Technical Manual

Motorola SDK for the J2ME™ Platform

Developer Edition

Version 5.2.1

MOTOROLA
intelligence everywhere™

Table of Contents

Introduction

The Java™ 2 Platform, Micro Edition (J2ME) is a small footprint Java product designed by Sun® Microsystems for resource-limited devices such as pagers and cell phones. J2ME platform consists of a Java virtual machine, along with associated Java class libraries to provide a set of core and enhanced services.

The Motorola Software Development Kit (SDK) Components for the J2ME platform provide tools used for developing and testing programs written in the Java programming language. The configuration editor helps you define your device, and the emulator simulates the J2ME platform of your device. Together these tools allow you to develop and debug your J2ME programs in emulation before downloading them to a target device.

How This Guide is Organized

Chapter 2 - Installation: Shows you how to install the Motorola SDK Components as well as Sun Microsystems' Java 2 SDK, Standard Edition, version 1.4 or later.

Chapter 3 - Tutorial: Developing an Application: Teaches you how to develop a J2ME application by walking you through the basic steps of writing, compiling and running an application.

Chapter 4 - Using the Motorola SDK Components: Describes how the components work and shows you how to use them. These components include:

- Working with Configurations: Provides an overview on creating new target device configurations
- The Configuration Editor: A tool for creating and modifying target device configurations
- Working with the Emulator: A tool for testing J2ME programs on the desktop by emulating a target device

Chapter 5 - Using the Launchpad Application: Assists you in the process of MIDlet application development.

Appendix A - Using CodeWarrior Wireless Studio: Shows you how to install and use CodeWarrior Wireless Studio, version 7.0 for developing J2ME applications.

Appendix B - Sample MIDlets: Provides descriptions of the sample applications provided with the Motorola SDK Components for the J2ME platform.

Appendix C - Bytecode Verifier: Introduces the process of bytecode verification.

Appendix D - Using Foreign Fonts in Motorola Emulators: Provides information on using UNICODE fonts with J2ME and the Motorola SDK.

Appendix E - Open Classes: Provides information about the APIs available in this version of Motorola SDK.

Appendix F - Unified Emulator Interface: Future implementation of this feature in Motorola SDK.

Motorola SDK Component Changes

The following changes should be noted since the previous release of this SDK:

- New phone integrated: ROKR E1.

Guide Conventions

This guide uses the following type conventions:

Convention	**Purpose**
`variable`	Words appearing in italics and Courier font are variables that you must replace with appropriate values, as in filename.
`computer`	Words formatted as Courier font represent output, lines of code, pathnames, and filenames.
bold typeface	Words formatted as Bold font represent items that appear on the screen, like a menu command (File), a window title (Target Settings), or a control (OK).
| character	The | character separates each command in a menu hierarchy. For example: File | Open, represents selecting File in the application menu,

	and then selecting the Open command.

Table 1. Convensions Table

Glossary

Here are definitions of common terms used in this manual:

Term	Definition
ADK	Application Development Kit
API	Application Programming Interface
CLDC	Connected, Limited Device Configuration
J2ME	Java™ 2 Micro Edition
JDK	Java Development Kit
JRE	Java Runtime Environment
KVM	Kilobyte Virtual Machine
MIDlet	Mobile Information Device appLet
MIDP	Mobile Information Device Profile
MO	Mobile Originator
PC	Personal Computer
SDK	Software Development Kit
SMS	Short Messaging Service
UI	User interface
URL	Uniform Resource Locator
VM	Virtual Machine

Table 2. Glossary Table

References

The following references provide additional information on the technologies and materials discussed in this manual:

Organization	URL
Metrowerks	www.metrowerks.com
Motorola Developer Program	www.motocoder.com
World Wide Web Consortium	www.w3.org
Sun Microsystems	java.sun.com

Table 3. References Table

For more information on J2ME and on Java programming in general, see the following Web sites and publications:

1. *Sun's Web site on Java 2 Platform, Micro Edition*
   `http://java.sun.com/j2me/`
2. *J2ME Connected, Limited Device Configuration Specification*
   `http://jcp.org/aboutJava/communityprocess/final/jsr030/index.html`
3. *J2ME Mobile Information Device Profile Specification*
   `http://jcp.org/aboutJava/communityprocess/final/jsr037/index.html`
4. *The Java Virtual Machine Specification, Tim Lindholm and Frank Yellin*
   `http://java.sun.com/docs/books/vmspec/index.html`
5. *The Java Language Specification, James Gosling, Bill Joy, and Guy Steele*
   `http://java.sun.com/docs/books/jls/index.html`
6. *The API document for the CLDC specification is in*
   `<MOTOROLA_SDK_HOME>\Docs\cldc1.0\index.html`
7. *The API document for the MIDP specification is in*
   `<MOTOROLA_SDK_HOME>\Docs\midp2\index.html`
8. *The Impronto Simulator Manual is in*
   `<MOTOROLA_SDK_HOME>\Docs\jblend_micro_jabwt_simulator_mj_en_01_00_01.pdf`
9. *The WMA Test Server Manual is in*
   `<MOTOROLA_SDK_HOME>\Docs\jblend_micro_wma20_testtool_mj_en_1_00_00.pdf`

Installation

This chapter shows you how to install the Sun Microsystems' Java™ 2 SDK, Standard Edition, version 1.4.2 and the Motorola SDK Components. Topics include:

- System Requirements
- Installing Sun's Java 2 SDK
- Installing the Motorola SDK Components
- Motorola SDK Directory Structure

System Requirements

To use the Motorola SDK Components, your system must meet these minimum requirements:

Required System Components	**Requirement**
Processor	Pentium™-100 MHz
Memory	64 MB RAM
Operating System	Windows® XP, Windows® 2000.
Hard Disk Space	Approximately 65 MB for the Motorola® SDK Components
Java Environment	Java™ 2, Standard Edition v1.4 (recommended), Java 1.3.1 and later supported

Table 4. System Requirements Table

Installing Sun's Java™ 2 SDK

To use the Motorola SDK Components, Motorola recommends that you use the Java 2 SDK, Standard Edition (J2SE™) version 1.4, although versions 1.3.1 and later are supported.

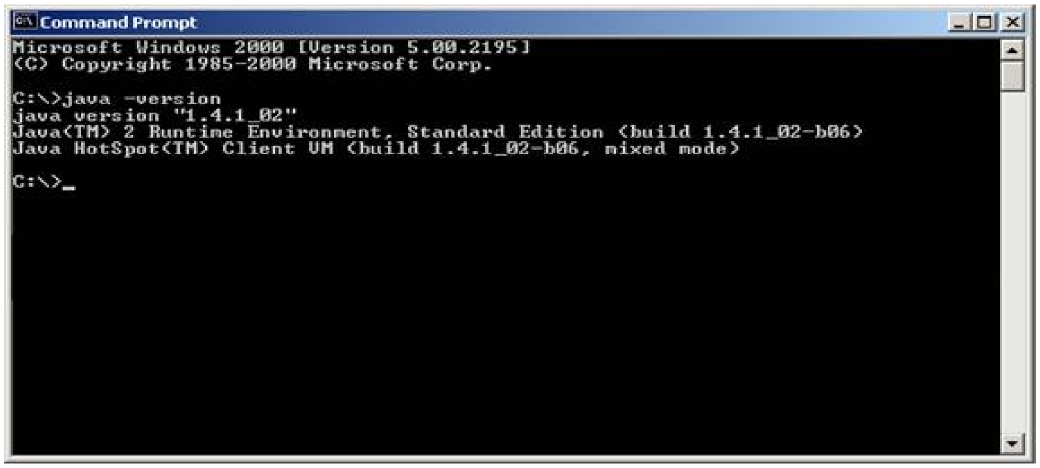
Reading the Java 2, Standard Edition Version Number

To ensure that the correct version of the J2SE platform is installed:

- Select **Start | Settings | Control Panel**
- Open **Add/Remove Programs**

If Java 2 SDK Standard Edition v1.4 or later is listed, click **Cancel**. If a previous version is listed, select it and click **Add/Remove** to uninstall the previous version.

You can also type java -version into a **Command Prompt** window, see Figure 1, to determine which JDK version you have.



```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>java -version
java version "1.4.1_02"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1_02-b06)
Java HotSpot(TM) Client VM (build 1.4.1_02-b06, mixed mode)

C:\>_
```

Figure 1. Using a Command Prompt to determine the Java version

Installing the Java 2, Standard Edition Platform

To install the J2SE platform, follow these steps:

- Download the J2SE SDK from `http://java.sun.com/j2se/1.4/index.jsp`
- Follow Sun's instructions for installation

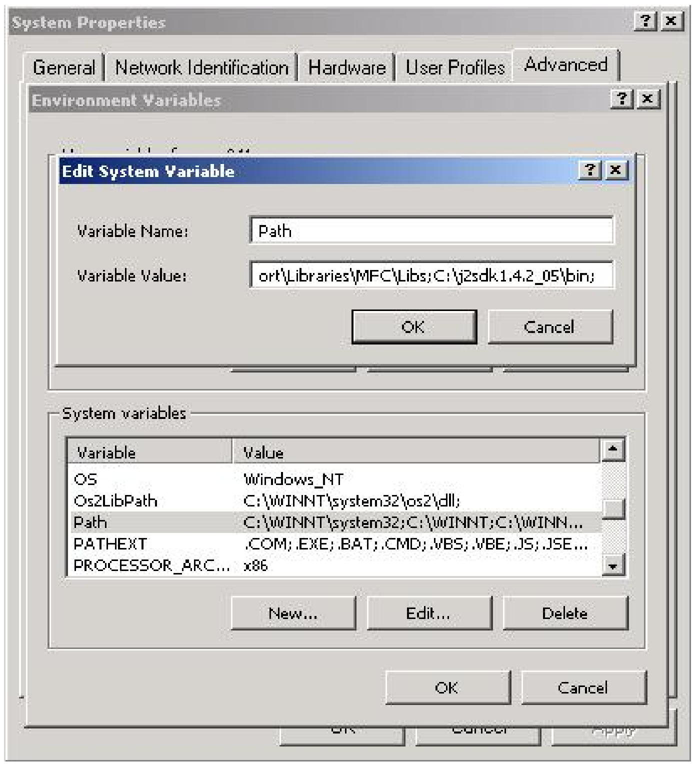
To ensure that the Motorola SDK Components work correctly, the JDK install directory must be in the system path environment variable.

Verifying the Path Environment Variable

To ensure that the Path environment variable is correct, follow these steps:

1. *Select* ***Start | Settings | Control Panel***

2. *Open the **System** control panel*
3. *Select **Advanced** tab and click in **Environment Variables***
4. *Select **Path** under **System Variables**, see Figure 2.*
5. *Look for `c:\j2sdk1.4.2\bin`; in the Variable Value field If the path doesn't already exist, click **Edit**, then enter `c:\j2sdk1.4.2`; in front of the other variables as shown in Figure 2.*
6. *Click **OK** (closes Edit System Variables window)*
7. *Click **OK** (closes Environmental Variables window)*



Variable Name: Path

Variable Value: ort\Libraries\MFC\Libs;C:\j2sdk1.4.2_05\bin;

System variables

Variable	Value
OS	Windows_NT
Os2LibPath	C:\WINNT\system32\os2\dll;
Path	C:\WINNT\system32;C:\WINNT;C:\WINN...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE...
PROCESSOR_ARC...	x86

Figure 2. Edit System Variable window (Windows 2000)

Installing the Motorola SDK Components

This section describes how to install the Motorola SDK for the J2ME platform software on your PC.

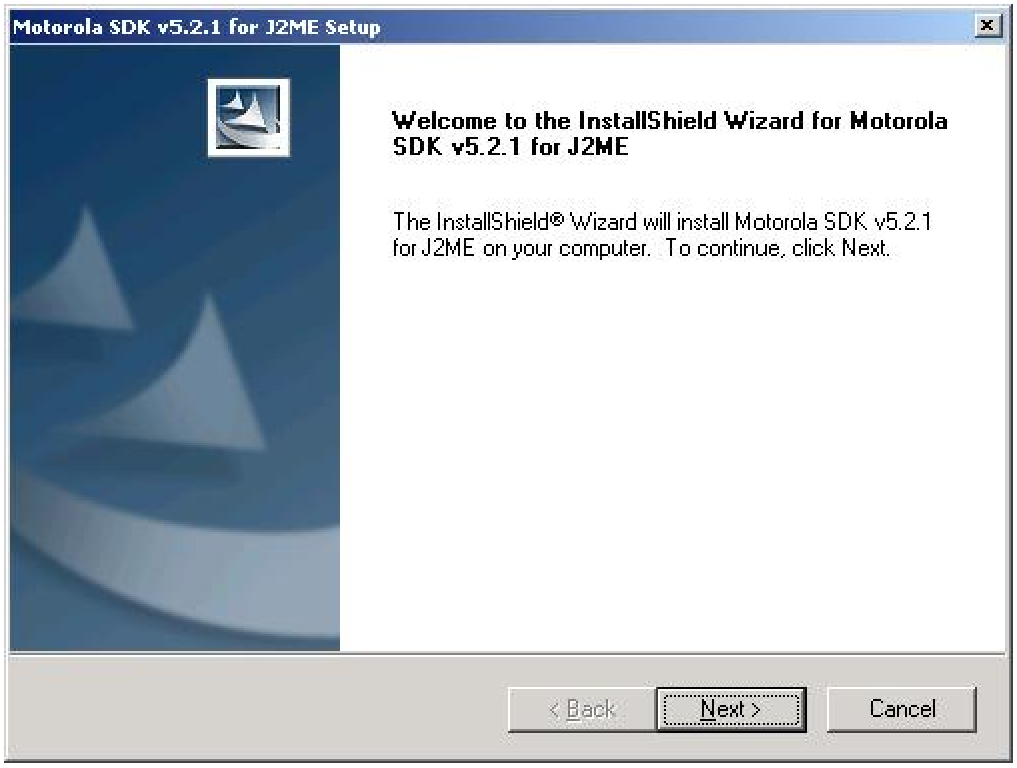
You no longer need to uninstall previous version of the Motorola SDK.

The installer places the latest version in its own folder within the `<Motorola_SDK_HOME>` folder, enabling you to develop and test your software across different releases.

Installing the Motorola SDK Components

You must install the Motorola SDK Components to use them in your developer efforts.

We highly recommend that you install the Motorola SDK v5.2.1 for J2ME components using the default directory specified by the SDK installer. Otherwise problems could occur later.


Motorola SDK v5.2.1 for J2ME Setup

Welcome to the InstallShield Wizard for Motorola SDK v5.2.1 for J2ME

The InstallShield® Wizard will install Motorola SDK v5.2.1 for J2ME on your computer. To continue, click Next.

Figure 3. Installer start-up window

To install the Motorola SDK:

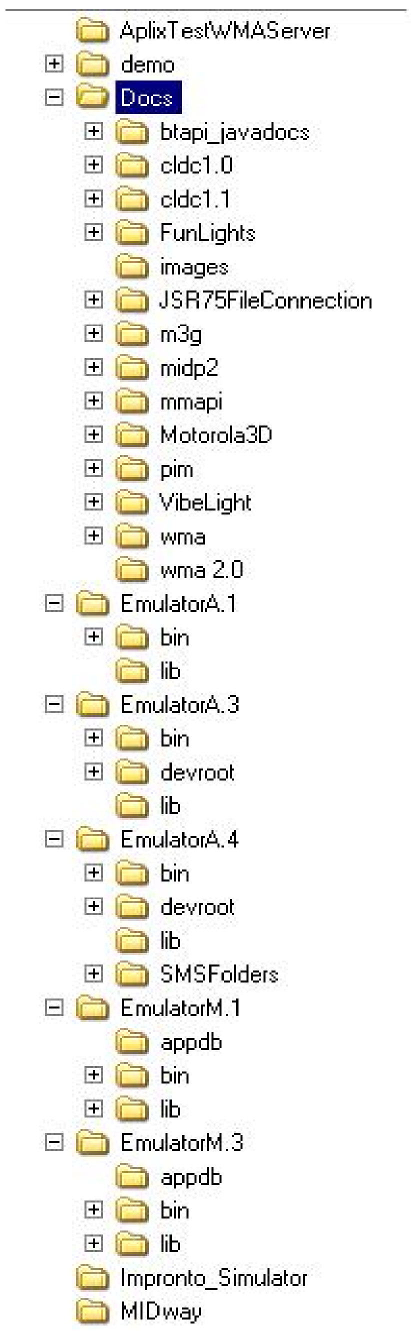
1. *Run the `Motorola_SDK_v5.2.1_for_J2ME.exe` installer program Run the installer program by using one of the following ways:*
   - Double-click the `Motorola_SDK_v5.2.1_for_J2ME` icon in the **Window Explorer** window

- From a command line, enter
  `Motorola_SDK_v5.2.1_for_J2ME.exe`

2. *Follow the prompts in the installer program*

Motorola SDK Directory Structure

The installer automatically installs Motorola SDK Components in your `<MOTOROLA_SDK_HOME>` directory unless you indicate otherwise. Once installed, the Motorola SDK Components directory structure looks like this:



```
AplixTestWMAServer
demo
Docs
    btapi_javadocs
    cldc1.0
    cldc1.1
    FunLights
    images
    JSR75FileConnection
    m3g
    midp2
    mmapi
    Motorola3D
    pim
    VibeLight
    wma
    wma 2.0
EmulatorA.1
    bin
    lib
EmulatorA.3
    bin
    devroot
    lib
EmulatorA.4
    bin
    devroot
    lib
    SMSFolders
EmulatorM.1
    appdb
    bin
    lib
EmulatorM.3
    appdb
    bin
    lib
Impronto_Simulator
MIDway
```

Figure 4. Motorola SDK directory structure

Use of the default directory location is highly recommended, as several of the scripts that accompany the SDK are dependent upon it. Relocation may cause script errors.

The table below lists the major directories plus a summary of their contents.

Sub-Directories	**Contents**
AplixTestWMAServer\	Contains the server used for testing the WMA functionality
demo\	Demonstration scripts and files
demo\com\mot\j2me\midlets	Contains various MIDlets and LWT example files
demo\Scripts	Contains the following scripts: • makeAll.bat -builds all MIDlets, tests,and tutorials • makeOneEA1.bat- builds a single MIDlet that uses the E A.1 emulator • makeOneEA3.bat- builds a single MIDlet that uses the E A.3 emulator • makeOneEA4.bat- builds a single MIDlet that uses the E A.4 emulator • makeOneEM1.bat- builds a single MIDlet that uses the E M.1 emulator • makeOneEM3.bat- builds a single MIDlet that uses the E M.3 emulator
Docs\Start.htm	Used to access all the API's documentation.
Docs\	Includes documentation for Motorola SDK Components Guide (this guide : Motorola SDK for J2ME Users Guide.pdf)
EmulatorA.1	Contains the bin and lib for the Emulator A.1 release
EmulatorA.1\bin\	Contains files for the configuration editor, emulator and resources
EmulatorA.1\lib\	Class libraries (com, java, javax) for Emulator A.1
EmulatorA.3	Contains the bin and lib for the EmulatorA.3 release
EmulatorA.3 \bin\	Contains files for the configuration editor, emulator and resources
EmulatorA.3 \lib\	Class libraries (com, java, javax) for EmulatorA.3
EmulatorA.4	Contains the bin and lib for the EmulatorA.4 release

EmulatorA.4\bin\	Contains files for the configuration editor, emulator and resources
EmulatorA.4\lib\	Class libraries (com, java, javax) for EmulatorA.4
EmulatorM.1	Contains the bin and lib for the Emulator M.1 release
EmulatorM.1\bin\	Contains files for the configuration editor, emulator and resources
EmulatorM.1\lib\	Class libraries (com, java, javax) for Emulator M.1
EmulatorM.3	Contains the bin and lib for the Emulator M.3 release
EmulatorM.3 \bin\	Contains files for the configuration editor, emulator and resources
EmulatorM.3 \lib\	Class libraries (com, java, javax) for Emulator M.3
Impronto_Simulator	Contains the Impronto Simulator application used to emulate the Bluetooth hardware.
MIDway\	Contains the MIDway application used to transfer the midlets from a PC to the phone through the USB port.

Table 5. Directory Structure Table

Tutorial: Developing an Application

This tutorial shows you how to develop a J2ME MIDP-compliant application, called a MIDlet, using the Motorola SDK Components. In this tutorial you will write, build, compile, and run an application from the command line. The steps include:

- Step 1: Writing an Application
- Step 2: Compiling an Application
- Step 3: Running and Testing an Application

Before you begin, note the following:

- The Java language is case-sensitive; therefore, you must type all Java code exactly as shown in the example.
- Applications developed with the Motorola SDK Components must extend `javax.microedition.midlet.MIDlet` and implement the following three methods:

J2ME Application Methods

Method	Description
`protected void startApp()`	Starts a MIDlet and puts it in Active state.
`protected void pauseApp()`	Stops a MIDlet and puts it in Paused state.
`protected void destroyApp(boolean unconditional)`	Terminates a MIDlet and puts it in Destroyed state.

Table 6. J2ME Application Methods Table

The chart in Figure 5 illustrates the flow of these application methods.

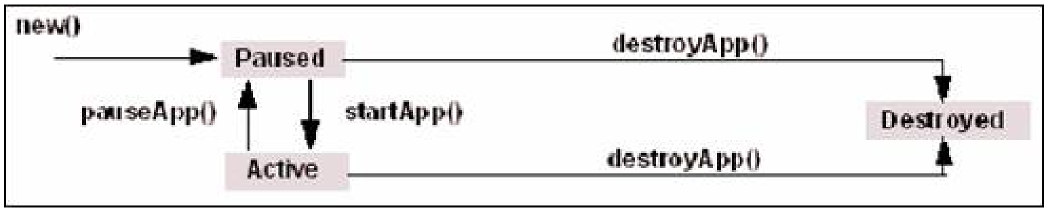


Figure 5. Application methods chart

Step 1: Writing an Application

In this tutorial, you will develop the application HelloWorld. When you compile and run this application, it displays "Hello World. This is a J2ME MIDlet" on the simulator.

The first step to developing is writing. For this tutorial, instead of writing a new application, modify the HelloWorld.java file included with the Motorola SDK Components. This file is located at:

`<MOTOROLA_SDK_HOME>\demo\com\mot\j2me\midlets\tutorials\HelloWorld.java`

Modifying the HelloWorld Application

In the code editor of your choice, open and modify `HelloWorld.java` to match the source code shown in List 1. After modifying the file to match the example, save and close `HelloWorld.java`.

```
/*
* HelloWorld.java
*
* Jan 1, 2003
*
* © Copyright 2003 Motorola, Inc. All Rights Reserved.
* This notice does not imply publication.
*/

package com.mot.j2me.midlets.tutorials;

import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

/**
* A simple Hello World midlet
*
* @see MIDlet
*/
public class HelloWorld extends MIDlet {
/**
* Main application screen
*/
private Form mainScreen;

/**
* A reference to the Display
*/
private Display myDisplay;
/**
* Creates a screen for our midlet
*/
HelloWorld() {

myDisplay = Display.getDisplay(this);
mainScreen = new Form("Hello World");
/*
* Create a string item
*/
StringItem strItem = new StringItem("Hello",
"This is a J2ME MIDlet.");
mainScreen.append(strItem);

}

/**
* Start the MIDlet
*/
public void startApp() throws MIDletStateChangeException {
myDisplay.setCurrent(mainScreen);
}

/**
* Pause the MIDlet
*/
public void pauseApp() {
}

/**
* Called by the framework before
* the application is unloaded
*/
public void destroyApp(boolean unconditional) {
}
}
```

List 1. HelloWorld.java example

Step 2: Compiling an Application

You compile the sample MIDlets included with the Motorola SDK using the `makeAll` batch file. This file will invoke the java compiler "javac" for each of the midlets, tests, and tutorials found under the `<MOTOROLA_SDK_HOME>\demo` directory. A separate script is included in each source directory to build the targets in that directory.

Compiling the HelloWorld MIDlet

1. *Change to the directory of the compiler. Enter:*

```
cd Program Files\Motorola\SDK v5.2.1 for
J2ME\demo\Scripts
```

2. *Run the compiler. Enter:*

   ```
   makeAll
   ```

If `makeAll` does not run, verify that the JDK 1.4 install directory and bin subdirectory are in the PATH system environment variable. For more information, see the section "Verifying the Path Environment Variable".

Using the Motorola SDK Components

The Motorola SDK Components consist of a configuration editor, a simulator and sample applications. Use the information in this chapter to add new device configurations to the development environment. In most cases, device configurations are already supplied and ready to use. However, in those rare cases where you need a custom configuration, this chapter can help you setup a new device configuration.

The topics covered in this section include:

- **Working with the Emulator:** Describes the workings of the simulator, including running and debugging MIDlets.
- **Sample Applications:** Brief descriptions of the various sample applications included with the Motorola SDK.

Working with the Emulator

The emulator is a tool you can use to design and debug J2ME applications using a desktop computer. As described in the previous section, Working with Configurations, you can configure the emulator to emulate different types of devices such as cell phones, personal digital assistants (PDA's), and pagers.

The emulator is an interactive image of your device. You can interact with the emulator through the keys on the image, the desktop keyboard, or the mouse. The emulator can also display output of an application on the virtual LCD of the emulated device.

Only the J2ME environment is emulated; standard cell phone features such as placing a call are not supported.

The profiles for the Motorola SDK Components under Emulator A.1 include:

- Motorola A630
- Motorola A845
- Motorola C380

- Motorola C381p
- Motorola C381/C386
- Motorola C384/C385/C390/C391
- Motorola C650/C651
- Motorola C698p
- Motorola E375
- Motorola E398
- Motorola E550/V535/V560
- Motorola T725
- Motorola V180/V185/V186/V188
- Motorola V220/V220i/V226
- Motorola V3/RAZR V3b
- Motorola V300/V400/V500
- Motorola V330/V547/V551/V555
- Motorola V400p/V303p
- Motorola V550/V545
- Motorola V600
- Motorola V600i
- Motorola V620
- Motorola V635
- Motorola V80

The profiles for the Motorola SDK Components under Emulator A.3 include:

- Motorola C975
- Motorola C980
- Motorola E1000/E1000R
- Motorola RAZR V3g
- Motorola V975
- Motorola V980

The profiles for the Motorola SDK Components under Emulator A.4 include:

- Motorola PEBL
- Motorola RAZR V3i
- Motorola ROKR E1
- Motorola SLVR
- Motorola V190
- Motorola V230/V235
- Motorola V270/V280
- Motorola V360/V361
- Motorola V540/V557/V557p

The profiles for the Motorola SDK Components under Emulator M.1 include:

- Motorola A760/A760i
- Motorola A768/A768i

The profiles for the Motorola SDK Components under Emulator M.3 include:

- Motorola A780
- Motorola E680/E680i

You can use these cell phone profiles to test your applications with different memory resources, LCD screen sizes, and keypad layouts. See Configuring the Emulator for the Target Device for more information on these configuration profiles.

Figure 6. Example profile images include A845, V180/V185/V186/V188, A630 (not shown to scale)

If you have trouble running the emulator, you may need to check the system environment path. See Verifying the Path Environment Variable for more information. The emulator may also display debugging information in the command line window.

Using the Emulator Keys

Use the guidelines shown below when using the emulator keys to simulate device keys:

Emulator keys

Key Action	Simulator
Left, right soft keys	• Use the mouse • Page Up, Page Down
0-9, *, #, left, right, up, down	• Use the mouse • Press keyboard numbers (not numeric keypad)
Select an item	• Press Enter • Click phone key on the emulator

Table 7. Emulators Keys Table

Exiting the Emulator

To exit the emulator:

- Click the emulator window's close box

Configuring the Emulator for the Target Device

You can use the device configurations provided with the Motorola SDK Components, or create new configurations for your own device. The tables below, Emulator A.1 Device Configurations, Emulator M.1 Device Configurations, Emulator A.3 Device Configurations, Emulator A.4 Device Configurations and Emulator M.3 Device Configurations list the available configurations for each emulator.

Currently, the configuration of properties files must be done manually.

Emulator A.1 Device Configuration

Device Emulated	**Device Properties**
A630	A630.props A630.jpg
A845	A845.props A845.jpg
C380	C380.props C380.jpg
C381p	C381p.props C381p.jpg
C381 / C386	C381_C386.props C381.jpg
C384 / C385 / C390 / C391	C384_C385_C390_C391.props C384_C385_C390_C391.jpg
C650 / C651	C650_C651.props C650_C651.jpg
C698p	C698p.props C698p.jpg
E375	E375.props E375.jpg
E398	E398.props E398.jpg
E550/V535/V560	E550_V535_V560.props E550_V535_V560.jpg
T725	T725.props T725.jpg
V180 / V185 / V186 / V188	V180_V185_V186_V188.props V180_V185_V186_V188.jpg
V220 / V220i / V226	V220_V220i_V226.props V220_V220i_V226.jpg
V3/RAZR V3b	V3_RAZR_V3b.props V3_RAZR_V3b.jpg
V300 / V400 / V500	V300_V400_V500.props V300_V400_V500.jpg
V330 / V547 / V551 / V555	V330_V547_V551_V555.props V330_V547_V551_V555.jpg

V400p / V303p	V400p_V303p.props V400p_V303p.jpg
V550 / V545	V550_V545.props V550_V545.jpg
V600	V600.props V600.jpg
V600i	V600i.props V600i.jpg
V620	V620.props V620.jpg
V635	V635.props V635.jpg
V80	V80.props V80.jpg

Table 8. A.1 Table

Emulator A.3 Device Configuration

Device Emulated	Device Properties
C975	C975.props C975.bmp
C980	C980.props C980.bmp
E1000/E1000R	E1000_E1000R.props E1000_E1000R.bmp
RAZR V3g	RAZR_V3g.props RAZR_V3g.bmp
V975	V975.props V975.bmp
V980	V980.props V980.bmp

Table 9. A.3 Table

Emulator A.4 Device Configuration

Device Emulated	Device Properties
PEBL	Pebl.props Pebl.bmp
RAZR V3i	RAZR_V3i.props RAZR_V3i.bmp
ROKR E1	ROKR_E1.props ROKR_E1.bmp
SLVR	SLVR.props SLVR.bmp
V190	V190.props V190.bmp
V230/V235	V230_V235.props V230_V235.bmp
V270 / V280	V270_V280.props V270_V280.bmp
V360 / V361	V360_V361.props V360_V361.bmp
V540 / V557 / V557p	V540_V557_V557p.props V540_V557_V557p.bmp

Table 10. A.4 Table

Emulator M.1 Device Configuration

Device Emulated	Device Properties
A760 / A760i	A760_A760i.props A760_A760I.bmp
A768 / A768i	A768_A768i.props A768_A768I.bmp

Table 11. M.1 Table

Emulator M.3 Device Configuration

Device Emulated	Device Properties

A780	A780.props A780.bmp
E680 / E680i	E680_E680i.props E680_E680i.bmp

Table 12. M.3 Table

Debugging J2ME Applications on the Device

Once you have your J2ME applications working on the device emulator, it is time to test the applications on the device itself. If your integrated development environment (IDE) comes with a debug agent for debugging J2ME applications on the emulator, you may be able to debug those applications on the actual device. To use your IDE for on-device debugging, the following conditions must be met:

- The device must have J2ME debugging capability enabled and be KDWP-compliant.
- The debug agent talks to the device using the KVM Debug Wire Protocol (KDWP).
- The manufacturer must have provided a means of downloading the MIDlets to the device for debugging.
- The medium of KDWP communication used by the phone (i.e., serial port or UDP) must be supported by the IDE and its debug agent.
- If you are using Metrowerks CodeWarrior® as your IDE, see Appendix A: Using CodeWarrior Wireless Studio for instructions on using the debug agent.

Sample Applications

The sample applications described are demos that give you an idea of the application capabilities available for the J2ME platform. These samples are stored in the `<MOTOROLA_SDK_HOME>\demo\com\mot\j2me\midlets` directory.

Some of these MIDlets appear in the sub-directories `Tests` or `Tutorials`.

For details on a particular sample application, see Appendix B: Sample MIDlets.

Working with Impronto Simulator

The Java APIs for Bluetooth Wireless Technology (JABWT) specified in

JSR 82 is an API for implementing Bluetooth connectivity between multiple Bluetooth enabled devices. Confirming this functionality on a Windows based emulator requires some means of simulating communication between emulated devices. The Impronto Simulator described in [9] serves this purpose. Using this tool, developers can test the application using the JABWT functionality without actual devices.

There are a number of environment variables that must be set in order to run the Impronto Simulator:

SIMULATOR_HOME The absolute path to the impronto_simulator\impronto directory.

JAVA_HOME Location of the Java SDK installation on the machine.

SIM_HOST The IP address or host name of the machine that the simulator is running on. The string "localhost" should be set.

SIM_FRIENDLY_NAME The friendly name of the virtual device being launched in the simulator. The virtual device with the name specified in this environment variable must have been created using the Bluetooth Simulator Console. A virtual device is a Bluetooth enabled virtual device that works with the Impronto Simulator. To create a virtual device, characteristics of the device such as Bluetooth address, its friendly name and type of service offered by the device must be set in the Bluetooth Simulator Console. The friendly name is used to associate the device emulator with the virtual device when the device emulator runs.

How to integrate Impronto Simulator with Launchpad

1. Set enviroment variables

- SIMULATOR_HOME= <Impronto simulator home>
- SIM_HOST=localhost
- SIM_XML_VERBOSE=0

2. For each MIDlet execution, these cases should be taken into account

- SIM_FRIENDLY_NAME= < XML file >
- It is not possible to execute two MIDlets with the same SIM_FRIENDLY_NAME
- Into each XML file the "bluetoothAddress" atribute shall be unique.
- Into each XML file the "friendlyName" atribute shall be accodording with SIM_FRIENDLY_NAME *
- The XML filename shall be according with SIM_FRIENDLY_NAME *

Note.: The XML file is located in "SIMULATOR_HOME/etc/"

3. In order to run a Bluetooth application in SDK, the emulator must be started from a command prompt, after the Impronto Simulator is running, due to the fact that the Impronto Simulator needs environment variable

settings for each Bluetooth application. To start a MIDlet in SDK from the command prompt, follow the steps described in the section "Save Command to Batch File" of this document. Or create a batch file that first sets the needed environment variables, and then launch a Bluetooth specific midlet.

Setting the environment variables:

- `set SIM_HOST=localhost`
- `set SIM_FRIENDLY_NAME=sample_jabwt_chat_Client2`
- `set SIM_XML_VERBOSE=0`

Starting Launchpad from Command Prompt:

1. *`C:`*
2. *`cd "\Program Files\Motorola\SDK v5.2.1 for J2ME\EmulatorA.4\bin"`*
3. *`c:\Program Files\Motorola\SDK v5.2.1 for J2ME\EmulatorA.4\bin\jblend.exe -Xdescriptor:"C:\sampleApplications\sample_bluetooth.jad" -Xdevice:PEBL`*

Note that `c:\sampleApplications\sample_bluetooth.jad` is just an example of usage, this midlet does not come with SDK.

For further details on starting the Impronto Simulator to work with SDK in order to run a Bluetooth application, see [8].

Note: the JSR 82 Bluetooth API is available only for Emulator A.4

Working with WMA Test Server

The Wireless Messaging API (WMA) specified in JSR 205 is an API for implementing messaging between a cellular phone and outside server. Confirming this functioning on an emulator requires some means of simulating communication between the mobile device and outside server. The JSR 205 Wireless Messaging API Connection Test Tool described in [9] servers this purpose. Using this tool, the developer can test the basic WMA functions.

In order to run a JSR 205 application in SDK, the emulator must be started after the WMA Test Server is running,

To start running the WMA Test Server, enter the following command statement in a command prompt:

- SMS/CBS: `TestWmaServer`
- MMS: `TestWmaServer -server_port 20001 -client_port 20000`

To receive messages in the WMA Test Server, enter the following command statement in the command prompt where the WMA Test Server is running:

- SMS/CBS: `tx SMS send.txt`
- MMS: `tx MMS sendMulti.txt`

Where `send.txt` and `sendMulti.txt` are the files that needed to be received.

For details on starting the WMA Test Server to work with SDK in order to run a JSR 205 application, see [9]. The steps needed to compose MMS and SMS messages are also described in [9].

Note: the JSR 205 WMA 2.0 API is available only for Emulator A.4

Using the Launchpad Application

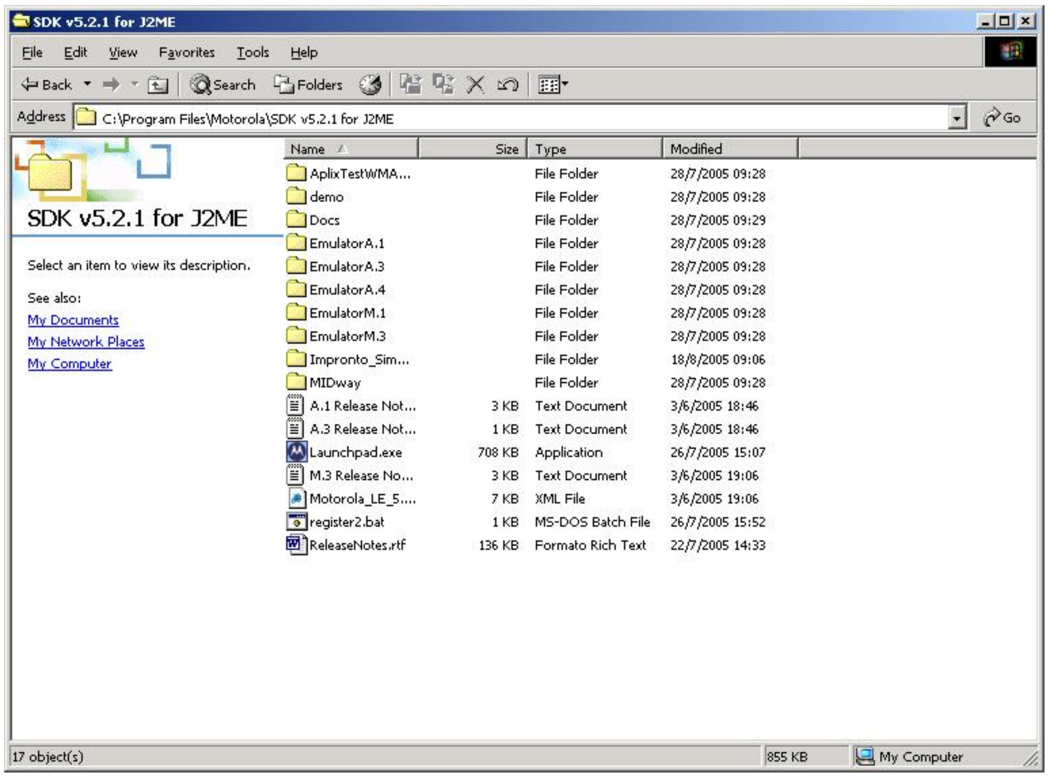
The Launchpad application assists you in the process of MIDlet application development. It does this by providing an easy-to-use GUI that lets you quickly launch MIDlets using the appropriate emulation and a particular handset.

The topics in this section include:

- **Where to find Launchpad:** Contains instructions on where to locate the Launchpad executable in the J2ME™ directory.
- **The Launchpad UI:** Describes the controls that let you select a specific MIDlet, emulator and handset skin by simply pointing and clicking.
- **Running Launchpad:** Describes how to use Launchpad to start a demo MIDlet program.
- **Saving a Launchpad configuration:** Describes how you can save the configuration. Launchpad uses to start a specific MIDlet into a `.bat` file for future use.

Where to Find Launchpad

The Launchpad executable file, Launchpad.exe, is in the installation directory made by the installer application. By default, this is the Motorola SDK v5.2.1 for J2ME directory. Figure 7 shows where the SDK v5.2.1 for J2ME directory is located at `c:\Program Files\Motorola`, but this directory could be elsewhere.



Name	Size	Type	Modified
AplixTestWMA...		File Folder	28/7/2005 09:28
demo		File Folder	28/7/2005 09:28
Docs		File Folder	28/7/2005 09:29
EmulatorA.1		File Folder	28/7/2005 09:28
EmulatorA.3		File Folder	28/7/2005 09:28
EmulatorA.4		File Folder	28/7/2005 09:28
EmulatorM.1		File Folder	28/7/2005 09:28
EmulatorM.3		File Folder	28/7/2005 09:28
Impronto_Sim...		File Folder	18/8/2005 09:06
MIDway		File Folder	28/7/2005 09:28
A.1 Release Not...	3 KB	Text Document	3/6/2005 18:46
A.3 Release Not...	1 KB	Text Document	3/6/2005 18:46
Launchpad.exe	708 KB	Application	26/7/2005 15:07
M.3 Release No...	3 KB	Text Document	3/6/2005 19:06
Motorola_LE_5....	7 KB	XML File	3/6/2005 19:06
register2.bat	1 KB	MS-DOS Batch File	26/7/2005 15:52
ReleaseNotes.rtf	136 KB	Formato Rich Text	22/7/2005 14:33

Figure 7. Location of the Launchpad executable in the SDK v5.2.1 for J2ME directory.

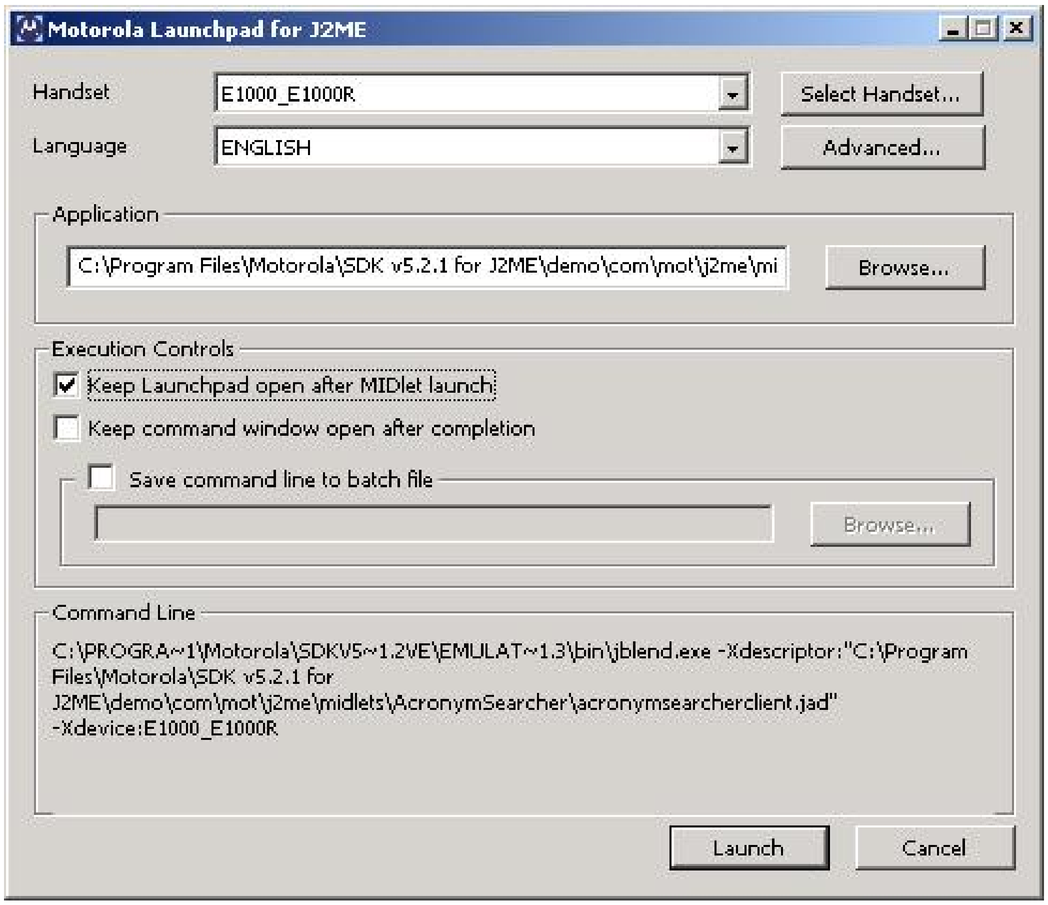
The Launchpad UI

Launchpad presents a UI that allows you to quickly select an emulator and a handset skin to execute and test with a particular MIDlet. This section explains the details of the UI and how you apply them to configure and launch a Java MIDlet.

The topics in this section include:

- The Launchpad window
- The Launchpad Components
- Using Launchpad to run a MIDlet
- The Launchpad Emulator Thumbnail Window
- The Launchpad Advanced Settings window

When Launchpad starts, it presents the Motorola J2ME Launchpad window, as shown in Figure 8.



Motorola Launchpad for J2ME

Handset: E1000_E1000R

Language: ENGLISH

Application: C:\Program Files\Motorola\SDK v5.2.1 for J2ME\demo\com\mot\j2me\mi

Execution Controls

[✓] Keep Launchpad open after MIDlet launch

[ ] Keep command window open after completion

[ ] Save command line to batch file

Command Line

```
C:\PROGRA~1\Motorola\SDKV5~1.2VE\EMULAT~1.3\bin\jblend.exe -Xdescriptor:"C:\Program
Files\Motorola\SDK v5.2.1 for
J2ME\demo\com\mot\j2me\midlets\AcronymSearcher\acronymsearcherclient.jad"
-Xdevice:E1000_E1000R
```

Figure 8. The Motorola J2ME Launchpad window

This window presents the controls that configure and assemble a command line that Launchpad passes to the Java™ Runtime Environment (JRE™) for execution. The window has various components for selecting the handset type, the language for the emulation to display, the name of the class in the MIDlet to execute, and diagnostic control settings. The table below summarizes the purpose of these components.

Launchpad window components

Component	Description
Handset	Chooses the handset skin that the MIDlet interacts with.
Language	Chooses the language-appropriate display fonts that the emulator uses.
Select Handset	Displays a window with thumbnails of the supported handsets. You pick a

	handset and the language that it uses from this window.
Advanced	Displays a window where you can adjust advanced options.
Browse	Displays a window to select a JAD file to run a MIDlet.
Keep Launchpad open after MIDlet launch	Launchpad does not exit after issuing the command line to the JRE.
Keep command window open after completion	The command window does not exit when the MIDlet exits.
Save command line to batch file	Saves the current configuration of Launchpad's command line into a .bat file
Command line	Displays the command line that Launchpad generates.

Table 13. Launchpad's components Table

Launchpad Components

This section describes the UI components presented by the Motorola J2ME Launchpad window in further detail. The topics covered in the section are:

- Handset
- Language
- Select Handset
- Advanced
- Browse
- Keep Launchpad open after MIDlet launch
- Keep command window open after completion
- Save Command to Batch file
- Command line

Handset

The Launchpad application scans the installation directory tree to locate any installed handset properties (.props) files. Each discovered handset is attached to the drop-down list that the Handset component displays. When you select a specific handset, Launchpad uses this information to determine what languages it supports. Launchpad then updates the Language component to display only valid language selections for the chosen handset. In addition, the choice of handset determines what skin the Display Emulator presents when the MIDlet executes. Use the Select Handset button to open the Simulator Thumbnail window. You use this window to choose a handset and--for certain skins--the language that it displays. For more information, consult the section, Simulator Thumbnail Window.

Language

This component displays a drop-down list of languages available for the handset. This list is determined by the handset specified in the Handset component.

Select Handset

Click the Select Handset button to open Simulator Thumbnail Window. Select one of the available handsets and click OK.

Advanced

Click the Advanced button to open the Advanced Settings window. For more information, consult the section, Advanced Settings Window.

Browse

Click the Browse button to open a dialog box to select the JAD file.

You can use Browse button to search for a JAD file or type the directory path and file name into the text entry section of this component. If you fail to provide a file name, Launchpad displays an error message.

When running from a JAD file, the Launchpad opens the file and searches for entries in the format `MIDlet-n:<value>`, where n is the number of the MIDlet in the suite. If no such entry exists, then Launchpad opens the corresponding JAR file, and reads the `MANIFEST.MF` file to determine the names of the MIDlet(s) stored there. Once Launchpad has determined the names of the stored MIDlets, then one of three actions will occur:

- No MIDlets-an error message appears
- A single MIDlet-the MIDlet is launched
- Multiple MIDlets-a dialog appears where you can select which MIDlet to launch.

Keep Launchpad open after MIDlet launch

Checking this component instructs Launchpad not to exit after you click on the Launch button. In addition, the Motorola J2ME Launchpad window remains visible. This is valuable when executing several MIDlets simultaneously.

Keep command window open after completion

Checking this component has the command line window remain open after the MIDlet exits. This feature is useful if the MIDlet uses the standard output device to display diagnostic information. It is also useful for displaying error messages issued by the JRE if the MIDlet fails to launch.

Save command to batch file

Checking this component has Launchpad save the current settings required to execute the MIDlet into a batch (.bat) file. This is useful for assembling command sequences into sophisticated MIDlet test suites. For example, you might use Launchpad to generate a set of batch files that execute the same MIDlet, but with different skins. You then use an editor to string these batch files together so that you have one test suite batch file that tests the one MIDlet with every handset skin it supports. Type a file name into the text entry area of this component. The file extension should be .bat. Alternately, you can click on the **Browse** button to select a specific directory and choose an existing batch file. This feature can also be useful to start a MIDlet from the command line. Just check this component, type a file name into the text entry area, press the Launch button in the Launchpad and then the MIDlet will be launched and the command needed to start this same MIDlet will be saved in the batch file. After this, the MIDlet can be launched from the command line just running the generated batch file.

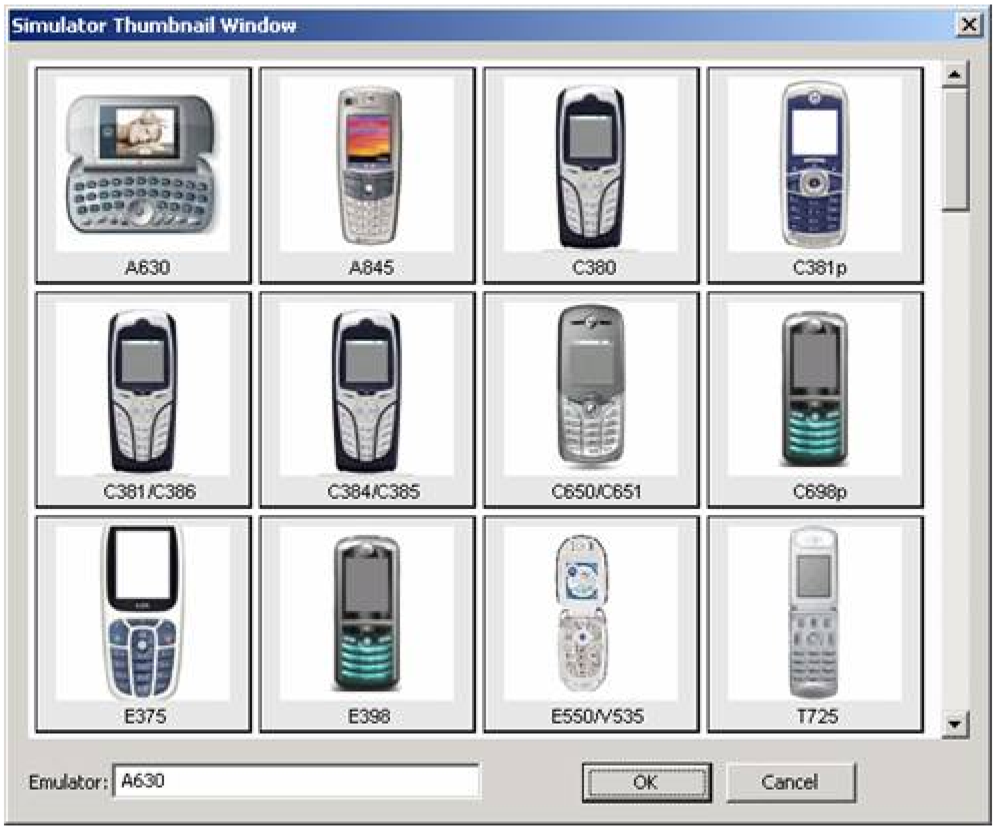
Command line

This component displays the command line that Launchpad generates to execute the MIDlet. This line changes as you make selections from the various components in the Motorola J2ME Launchpad window.

Simulator Thumbnail Window

This section describes the components of the Simulator Thumbnail

Window and their purpose. This window appears when you click Select Handset in the Launchpad for J2ME window. The Simulator Thumbnail Window (Figure 9) displays thumbnail images of all of the handsets that the J2ME release supports. Clicking on an image selects the simulator (or handset interface skin) to simulate, and it also determines the choice of emulator that executes the MIDlet code.



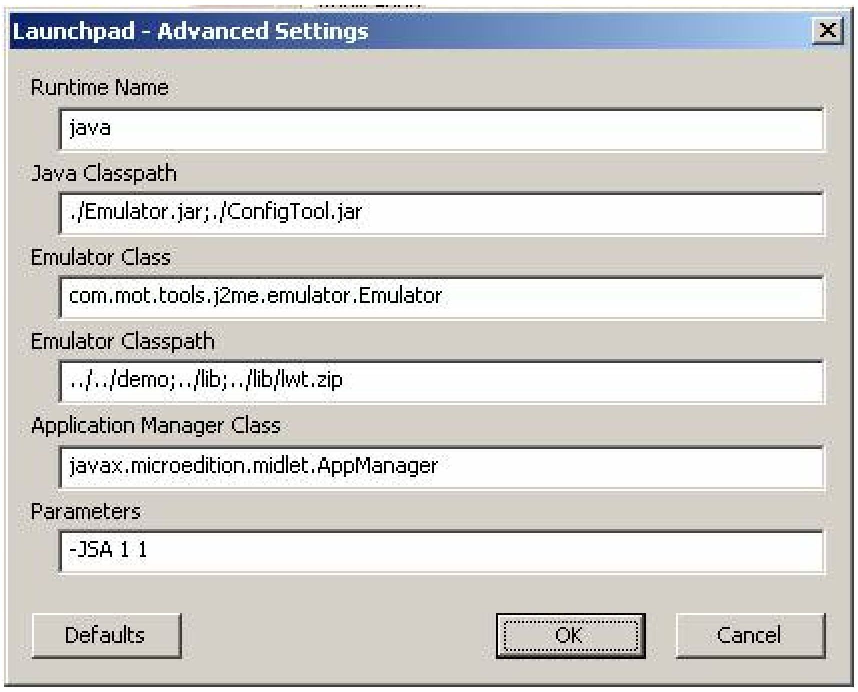
Emulator: A630

Figure 9. The Simulator Thumbnail Window

Each image provides the name of the handset that it represents. Certain image names also indicate the language fonts that the handset displays when the MIDlet executes. For other handsets to display different language fonts, you must make adjustments to the handset's .props files or elsewhere. For more information on managing the fonts that emulator uses, consult Appendix D, Using Foreign Fonts in Motorola Emulators.

Advanced Settings Window

This section describes the components of the **Launchpad-Advanced Settings** window and their purpose. When you click on the Advanced button in the Motorola J2ME Launchpad window, the Launchpad-Advanced Settings window appears, as shown in Figure 10.



Runtime Name: java
Java Classpath: ./Emulator.jar;./ConfigTool.jar
Emulator Class: com.mot.tools.j2me.emulator.Emulator
Emulator Classpath: ../../demo;../lib;../lib/lwt.zip
Application Manager Class: javax.microedition.midlet.AppManager
Parameters: -JSA 1 1

Figure 10. The Advanced Settings Window

Normally, the Launchpad application generates a syntactically correct command line to present to the JRE. However, the Advanced component allows you to modify the command line to issue parameters for special situations.

Advanced Settings window components

Componenet	Description
Runtime Name	This component selects the executable name of the Java runtime. The runtime executable should be accessible in the Windows PATH.
Java Classpath	Enter the directory path the Java runtime needs to find the specified emulator and config tool libraries.
Emulator Class	This is the class name of the handset emulator that executes the MIDlet.
Emulator Classpath	The directory path the emulator uses to

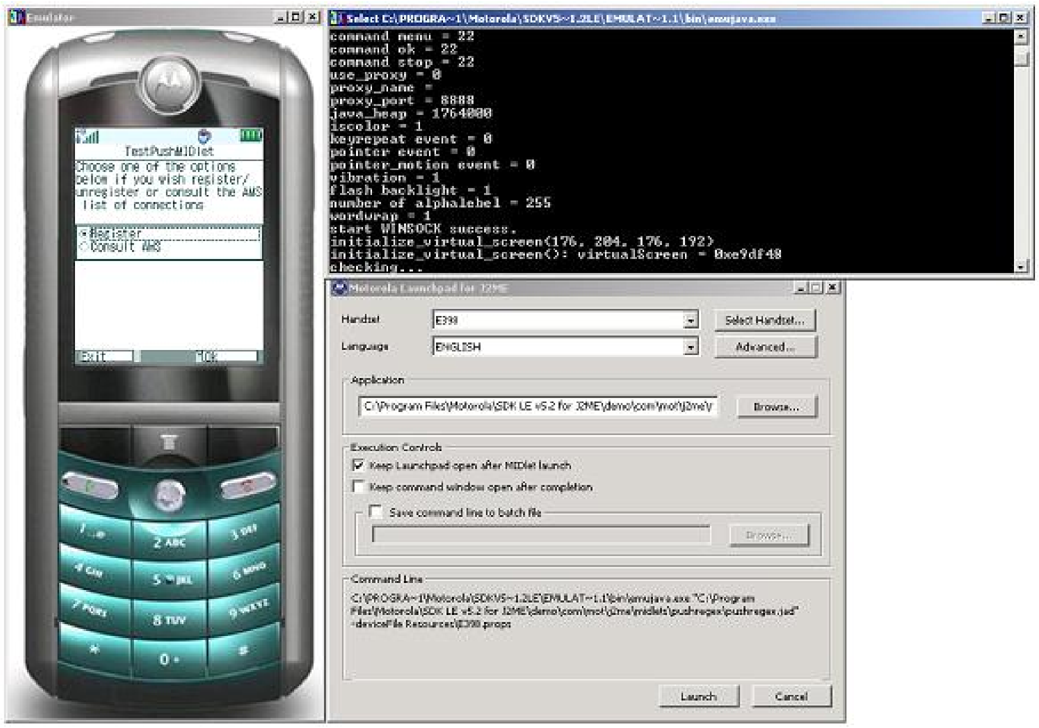
	locate the Java Application Manager (JAM™) and MIDlets.
Application Manager	Enter the class name of the Java Application Manager (JAM).
Parameters	Enter the Application Manager-specific parameters that modify its operation.

Table 14. Advanced Settings Table

Using Launchpad to Run a MIDlet

This section describes how to use Launchpad to execute one of the demo MIDlets.

1. *Start Launchpad by double-clicking on the launchpad.exe file. The* ***Motorola J2ME Launchpad*** *window appears.*
2. *Click on the* ***Handset*** *component and choose E398 from the pop-up menu list.*
3. *Click on the* ***Language*** *component and pick ENGLISH from the pop-up menu list.*
4. *Click on the* ***Application*** *component and type* *`com.mot.j2me.midlets.paddleball.PaddleBall`* *into the text entry field.*
5. *Click the* ***Browse*** *button and navigate to the* *`<MOTOROLA_SDK_HOME>\demo\com\mot\j2me\midlets`* *paddleball directory and select the paddleball.jar file.*
6. *Click Launch. The E398 skin should appear, and the paddleball MIDlet should start running (Figure 11).*


Emulator
TestPushMIDlet
Choose one of the options below if you wish register/ unregister or consult the AMS list of connections
Register
Consult AMS
Exit
OK
Select C:\PROGRA~1\Motorola\SDKV5~1.2LE\EMULAT~1.1\bin\emujava.exe
Motorola Launchpad for J2ME
Select Handset...
Advanced...
Browse...
Browse...
Launch
Cancel

```
command menu = 22
command ok = 22
command stop = 22
use_proxy = 0
proxy_name =
proxy_port = 8888
java_heap = 1764000
iscolor = 1
keyrepeat event = 0
pointer event = 0
pointer_motion event = 0
vibration = 1
flash backlight = 1
number of alphalebel = 255
wordwrap = 1
start WINSOCK success.
initialize_virtual_screen(176, 204, 176, 192)
initialize_virtual_screen(): virtualScreen = 0xe9df48
checking...
```

Handset: E398

Language: ENGLISH

Application: C:\Program Files\Motorola\SDK LE v5.2 for J2ME\demo\com\mot\j2me\

Execution Controls

[✓] Keep Launchpad open after MIDlet launch

[ ] Keep command window open after completion

[ ] Save command line to batch file

Command Line:

```
C:\PROGRA~1\Motorola\SDKV5~1.2LE\EMULAT~1.1\bin\emujava.exe "C:\Program
Files\Motorola\SDK LE v5.2 for J2ME\demo\com\mot\j2me\midlets\pushregex\pushregex.jad"
-deviceFile Resources\E398.props
```

Figure 11. Launchpad executing the PaddleBall MIDlet.

Saving a Launchpad Configuration

As discussed in the section on Launchpad Components, you use the Save command line to batch file component to save the command line that Launchpad generates. To save a working Launchpad configuration, do the following:

1. *Click on the Save command line to batch file component, so that a checkmark appears.*
2. *Type a file name into the text entry area of this component. The file extension should be .bat.*
3. *Alternately, you can you the Browse button to select a specific directory and an existing batch file.*
4. *Click on Launch. Launchpad saves the command line it generated to the specified file. If the file already exists, it is overwritten.*

Appendix A: Using CodeWarrior Wireless Studio

This appendix provides instructions on using Metrowerks' CodeWarrior™ Wireless Studio, version 7.0 and the Motorola SDK Components into its integrated development environment (IDE). If you use the CodeWarrior tools, do not install the Motorola SDK Components separately.

This appendix shows you how to:

- Install CodeWarrior Wireless Studio
- Create a J2ME Project
- Debug a MIDlet
- Runn a MIDlet on the Emulator
- Troubleshooting Tips

The instructions in this appendix primarily involve using the Motorola emulator and debug agent with the CodeWarrior development tools. For complete instructions on how to use the CodeWarrior IDE in general, consult the CodeWarrior documentation. For more information on Metrowerks, see www.metrowerks.com.

Installing CodeWarrior Wireless Studio

To successfully install the CodeWarrior Wireless Studio tools, you must:

- Remove any previous CodeWarrior for Java installation
- Remove any previous Motorola SDK Component installation
- Install the latest CodeWarrior Wireless Studio tools and Motorola SDK Components

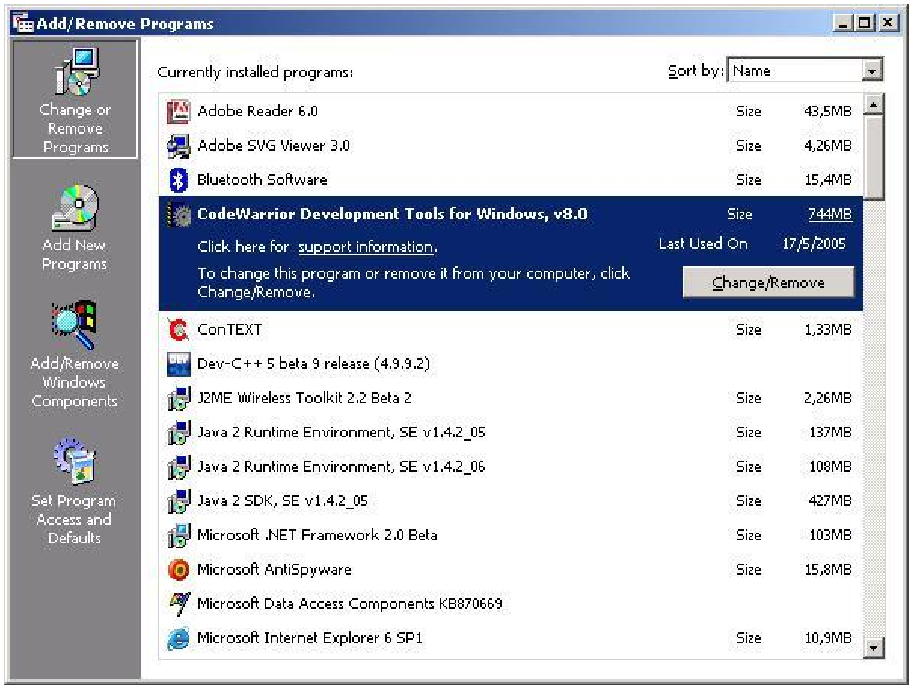
Removing Previous Versions of CodeWarrior Tools

If you have a previous version of the CodeWarrior for Java™ on your system, remove the old version using Window's built-in Add/Remove Programs control panel. To access this control panel:

- Select **Start | Settings | Control Panels | Add/Remove Programs**

OR

- Select **Start | Settings | Control Panels to show the Control Panels** window
- Double-click the **Add/Remove Programs** icon
- Use the control panel (Figure 12) to remove any previous versions of CodeWarrior for Java or CodeWarrior Wireless Studio.


Add/Remove Programs
Change or Remove Programs
Add New Programs
Add/Remove Windows Components
Set Program Access and Defaults
Currently installed programs:
Sort by: Name
Change/Remove

Adobe Reader 6.0	Size	43,5MB
Adobe SVG Viewer 3.0	Size	4,26MB
Bluetooth Software	Size	15,4MB
CodeWarrior Development Tools for Windows, v8.0	Size	744MB
Click here for support information.	Last Used On	17/5/2005
To change this program or remove it from your computer, click Change/Remove.		
ConTEXT	Size	1,33MB
Dev-C++ 5 beta 9 release (4.9.9.2)		
J2ME Wireless Toolkit 2.2 Beta 2	Size	2,26MB
Java 2 Runtime Environment, SE v1.4.2_05	Size	137MB
Java 2 Runtime Environment, SE v1.4.2_06	Size	108MB
Java 2 SDK, SE v1.4.2_05	Size	427MB
Microsoft .NET Framework 2.0 Beta	Size	103MB
Microsoft AntiSpyware	Size	15,8MB
Microsoft Data Access Components KB870669		
Microsoft Internet Explorer 6 SP1	Size	10,9MB

Figure 12. Use the Add/Remove Programs control panel to remove previous versions of the CodeWarrior tools.

Installing CodeWarrior Wireless Studio, version 7.0

To install CodeWarrior Wireless Studio:

1.Insert the CodeWarrior Wireless Studio CD

2.If the installer fails to auto-launch, double-click Setup.exe

3.Follow the online instructions making sure to select:

- **Setup Type**: Full Install
- **File Association Option**: Typical
- Click **Yes** to install Personal Java Emulation Environment

4.Reboot your computer after installing the CodeWarrior tools

Merging Motorola SDK v5.2.1 for J2ME Platform with CodeWarrior Wireless Studio

To have complete access to the Motorola SDK v5.2.1 for J2ME platform from within the CodeWarrior Wireless Studio, you must specify where it will find the Motorola SDK v5.2.1 for J2ME files. To do that, exit CodeWarrior Wireless Studio and follow these instructions:

1. Install the Motorola SDK v5.2.1 for J2ME Platform (if not already installed) See Installing the Motorola SDK Components for details.

The following instructions will not work if the Motorola SDK v5.2.1 for J2ME Platform installation was done to an alternate directory location. The default location is `c:\Program Files\Motorola\SDK v5.2.1 for J2ME\`.

2. Create a new directory called "Motorola SDK 5.2.1 " inside the `\Metrowerks\CodeWarrior\Java_Support\` directory

3. Copy the files "Motorola_5.2.1_J2ME.xml" and "register2.bat" from the `\Motorola\SDK v5.2.1 for J2ME\` directory to the `\MetrowerksCodeWarrior\Java_Support\Motorola SDK 5.2.1` directory

4. Open a command prompt window

5. Change to the `\Metrowerks CodeWarrior\Java_Support\ Motorola SDK v5.2.1` directory

6. Type "`register2`" to update the path to the Motorola SDK v5.2.1 for J2ME platform files

7. Restart CodeWarrior Wireless Studio

You now have complete access to the Motorola SDK v5.2.1 for J2ME files from within the CodeWarrior for Wireless development tools.

If CodeWarrior 7.0 was installed prior to installing the Motorola SDK v5.2.1 for J2ME, the SDK installer performs this registration process for you.

Creating a J2ME Project

In the following exercise, use bouncetest as the sample MIDlet to create a new J2ME project. The new project process involves:

- Creating a new J2ME project
- Deleting any unnecessary files from the project
- Adding additional files to the project

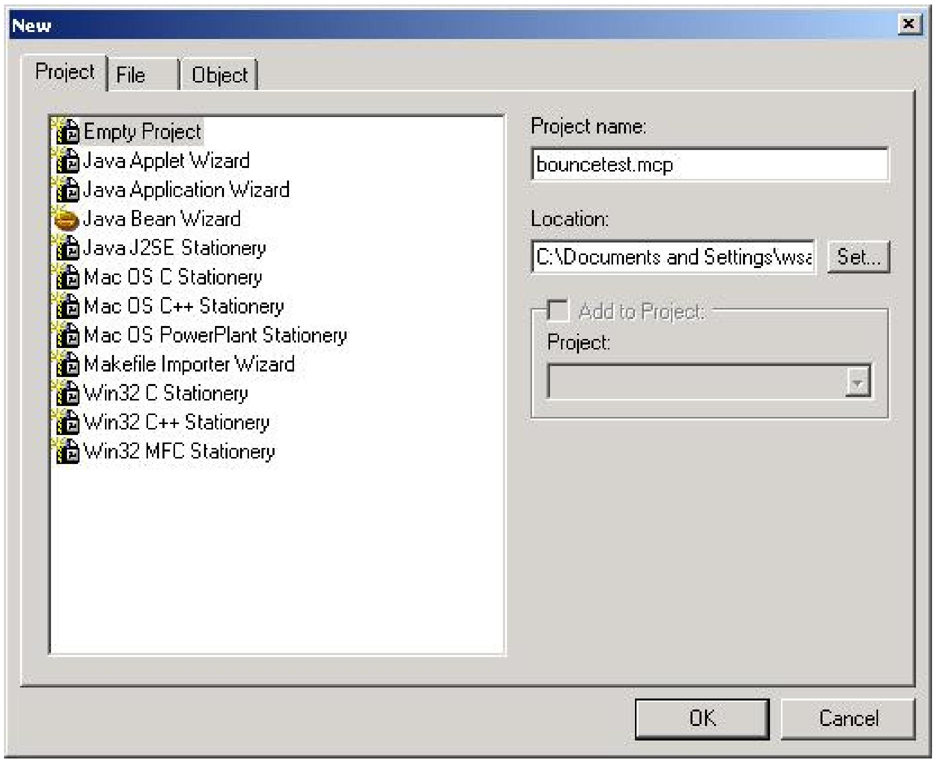
Creating a New Project

To create a new project named bouncetest.mcp:

1. Select **Start | Programs | CodeWarrior Wireless Studio, version 7.0 | CodeWarrior IDE**

2. Select **File | New** from the Metrowerks CodeWarrior window The **New** window appears (Figure 13).

3. Select **Java J2ME Stationery** from the Project list



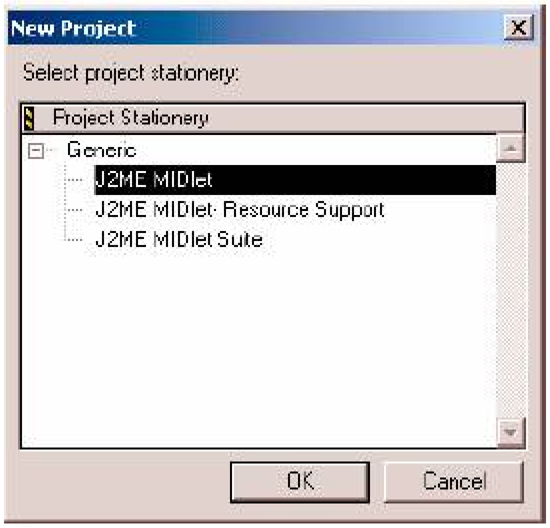
Project name: bouncetest.mcp

Location: C:\Documents and Settings\wsa

Figure 13. CodeWarrior's New window

4. Type a project name in the **Project name** field For this example, type bouncetest.mcp.

5. Click **OK** The **New Project** window (Figure 14) appears.



New Project

Select project stationery:

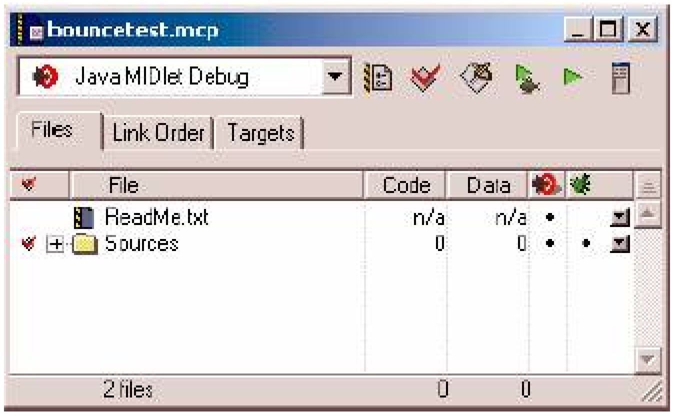
Project Stationery

- Generic
  - J2ME MIDlet
  - J2ME MIDlet Resource Support
  - J2ME MIDlet Suite

Figure 14. New Project window

6. Select and verify the new project is **J2ME MIDlet**, click **OK**.

The IDE generates a project window (Figure 15), in this case called bouncetest.mcp. This window shows the default sources of a J2ME MIDlet. The Sources group contains the default HelloWorld.java; however, for this example, you'll replace it with Bounce.java.



bouncetest.mcp

Java MIDlet Debug

File	Code	Data
ReadMe.txt	n/a	n/a
Sources	0	0
2 files	0	0

Figure 15. New bouncetest project window

Deleting Files from a Project

A project built from stationery often contains source files. If the original sources are not required by the project, you should remove them. In this example, the project does not need the HelloWorld.java file inside the Sources group. To remove it:

1. *Select `HelloWorld.java` in the Sources group*
2. *Select **Edit | Delete***
3. *Click **OK** to confirm the "Removing file from project..." warning*

The IDE removes the file from the project.

Adding Files to a Project

To make a project usable, you must add source files. In this example, the project needs the `Bounce.java` file.

To add a file:

1. In the bouncetest.mcp project window, highlight the Sources folder

2. Select Project | Add Files

3. Locate the Bounce.java file using the Select files to add dialog The file should be found at:
<MOTOROLA_SDK_HOME>\demo\com\mot\j2me\midlets\Bounce\Bounce.java

4. Click **Open The Add Files** dialog (Figure 16) appears, use it to select which build targets use the file.

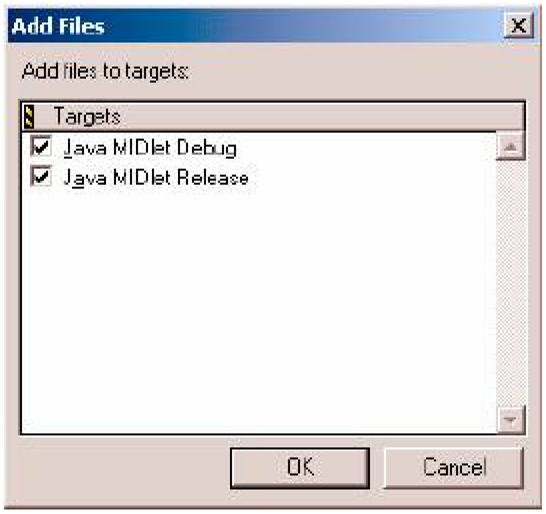
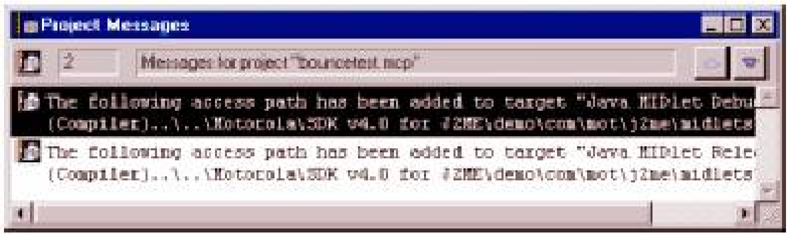


Figure 16. Add Files window

5. Leave **Debug** and **Release** options checked, click **OK** The IDE adds the file to the project and a **Project Messages** window appears if any messages are generated.

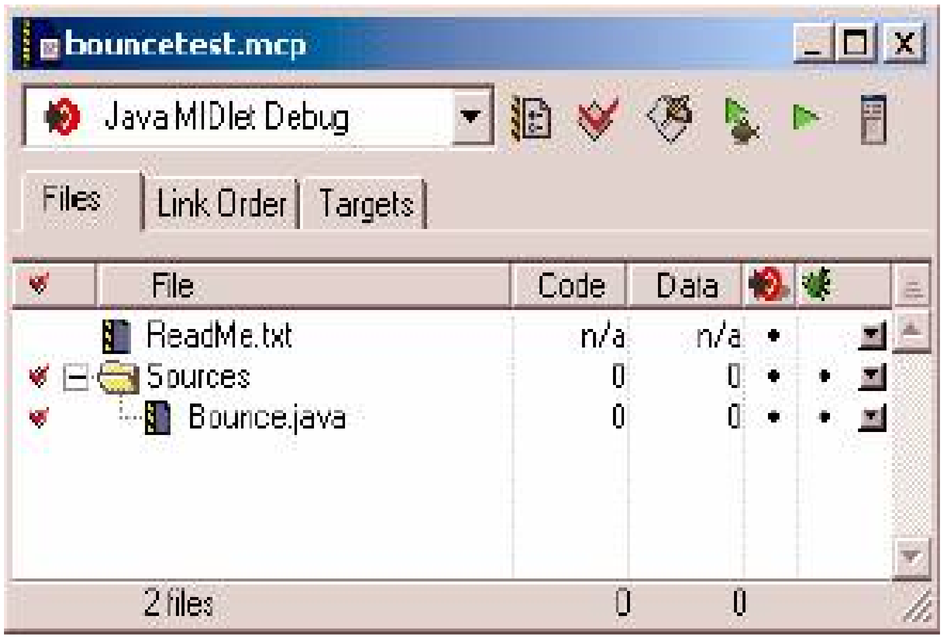
6. Close the **Project Messages** window (Figure 17)



```
The following access path has been added to target "Java MIDlet Debu
(Compiler)..\..\Motorola\SDK v4.0 for J2ME\demo\com\mot\j2me\midlets
The following access path has been added to target "Java MIDlet Rele
(Compiler)..\..\Motorola\SDK v4.0 for J2ME\demo\com\mot\j2me\midlets
```

Figure 17. Project Messages window

7. Open the **Sources** folder to make sure Bounce.java is there (Figure 18) The project is now ready for debugging.



	File	Code	Data		
	ReadMe.txt	n/a	n/a	•	
✓	Sources	0	0	•	•
✓	Bounce.java	0	0	•	•
	2 files	0	0		

Figure 18. Project window - after adding file

Debugging a MIDlet

The debug agent is a Motorola SDK Component that works with the IDE and allows you to debug your J2ME applications.

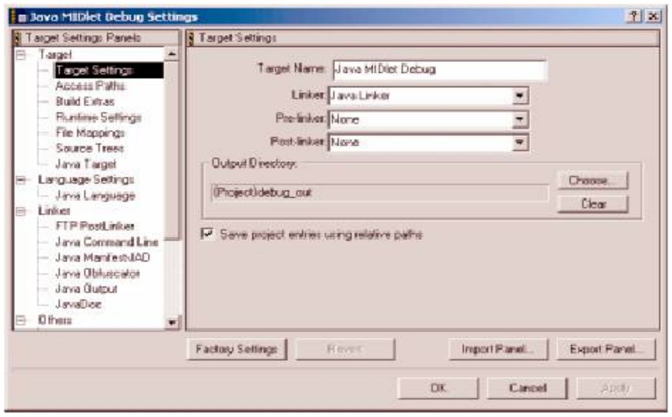
Setting Target Debugger Settings

To set up your debug environment for bouncetest:

1. Choose **Edit | Java MIDlet Debug Settings** The **Settings** window appears. Use the Settings window to set options that apply to a specific build target. In this case, that is the **Java MIDlet Debug** build target, as represented by the window title, **Java MIDlet Debug** Settings.

2. Select **Target Settings** in the **Target Settings Panels** list (Figure 19) For this example, use the default options as follows:

- **Target Name:** Java MIDlet Debug
- **Linker:** Java Linker
- **Pre-linker:** none
- **Post-linker:** none
- **Output Directory:** {Project}debug_out
- Save project entries using relative paths: selected


Java MIDlet Debug Settings

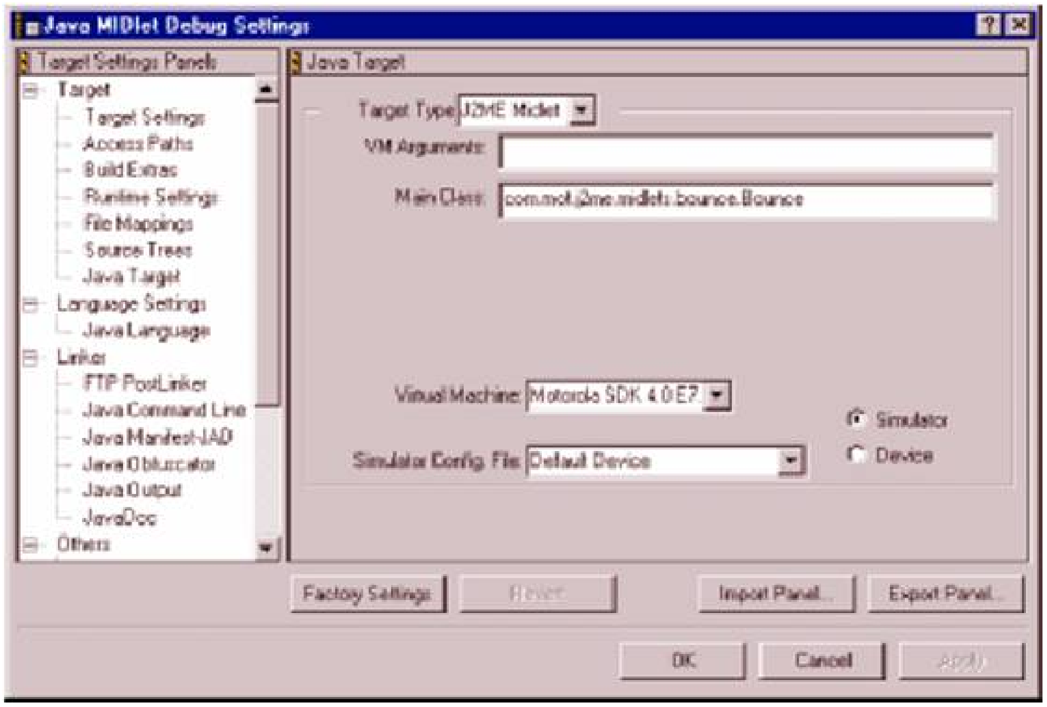
Target Settings

Target Name: Java MIDlet Debug
Linker: Java Linker
Pre-linker: None
Post-linker: None
Output Directory: {Project}debug_out
✓ Save project entries using relative paths

Figure 19. Target Settings panel option settings

3. Select **Java Target** in the **Target Setting Panels** list (Figure 20) For this example, set the options as follows:

- **Target Type:** J2ME Midlet
- **VM Arguments:** empty
- **Main Class:** com.mot.j2me.midlets.bounce.Bounce
- **Virtual Machine:** Motorola SDK v5.2.1 EA.3
- **Simulator:** selected
- **Simulator Config. File:** Default Device



Target Type: J2ME Midlet
VM Arguments:
Main Class: com.mot.j2me.midlets.bounce.Bounce
Virtual Machine: Motorola SDK 4.0 E7
Simulator Config File: Default Device
[✓ Simulator] Device

Figure 20. Java Target Settings panel option settings

4. Click **OK** The IDE saves the new options.

5. Select **Project | Make** to compile the MIDlet The IDE compiles the project.

6. Select **Project | Debug** to debug the MIDlet After a few seconds, the debug agent and the emulator appears. You can now modify the source code using the debug agent.

For detailed instructions on using the debugging interface, refer to the CodeWarrior documentation. See Working with the Emulator in this guide for instructions on how to use the emulator.

Running a MIDlet on the Emulator

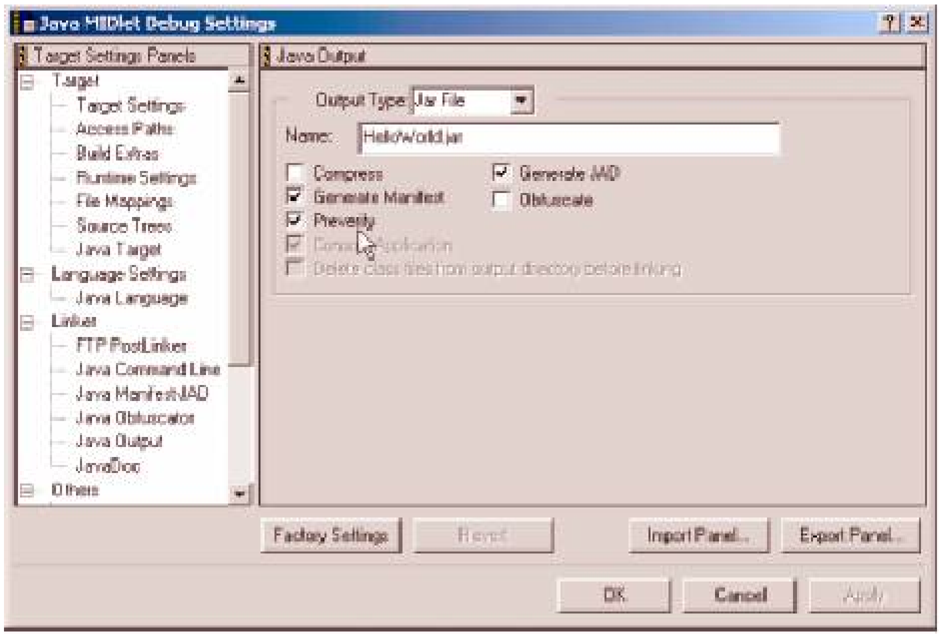
All classfiles running on the Motorola SDK for J2ME platform must be preverified prior to running. The CodeWarrior Wireless Studio tools do this automatically when you create a new project using the MIDlet stationery.

Previous versions of CodeWarrior development tools used the Pre-linker setting in the Target Settings panel to activate preverification. If you converted a project that used this setting, open the Target Settings panel and change the Pre-linker setting to None.

For more information on preverification, refer to Appendix C: Bytecode Verifier.

Preverifying a MIDlet

To run a MIDlet on the emulator, preverification must be done.



Output Type: Jar File

Name: HelloWorld.jar

- [ ] Compress
- [✓] Generate Manifest
- [✓] Preverify
- [✓] Compress Application
- [ ] Delete class files from output directory before linking
- [✓] Generate JAD
- [ ] Obfuscate

Figure 21. Setting the Preverifiy option

To enable Preverification of a project:

1. *Click **Edit | Java MIDlet Debug Settings***
2. *Select **Java Output** in the **Target Settings Panels** list*
3. *Enable the **Preverify** option (Figure 21)*
4. *Click **OK***
5. *Select **Project | Make** (The IDE compiles the MIDlet).*
6. *Select **Project | Run** to run the MIDlet (The IDE launches the MIDlet using the emulator).*

Troubleshooting Tips

Here are some answers to common troubleshooting questions.

1. *When I try to launch the emulator, the command window appears, then immediately disappears, and the emulator never launches. Can this be fixed?*

   The CodeWarrior IDE is unable to find the emulator. The most common cause is a corrupt `jvmdb.xml` file. The `jvmdb.xml` file contains information about installed virtual machines, as well as their supported devices, and the IDE uses this data to create the Java Target panel's Virtual Machine list settings. To correct this problem, exit the IDE if it's running, then locate and throw away the jvmdb.xml file. On Windows 2000, it is located in `C:\Document and Settings\user_name\My Documents\Metrowerks\` folder. The next time the CodeWarrior IDE launches, it recreates the file. Then, if you're using the Motorola SDK v5.2.1 for J2ME Platform, run `register2.bat` to repopulate the file with VMs and device information. See Merging the Motorola SDK v5.2.1 for J2ME Platform with CodeWarrior Wireless Studio for details.

Appendix B: Sample MIDlets

The Motorola SDK Components also provides sample MIDlets you can run to see examples of J2ME programming capabilities. The MIDlet classfiles and source files for these samples are in `<MOTOROLA_SDK_HOME>\demo\midlets\com\mot\j2me\midlets`.

Sample MIDlet Descriptions

This section describes each of the sample MIDlets included with this version of the Motorola SDK for the J2ME Platform.

AcronymSearcher

The sample midlet Acronym Searcher, through a web server, allows the user to consult the meaning of an acronym.

See the file README.txt inside this midlet directory:
`<MOTOROLA_SDK_HOME>\demo\com\mot\j2me\midlets\AcronymSearcher`

AlertTest

AlertTest demonstrates how to use timed and modal alerts.

To see a timed alert:

- Click the left soft key

OR

- Press Page Up on the keyboard.

After two seconds, the emulator will return to the AlertTest menu.

To see a modal alert:

- Click the right soft key

OR

- Press Page Down on the keyboard.

Bounce

Bounce displays several squares bouncing around inside the screen.

Bluetooth Tic Tac Toe

This application consists of the game Tic Tac Toe with the capability of being played by 2 players simultaneously, through two different instances of the Motorola SDK for J2ME connected on a simulated Bluetooth environment running on the same machine. This Sample Midlet demonstrates how to use the JSR 82 (Java APIs for Bluetooth).

See the file README.txt inside this midlet directory:
`<MOTOROLA_SDK_HOME>\demo\com\mot\j2me\midlets\bluetoothtictactoe`

ChangeDate

ChangeDate demonstrates how to use Calendars.

To change the date:

- Press Enter on the keyboard

OR

- Click the Select key on the emulator.

To move around calendar:

- Press the arrow keys on the keyboard

OR

- Click the arrow keys on the emulator.

To view new select date:

- Click Save on the emulator screen

ChoiceGroupTest

ChoiceGroupTest demonstrates radio button groups and checkbox groups.

To move between the radio buttons and checkboxes:

- Press the up and down arrow keys on the keyboard
- Click the up and down arrow keys on the emulator
- Select with the mouse

To select a radio button:

- Press Enter
- Select with the mouse

You can only select one radio button at a time.

To select a checkbox:

- Press Enter
- Select with the mouse

You can select several checkboxes at one time.

CommandTest

CommandTest demonstrates how to use Commands and create menu items.

To move around the menu items:

- Press the arrow keys on the keyboard

OR

- Click the arrow keys on the emulator.

To select a menu item:

- Press Enter on the keyboard

OR

- Click the Select key on the emulator.

To go back to main menu:

- Click Back on the emulator screen.

CoolFLTest

CoolFLTest demonstrates how to use FunLights.

To enter sleep value:

- Click the number keys on the emulator

OR

- Press the number keys on the keyboard.

To see funlights working:

- Click Test on the emulator screen.

CoolFLTest will work only on handsets that supports FunLight API.

DateFieldTest

DateFieldTest demonstrates how to use DateFields in DATE, TIME, and DATE_TIME mode.

To move around the date and time fields:

- Press the arrow keys on the keyboard

OR

- Click the arrow keys on the emulator.

To enter numeric values:

- Click the number keys on the emulator

OR

- Press the number keys on the keyboard.

DateFieldTest does not recognize the numeric keypad except for the asterisk (*) key, which allows you to delete existing characters.

FontDemo

FontDemo demonstrates different combinations of fonts, faces, styles, and sizes.

To see the font examples:

- Click SELECT on the emulator screen to select a font type from the "Choose Font Face" menu.

Note: The Emulator A1 has only one available font type. No styles/sizes can be applied for font selection.

FormTest

FormTest demonstrates having multiple items on a form. The methods for navigating and selecting form items are identical to the methods used in ChoiceGroupTest. Notice, however, that the fields are only the first of eight form elements. Press the up and down arrow keys on the keyboard, or click the up and down arrow keys on the emulator to scroll up and down the form. The fields include text fields, radio button groups, checkbox groups, and interactive gauges.

GaugeTest

GaugeTest demonstrates how to use an interactive gauge.

To raise the gauge:

- Press the right arrow key on the keyboard
- Click the right arrow key on the emulator

- Press the left arrow key on the keyboard
- Click the left arrow key on the emulator

GraphicsDemo

GraphicsDemo demonstrates how to use various Graphics primitives.

To see the graphics examples:

- Click the left and right arrow keys on the emulator

OR

- Press the left and right arrow keys on the keyboard

The following routines are then shown:

- fillRect()
- drawRect()
- drawArc()
- fillArc()
- fillRoundRect()

GraphicsTest

GraphicsTest demonstrates how to use Graphics.

To move around the graphics:

- Press the arrow keys on the keyboard

OR

- Click the arrow keys on the emulator.

To change a graphic style:

- Click Solid on the emulator screen

OR

- Click Doted on the emulator screen.

GuiTests

GuiTests demonstrates how to use various GUI components.

To move around the menu items:

- Press the arrow keys on the keyboard

OR

- Click the arrow keys on the emulator.

To select a menu item:

- Press Enter on the keyboard

OR

- Click the Select key on the emulator.

HelloWorld

HelloWorld displays a "Hello World" message on the emulator screen.

ImageTest

ImageTest demonstrates how different types of images (colors, fonts, shapes) look on the emulated device screen.

To view all available images:

- Click the scroll arrows at the bottom of the emulator screen.

KeyEventsTest

KeyEventsTest demonstrates how to perform low-level key event handling. KeyEventsTest displays "Press a key!" when it starts. Press a key on the keyboard or click a key on the emulator to display the key's applicable values (key code, action, and key name) on the emulator screen.

KJavaTelephonyTest

KJavaTelephonyTest simulates a phone call.

To enter phone number:

- Click the number keys on the emulator

OR

- Press the number keys on the keyboard.

To simulate a phone call:

- Press Enter on the keyboard

OR

- Click the Select key on the emulator.

MobilePiano

The sample midlet "Mobile Piano" allows the user to play an emulated piano, configure volume and pitch, and play a tutorial video that demonstrates how to play a small song. To develop this midlet the Mobile Media API is used.

See the file README.txt inside this midlet directory:
`<MOTOROLA_SDK_HOME>\demo\com\mot\j2me\midlets\MobilePiano\`

Paddleball

PaddleBall is similar to the classic arcade game with the ball and paddle.

To start the game:

- Click START on the emulator screen

OR

- Click the right soft key on the emulator

OR

- Press the Page Down key on the keyboard

To play the game:

- Click the right and left arrow keys on the emulator

OR

- Press the left and right arrow keys on your keyboard to move the paddle

Personal Organizer

The sample midlet Personal Organizer allows the user to register contacts and events (such as meeting and friend's birthday) on the handset. The user can also send the event contents through SMS to other devices. To develop this midlet the Personal Information Management API and Wireless Messaging API are used.

Photo Blog Example

The sample midlet Photo Blog Example allows the user to take snapshot, include a brief description about it and save them on the emulator. The user can delete or visualize the picture and its comments, or also insert more comments about it. To develop this midlet the FileConnection API

(JSR 75) and Mobile Media API (JSR 135) are used.

PushRegistryExample

See the file README.txt inside this midlet directory:
`<MOTOROLA_SDK_HOME>\demo\com\mot\j2me\midlets\PushRegistryExample`

RecordStoreDemo

RecordStoreDemo demonstrates some of the basic functionality of the Record Management System (RMS) classes. Unlike FontDemo and GraphicsDemo, the output from RecordStoreDemo appears in the command line window, not the emulator screen. However, an empty emulator screen is also displayed.

To demonstrate RMS functionality, RecordStoreDemo does the following:

- Creates a recordStore of constants
- Inserts ten new integer records
- Inserts some string records
- Builds an enumeration that indexes through all records (Notice that ID #12 was deleted).
- Builds a RecordEnumeration that filters out all records except integer records
- Lists the remaining records in reverse order

Scene3D

See the file README.txt inside this midlet directory:
`<MOTOROLA_SDK_HOME>\demo\com\mot\j2me\midlets\Scene3D`

ShortMessage

The sample midlet Short Message Example offers short text message transmission service to and from a mobile phone in a GSM network. To develop this midlet the Wireless Messaging API (JSR 120) is used.

See the file README.txt inside this midlet directory:
`<MOTOROLA_SDK_HOME>\demo\com\mot\j2me\midlets\ShortMsgEx`

SoundTest

SounsTest demonstrates different sound types.

To move around the sound items:

- Press the arrow keys on the keyboard

OR

- Click the arrow keys on the emulator.

To select a sound item:

- Press Enter on the keyboard

OR

- Click the Select key on the emulator.

Important Note: In order to run this midlet successfully, the following sounds must be enabled in the Windows:

- Question ("Confirmation")
- Asterisk ("Alarm")
- Critical Stop ("Error")
- Exclamation ("Warning")
- Default Beep ("Info")

TextBoxTest

TextBoxTest demonstrates how to edit a text box screen.

To type letters in the text box:

- Click the desired key on the emulator keypad. For example, to type the letter b, click the 2 key twice.

OR

- Press the appropriate number key on your keyboard (not the numeric keypad) until the desired letter or number appears.

When the letter you want appears, click OK on the emulator screen, press Enter on the keyboard, or click the phone key on the emulator.

To move around the text box screen:

- Click the left and right arrow keys on the emulator, or press the left and right arrow keys on the keyboard.

To delete characters:

- Click the asterisk (*) key on the emulator, or press the asterisk key on the numeric keypad.

To insert spaces between words:

- Click the pound sign (#) key on the emulator.

To select a character [period (.), comma (,), colon (:), at symbol (@), dollar sign ($), exclamation point (!)]:

- Click the 1 key on the emulator, or press the 1 key on the keyboard repeatedly.

TextFieldTest

TextFieldTest demonstrates how to add text to a text field in a form. The methods for adding and deleting characters, as well as for navigating from text field to text field, are identical to those methods found in TextBoxTest.

TickerTest

TickerTest demonstrates how to use tickers. To navigate the three screens in this MIDlet:

- Click NEXT and PREV on the emulator screen, or press the Page Up and Page Down keys.

UDPSend and UDPReceive

UDP Send and UDP Receive demonstrates networking through the User Datagram Protocol (UDP).

Once the program launches, UDPReceive waits for a datagram to be sent through a given port. When the datagram is received, a message is displayed to both the emulator and command line windows

Next, open a second command line window and enter:

runEmul com.mot.j2me.midlets.tutorials.UDPSend

While not necessary, you may choose to do this step from another workstation on the same network as the workstation with the first command line window. This further illustrates the capabilities of UDP.

UDPSend sends a datagram with the message Hello UDP Networking to the local host. To run UDPReceive on a different machine than the one where UDPSend is located, you must edit the file UDPSend.java and replace localhost with the host name of the machine running UDPReceive. The localhost may be entered in one of the following two formats:

- Host name (i.e., jijo.risc.sps.mot.com)
- IP address (i.e., 205.16.177.105)

Wireless Multimedia Messaging Example

The sample midlet Wireless Multimedia Messaging Example (WMME) offers MMS message (audio, image and text) transmission service simulating communication between the mobile device and outside server. To develop this midlet the Wireless Messaging API 2.0 (JSR 205) and Mobile Media API (JSR 135) are used.

See the file README.txt inside this midlet directory:
`<MOTOROLA_SDK_HOME>\demo\com\mot\j2me\midlets\wirmultimsgex\`

Appendix C: Bytecode Verifier

The bytecode verifier is a tool that ensures that Java bytecodes and other items stored in Java classfiles do not contain references to invalid memory locations or memory areas outside the Java object memory (known as Java heap). The classfile verifier ensures that classfiles loaded into the virtual machine do not perform operations that are not allowed by the Java Virtual Machine Specification.

Class files must be verified before they are run on the KVM to ensure that the classfiles do not perform any illegal operations. Verification of classfiles is done in two steps: offdevice

preverification and in-device verification. Preverification of a classfile takes place somewhere other than the KVM such as on a server or on a developer's workstation. The runtime verification (or in-device verification) actually occurs within the device carrying the virtual machine using the special attribute provided by the preverifier. Preverification is transparent because it is written into the compiler.

The J2ME bytecode verifier is smaller than the bytecode verifier for the standard Java 2 SDK. The J2ME bytecode verifier verifies each bytecode linearly - that is, it does not require complex data flow algorithms to check the accuracy of the bytecodes. The preverifier runs over the classfiles on the file system and adds an attribute later used by the bytecode verifier to actually verify the class at runtime. Preprocessed classfiles containing the attribute from the preverifier are approximately 5% bigger than the original, unmodified classfiles.

Appendix D: Using Foreign Fonts in Motorola Emulators

This appendix describes how to manually add Unicode fonts to access them on a Motorola phone. We recommend that you use the Launchpad application to automate this process, but it can be done manually if desired. Read on to find out how.

Language Requirements

To support the use of foreign language characters you must:

- Enable Java™ to support foreign language fonts
- Install the fonts that contain the characters you want to display
- Modify the appropriate file for the emulator

Enable Java™ to Support Foreign Language Fonts

The Java™ Runtime Environment (JRE™) uses the font.properties file to determine which fonts are available and usable. You modify this file to add additional fonts to the supported list. For more information on how to accomplish this, see Sun's documentation "Adding Fonts to the Java Runtime" located at the URL:
`http://java.sun.com/products/jdk/1.1/docs/guide/intl/fontprop.html`

Supported Foreign Fonts

In order for the Motorola emulators to use a foreign font, it must be installed correctly on your PC. The following table lists the Asian fonts currently recognized by Motorola phones as well as their Unicode and Window's filenames. Use it as a guide to locate and install the correct Asian font for your development efforts.

Supported Asian languages

Language	**Java Typeface Name**	**Font File Name**
Chinese (Simplified)	\u5b8b\u4f53	SIMSUN.TTC
Chinese (Traditional)	\u7d30\u660e\u9ad4	MINGLIU.TTC

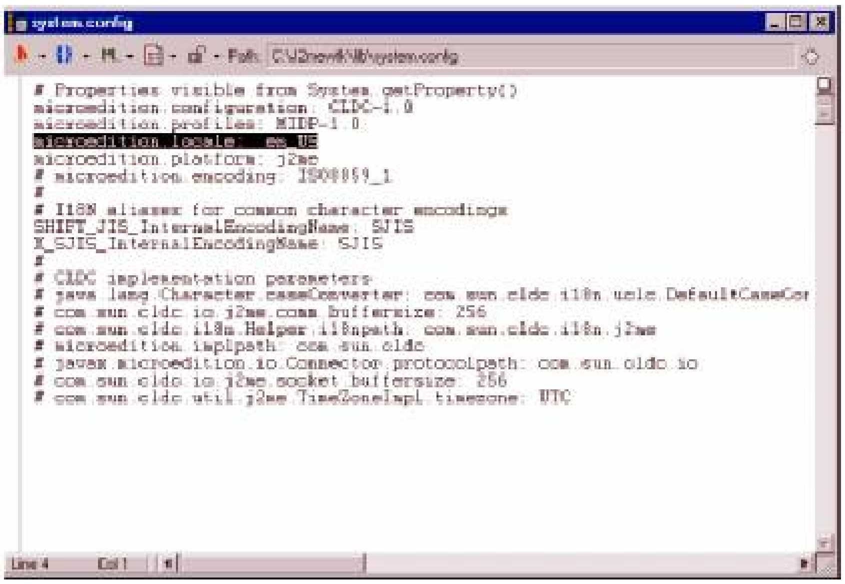
Modify Motorola Emulator Files

The final step in the process is to manually edit the appropriate files for the emulator that supports the device you are writing software for.

If MIDlets are not written to properly display Unicode characters, then ASCII characters are used instead.

Emulator M.1

The M.1 emulator uses settings defined by the MIDP specification to control the display fonts. This information is stored in the file system.config, located in the directory <`Wireless Toolkit root directory>\lib`. The entry microedition.locale is used to select the font used, as shown in Figure 22.



```
# Properties visible from System.getProperty()
microedition.configuration: CLDC-1.0
microedition.profiles: MIDP-1.0
microedition.locale:  en_US
microedition.platform: j2me
# microedition.encoding: ISO8859_1
#
# I18N aliases for common character encodings
SHIFT_JIS_InternalEncodingName: SJIS
X_SJIS_InternalEncodingName: SJIS
#
# CLDC implementation parameters
# java.lang.Character.caseConverter: com.sun.cldc.i18n.uclc.DefaultCaseCon
# com.sun.cldc.io.j2me.comm.buffersize: 256
# com.sun.cldc.i18n.Helper.i18npath: com.sun.cldc.i18n.j2me
# microedition.implpath: com.sun.cldc
# javax.microedition.io.Connector.protocolpath: com.sun.cldc.io
# com.sun.cldc.io.j2me.socket.buffersize: 256
# com.sun.cldc.util.j2me.TimeZoneImpl.timezone: UTC
```

Figure 22. Sample system.config file

To change the font used by the Emulator M.1

1. *Find the system.config file*

   This file is located in the <`Wireless Toolkit root directory>\lib` directory.
2. *Rename system.config file to system-OLD.config Save the old system.config file for later restoration of the old font, or in case you should make a mistake.*
3. *Open this file with an editor, and immediately save it as system.config*
4. *Locate the string microedition.locale:*
5. *Change the string en_US to ja_JP to display Japanese fonts*
6. *Save the file*

That's it. The next time you invoke the Emulator M.1, the device displays the new language.

Appendix E: Open Classes

This appendix provides descriptions of the Open Class (OC) application programming interfaces (API) available in the Motorola SDK for the J2ME™ platform release

Open Class Overview

For detailed information on a specific OC, see the JavaDoc for that API.

JSR 30 - CLDC 1.0 API

Connected Limited Device Configuration 1.0 API.

JSR 75 - PDA Optional Packages for the J2ME™ Platform

Two APIs compose this package: PIM API and FileConnection API. The PIM package defines APIs to access Personal Information Management (PIM) data. The FileConnection package describes file system access support based on the Generic Connection Framework.

Note: currently there are two implementations of file system access: Motorola FileSystem API and JSR 75 FileConnection. However, they can not be used simultaneously. i.e., only one of them can be used in a MIDlet at a given time

JSR 82 - Java APIs for Bluetooth

Bluetooth is an important emerging standard for wireless integration of small devices. The specification standardizes a set of Java APIs to allow Java-enabled devices to integrate into a Bluetooth environment.

JSR 118 - MIDP 2.0 API

Mobile Information Device Profile 2.0 API.

JSR 120 - Wireless Messaging API

It defines an API which allows applications to send and receive wireless messages. It also includes the platform networking interfaces which have been modified for use on platforms that support message connections.

JSR 135 - Mobile Media API

This document, Mobile Media API (JSR-135) Specification, defines the Multimedia API for the Java TM 2 Platform, Micro Edition (J2METM). The audience for this document is the public Java community reviewing this specification and the Java Community Process (JCP) expert group defining this specification, implementors of the Multimedia API, and application developers targeting the J2ME platform.

JSR 139 - CLDC 1.1 API

Connected Limited Device Configuration 1.1 API.

JSR 184 - Mobile 3D Graphics API

This specification defines the Mobile 3D Graphics API (M3G) for J2ME. It defines an API for rendering three-dimensional (3D) graphics at interactive frame rates, including a scene graph structure and a corresponding file format for efficient management and deployment of 3D content.

JSR 205 - Wireless Messaging API 2.0

This JSR will extend and enhance the "Wireless Messaging 2.0 API" (JSR-205).

Motorola Fun Light API

The Fun Light API provides developers a way to enhance applications like games, with access to several light features including: on/off control, light area control, intensity, and color, when supported by a device.

Motorola Get URL form Flex API

The Motorola Get URL from Flex API allows Java applications to read and display the URL stored in flex files.

Motorola Vibrate and Backlight API

The Vibrator and Backlight interfaces are Java classes that extend the standard J2ME platform, providing methods to control any vibrator or screen backlight and intensity on the device. Only devices that include KJava and the Vibrator and Backlight functionality can respond to these methods.

Appendix F: Unified Emulator Interface

The SDK development team is already working on the implementation of the complex code changes that are required to allow the KVM to support UEI commands. The tool will be fully UEI compliant in the near future. We will be sure to broadcast a release update and notify our developer community as soon as the tool is available.

Currently here are the UEI compliant statuses of the KVMs inside our tool:

Emulator A.1 - Supports UEI command-line commands, but it only supports on the default phone skin

Emulator A.3 - Does not support UEI commands.

Emulator A.4 - Does not support UEI commands.

Emulator M.1 - Does not support UEI commands.

Emulator M.3 - Supports UEI commands.