Button 3 has a Falling Edge Trigger.

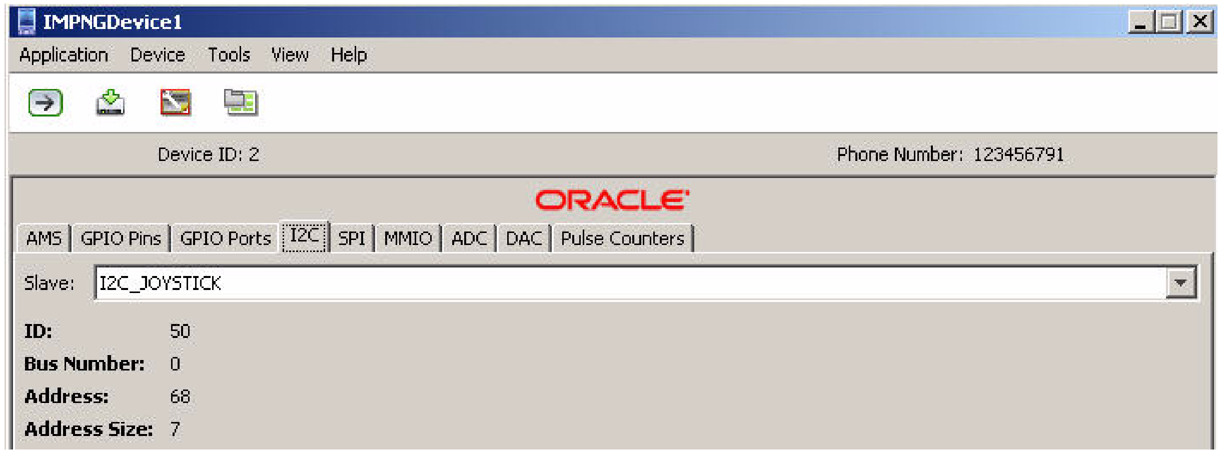
  For more information on Rising and Falling Edge Triggers, see "The Pulse Counter Tab."

The I2CDemo Sample Application

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

To launch the I2C demo, click the I2C tab in IMPNGDevice1.

Figure A–5 The I2C Tab



Device ID: 2

Phone Number: 123456791

Slave: I2C_JOYSTICK

ID: 50
Bus Number: 0
Address: 68
Address Size: 7

This demo acquires a slave named I2C_JOYSTICK, writes data to the slave, and retrieves it. The demo is successful if the Sent Data and Received Data match.

The NetworkDemoIMPNG Sample Application

The NetworkDemoIMPNG 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.

1. Create two instance projects of the NetworkDemoIMPNG sample project.
2. Right click the first project and select Properties.
3. In the Platform category choose the device `IMPNGDevice1`. In the Application Description category set the value of the following property:

   ```
   Oracle-Demo-Network-Mode:Server
   ```
4. Click OK.
5. Launch the first project. It opens on the emulator IMPNGDevice1 and waits for a connection. You should see messages like the following:

6. Right click the second project and select Properties.
7. In the Platform category choose the device `IMPNGDevice2`. In the Application Description category set the value of the following property:

   ```
   Oracle-Demo-Network-Mode:Client
   ```
8. Click OK.
9. Launch the second project. It opens on the emulator IMPNGDevice2.
10. The client attempts to connect to the server. If successful, you see the following in the output tab of the first project (the server):

    ```
    Waiting for connection on port 500
    [AMS-TRACE] MIDlet:NetworkDemoIMPNG status=2
    [AMS-TRACE] MIDlet:NetworkDemoIMPNG status=1
    Connection accepted
    Message received - Client messages
    ```

    The output of the second project (the client) shows the following:

    ```
    [AMS-TRACE] MIDlet:NetworkDemoIMPNG status=2
    [AMS-TRACE] MIDlet:NetworkDemoIMPNG status=1
    Connected to server localhost on port 500
    Message received - Server string
    ```

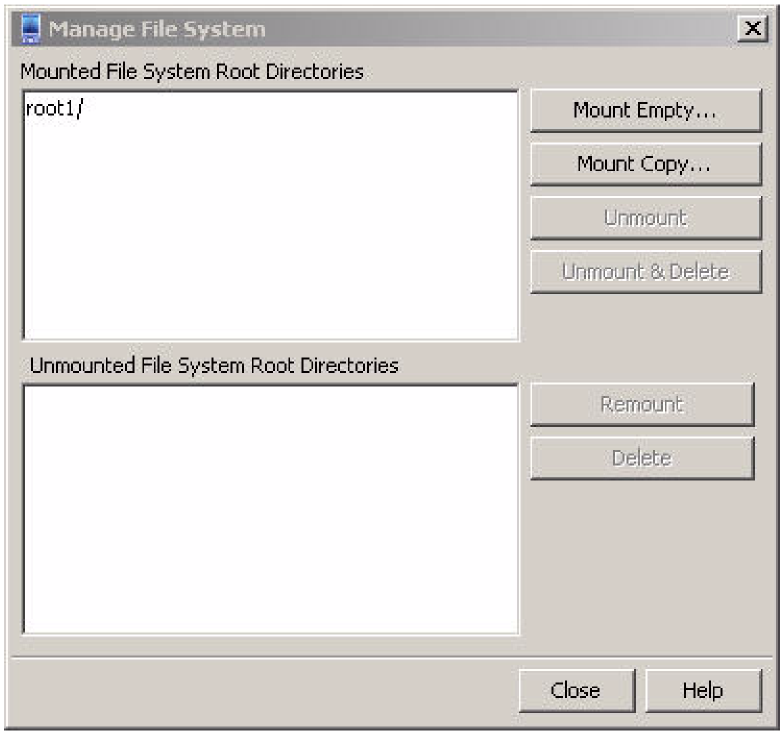
The PDAPDemoIMPNG Sample Application

Follow these steps to run the PDADemoIMPNG demo on the IMPNGDevice1 emulator:

1. Create test files and directories inside the emulator's file system:

   ```
   Documents and Settings\<user>\javame-sdk\<version>\work\IMPNGDevice1\app
   db\filesystem\root1
   ```
2. Right click the project and select Properties. In the Platform category choose the device IMPNGDevice1 and click OK.
3. Launch the project. It runs on IMPNGDevice1.
4. On the emulator menu, select Tools > File Connection to see a list of mounted file systems.

Figure A–6 The Manage File System Window



5. Open a terminal emulator and create a raw connection to `localhost` on port `5001`. For example:

   ```
   telnet localhost 5001
   ```

   Note: The Telnet negotiation mode must be set to Passive. The negotiation mode can be set inside a Telnet client application (for example, PuTTY), by choosing Category > Connection > Telnet > Passive.

   A command line opens where you can browse the emulator's file system. You can use the following commands:

   – `cd` <*dir_name*> - Change directory

   – `ls` - List information about the files for the current directory

   – `new file` <*file_name*> - Create a new file

   – `new dir` <*directory_name*> - Create a new directory

   – `prop` <*file_name*> - Show properties of a file

   – `rm` <*file_name*> - Remove the file

   – `view` <*file_name*> - View a file's content

The Light Tracker Sample Application

The Light Tracker sample application is aimed at showing off funtionality on a embedded device, such as the Keil MCBSTM32F200 reference platform.

In this demo, a certain number of LEDs are turned on and turned off on the reference board, in a sequence that you can control. It makes use of the Device Access API and a GPIO Port to demonstrate its functionality. It requires connection of an ADC channel to an on-board potentiometer.

For more information on the setup of the Light Tracker sample application, see the LightTrackDemo `readme.txt` file, in the location where you have installed the Oracle Java ME SDK sample applications. For example:

```
C:\Documents and Settings\username\Java_ME_SDK\samples\LightTrackDemo
```

For more information on the Keil MCBSTM32F200 platform, see the *Oracle Java ME Embedded Getting Started Guide for the Reference Platform (Keil)*.

The System Controller Sample Application

The System Controller sample application is aimed at showing off funtionality on a embedded device, such as the Keil MCBSTM32F200 reference platform.

The purpose of this demo is to control the lifecycle of IMlets on the reference platform. It makes use of the following functionalities:

- Multitasking Virtual Machine (MVM)
- IMlet auto-start
- Application Management System (AMS) API
- Logging API
- Device Access API
- General Purpose Input/Output (GPIO)
- Watchdog timer

For more information on the setup of the System Controller sample application, see the SystemControllerDemo `readme.txt` file, in the location where you have installed the Oracle Java ME SDK sample applications. For example:

```
C:\Documents and Settings\username\Java_ME_
SDK\samples\SystemControllerDemo
```

For more information on the Keil MCBSTM32F200 platform, see the *Oracle Java ME Embedded Getting Started Guide for the Reference Platform (Keil)*.

B

Using the Command Line Emulator

The Oracle Java ME SDK 3.3 emulator can be started from a Windows command line. Once started, it runs and behaves the same as it does when started from the NetBeans or Eclipse IDE.

Starting the emulator from the Windows command line allows you a number of emulator options. For more information, see "Useful Command Line Emulator Commands".

Finding the Oracle Java ME SDK Emulator

You can find the Oracle Java ME SDK command line emulator within the `bin` directory of the Oracle Java ME SDK 3.3 installation.

For example, if the Oracle Java ME SDK 3.3 is installed in `C:\Java_ME_platform_SDK_3.3`, then the emulator would be located at: `C:\Java_ME_platform_SDK_3.3\bin\emulator.exe`

Using the Oracle Java ME SDK Emulator

To start the emulator from the Windows command line:

1. Open the Windows command line from the Start Menu, using Start > Run.
2. Change to the `bin` directory of the Oracle Java ME SDK 3.3 distribution:

   C:\>**cd Java_ME_platform_SDK_3.3\bin**
3. Enter the following command, as shown here:

   `emulator.exe -Xdevice:IMPNGDevice1 -Xdescriptor:`*location_of_jad_file*

   For example:

   emulator.exe -Xdevice:IMPNGDevice1 -Xdescriptor:C:\Java_ME_platform_SDK_3.3\apps\sample\sample_imlet.jad

---

Note: You can run the `emulator` command without the `.exe` extension. It works both ways, with the extension and without

---

Useful Command Line Emulator Commands

This section provides a list of ME SDK emulator commands you may find useful, which can be entered on the Windows command line.

Note: For a complete list of ME SDK 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 a 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:

  EmulatorDir>`emulator -Xjam:run=[storage_name | storage_number]`

  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`

C

Installation and Runtime Security Guidelines

The Oracle Java ME SDK requires an execution model that makes certain networked resources available for emulator execution. These required resources might include - but are not limited to - a variety of communication capabilities between Oracle Java ME SDK components.

Note: the Oracle Java ME SDK installation and runtime system is fundamentally a developer system. It is not designed to guard against any malicious attacks from outside intruders.

During execution, the Oracle Java ME SDK architecture can present an insecure operating environment to the platform's installation file system, as well as its runtime environment. For this reason, it is critically important to observe the precautions outlined in these guidelines when installing and running the Oracle Java ME SDK.

Maintaining Optimum Network Security

To maintain optimum network security, Oracle Java ME SDK can be installed and run in a "closed" network operating environment, where the Oracle Java ME SDK system is not connected directly to the Internet. Or, it can be connected to a secure company Intranet environment that can reduce unwanted exposure to malicious intrusion.

An example of an Oracle Java ME SDK requirement for an Internet connection is when wireless functionality requires a connection to the Internet to support communications with the wireless network infrastructure that is part of an Oracle Java ME SDK application execution process. Whether or not an Internet connection is required depends on the particular application running on Oracle Java ME SDK. For example, some applications can use an HTTP connection.

In any case, if the Oracle Java ME SDK is open to any network access you must observe the following precautions to protect valuable resources from malicious intrusion:

- Installing the Oracle Java ME SDK Demos plugin is optional. Some sample projects use network access and open ports. Because the sample code does not include protection against malicious intrusion, you must ensure your environment is secure if you choose to install and run the sample projects.
- Install the Oracle Java ME SDK behind a secure firewall that strictly limits unauthorized network access to the Oracle Java ME SDK file system and services. Limit access privileges to those that are required for Oracle Java ME SDK usage while allowing all the bidirectional local network communications that are necessary for Oracle Java ME SDK functionality. The firewall configuration must

support these requirements to run the Oracle Java ME SDK while also addressing them from a security standpoint.

- Follow the principle of "least privilege" by assigning the minimum set of system access permissions required for installation and execution of the Oracle Java ME SDK.
- Do not store any data sensitive information on the same file system that is hosting the Oracle Java ME SDK.
- To maintain the maximum level of security, make sure the operating system patches are up-to-date on the Oracle Java ME SDK host machine.

Glossary

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, or for different bearer types that may be available on a device, such as Wi-Fi or Bluetooth.

ADC

Analog-to-Digital Converter. A hardware device that converts analog signals (time and amplitude) into a stream of binary numbers that can be processed by a digital device.

AMS

Application Management System. The system functionality that completes tasks such as installing applications, updating applications, and managing applications between foreground and background.

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.

ARM

Advanced RISC Machine. A family of computer processors using reduced instruction set (RISC) CPU technology, developed by ARM Holdings. ARM is a licensable instruction set architecture (ISA) and is used in the majority of embedded platforms.

AT commands

A set of commands developed to facilitate modem communications, such as dialing, hanging up, and changing the parameters of a connection. Also known as the Hayes command set, AT means *attention.*

AXF

ARM Executable Format. An ARM executable image generated by ARM tools.

BIP

Bearer Independent Protocol. Allows an application on a SIM Card to establish a data channel with a terminal, and through the terminal, to a remote server on the network.

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.

DAC

Digital-to-Analog Converter. A hardware device that converts a stream of binary numbers into an analog signal (time and amplitude), such as audio playback.

ETSI

European Telecommunications Standards Institute. An independent, non-profit group responsible for the standardization of information and communication technologies within Europe. Although based in Europe, it carries worldwide influence in the telecommunications industry.

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.

GPIO

General Purpose Input/Output. Unassigned pins on an embedded platform that can be assigned or configured as needed by a developer.

GPIO Port

A group of GPIO pins (typically 8 pins) arranged in a group and treated as a single port.

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

Secure HyperText Transfer Protocol. A protocol for transferring encrypted hypertext data using Secure Socket Layer (SSL) technology.

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 IMP-NG specification (JSR 228) is a subset of MIDP 2.0 that leverages many of the APIs of MIDP 2.0, including the latest security and networking+, but does not include graphics and user interface 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 cannot 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.

IMlet Suite

A way of packaging one or more IMlets for easy distribution and use. Similar to a MIDlet suite, but for smaller applications running in an embedded environment.

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.

I2C

Inter-Integrated Circuit. A multi-master, serial computer bus used to attach low-speed peripherals to an embedded platform

ISA

Instruction Set Architecture. The part of a computer’s architecture related to programming, including data type, addressing modes, interrupt and exception handling, I/O, and memory architecture, and native commands. Reduced instruction set computing (RISC) is one kind of instruction set architecture.

JAD file

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

JCP

Java Community Process. The global standards body guiding the development of the Java programming language.

JDTS

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.

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, set-top boxes, and embedded devices. More specifically, the Java ME platform consists of a configuration (such as CLDC) and a profile (such as MIDP or IMP-NG) tailored to a specific class of device.

JSR

Java Specification Request. 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 a small, limited memory device. 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.

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.

MVM

Multiple Virtual Machines. A software mode that can run more than one MIDlet or IMlet at a time.

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.

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 an embedded or 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 an embedded or mobile device over a network.

Pulse Counter

A hardware or software component that counts electronic pulses, or events, on a digital input line, for example, a GPIO pin.

Push Registry

The list of inbound connections, across which entities can push data. 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.

RISC

Reduced Instruction Set Computing. A CPU design based on simplified instruction sets that provide higher performance and faster execution of individual instructions. The ARM architecture is based on RISC design principles.

RL-ARM

Real-Time Library. 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 an IMlet or MIDlet to persistently store information and retrieve it later. MIDlets can also use the RMS to share data.

RTOS

Real-Time Operating System. An operating system designed to serve real-time application requests. It uses multi-tasking, an advanced scheduling algorithm, and minimal latency to prioritize and process data.

RTSP

Real Time Streaming Protocol. A network control protocol designed to control streaming media servers and media sessions.

RTX

The real-time operating system used on the Keil MCBSTM32F200 embedded platform.

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.

SD card

Secure Digital cards. A non-volatile memory card format for use in portable devices, such as mobile phones and digital cameras, and embedded systems. SD cards come in three different sizes, with several storage capacities and speeds.

SIM

Subscriber Identity 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 and embedded devices.

Slave Mode

Describes the relationship between a master and one or more devices in a Serial Peripheral Interface (SPI) bus arrangement. Data transmission in an SPI bus is initiated by the master device and received by one or more slave devices, which cannot initiate data transmissions on their own.

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 embedded board is powered down), 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.

SPI

Serial Peripheral Interface. A synchronous bus commonly used in embedded systems that allows full-duplex communication between a master device and one or more slave devices.

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 terminal (mobile or embedded device) passed 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.

UART

Universal Asynchronous Receiver/Transmitter. A piece of computer hardware that translates data between serial and parallel formats. It is used to facilitate communication between different kinds of peripheral devices, input/output streams, and embedded systems, to ensure universal communication between devices.

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.

USB

Universal Serial Bus. An industry standard that defines the cables, connectors, and protocols used in a bus for connection, communication, and power supply between computers and electronic devices, such as embedded platforms and mobile phones.

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. An application framework for small devices, which leverages other technologies, such as Wireless Application Protocol (WAP).

WAP

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

Watchdog Timer

A dedicated piece of hardware or software that watches an embedded system for a fault condition by continually polling for a response. If the system goes offline and no response is received, the watchdog timer initiates a reboot procedure or takes other steps to return the system to a running state.

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