Simulating LBS using Samsung SDK

Version 0.9, Draft

INFO

COPYRIGHT

Samsung Electronics Co. Ltd.
This material is copyrighted by Samsung Electronics. Any unauthorized reproductions, use or disclosure of this material, or any part thereof, is strictly prohibited and is a violation under the Copyright Law Samsung Electronics reserves the right to make changes in specifications at any time and without notice. The information furnished by Samsung Electronics in this material is believed to be accurate and reliable, but is not warranted true in all cases.

Trademarks and Service Marks

The Samsung Logo is the trademark of Samsung Electronics. Java is the trademark of Sun Microsystems.
All other company and product names may be trademarks of the respective companies with which they are associated.

About This Document

This document demonstrates how to test location based applications using the built in test facility provided in Samsung SDK.

Scope:

This document is for user who has knowledge of Java ME and needs a brief introduction on how to test location based applications using Samsung SDK.

Document History:

Date	Version	Comment
05/10/09	0.9	Draft

Abbreviations:

JSR	Java Specification Request
API	Application Programming Interface
CLDC	Connected Limited Device Configuration
BTS	Base Transceiver Station
GPS	Global Positioning System
EEG	External Event Generator

Table of Contents

Introduction 5
External Event Generator 5
Running Scripts: 12
Orientation: 15
LocationMidlet Example 17
Orientation Midlet Example 24

List of Figures

Figure 1: Samsung SDK 6
Figure 2: Selecting External Events 7
Figure 3: External Event Generator Window 8
Figure 4: Adding Location Information 10
Figure 5: Location MIDlet Emulator 11
Figure 6: Location MIDlet Output 12
Figure 7: Script Playing 14
Figure 8: Setting Orientation Values 15
Figure 9: Orientation MIDlet 16
Figure 10: Orientation MIDlet output 17

SAMSUNG mobile innovator

Introduction

Location API is an Optional Package *javax.microedition.location* that provides access to location based information. Location API provides a standard for developers to write mobile location-based applications. Location API gives information about the device's present physical location. It can be used with many Java ME profiles. The minimum platform is CLDC and targets are low memory devices.

Samsung SDK supports Location API and provides a built in facility to test location based application. Samsung SDK simulates the device providing location, orientation and landmark information to the application.

External Event Generator

Samsung SDK provides External Event Generator (EEG) Window for testing location applications. External Event Generator Window provides developer to test the location and orientation of the device. It also provides a script window through which script can be run.

EEG window is attached with the emulator and can be activated by selecting the External Events option from the MIDlet menu option located at the top of the emulator

To help demonstrate the support of Location API, Location MIDlet and Orientation MIDlet code example are provided at the end.

Once you successfully load the Location MIDlet in Samsung SDK, select specific Samsung Emulator that supports Location API such as a867 emulator. For demonstration we are using a867 emulator as this emulator supports Location API.

Following Steps shows how to use External Event Generator to test location Application.

Step 1: Run Location MIDlet by clicking on the Run option as shown in Figure 1.

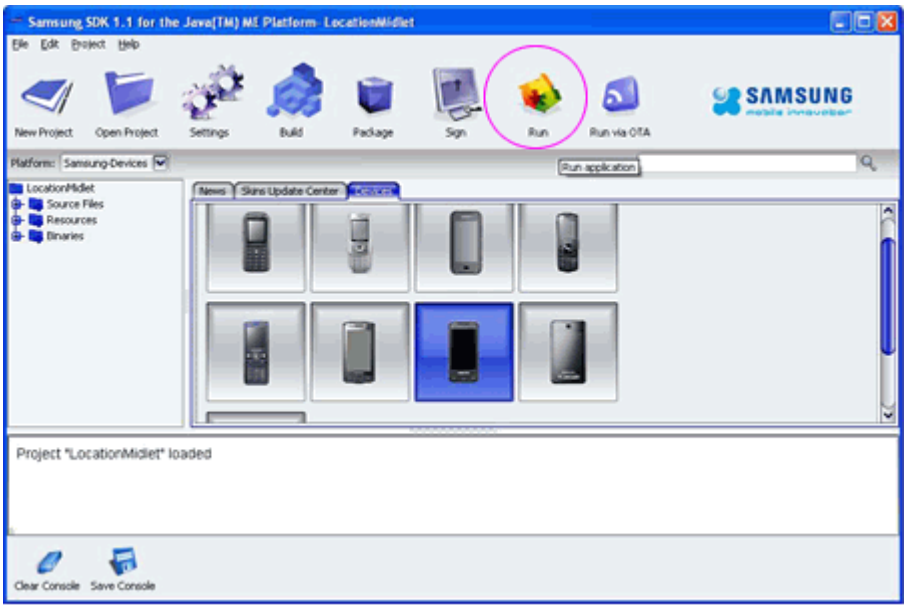

Figure 1: Samsung SDK

Step 2: Now when the emulator starts, you will need the External Event Generator (EEG) up to provide location data to the MIDlet. The EEG simulates data as if it was coming from the real device. On the emulator window, select External Events from the MIDlet menu option at the top of the emulator as shown in Figure 2.

Figure 2: Selecting External Events

This will cause the EEG window to appear. Select Location Tab as shown in Figure 3.

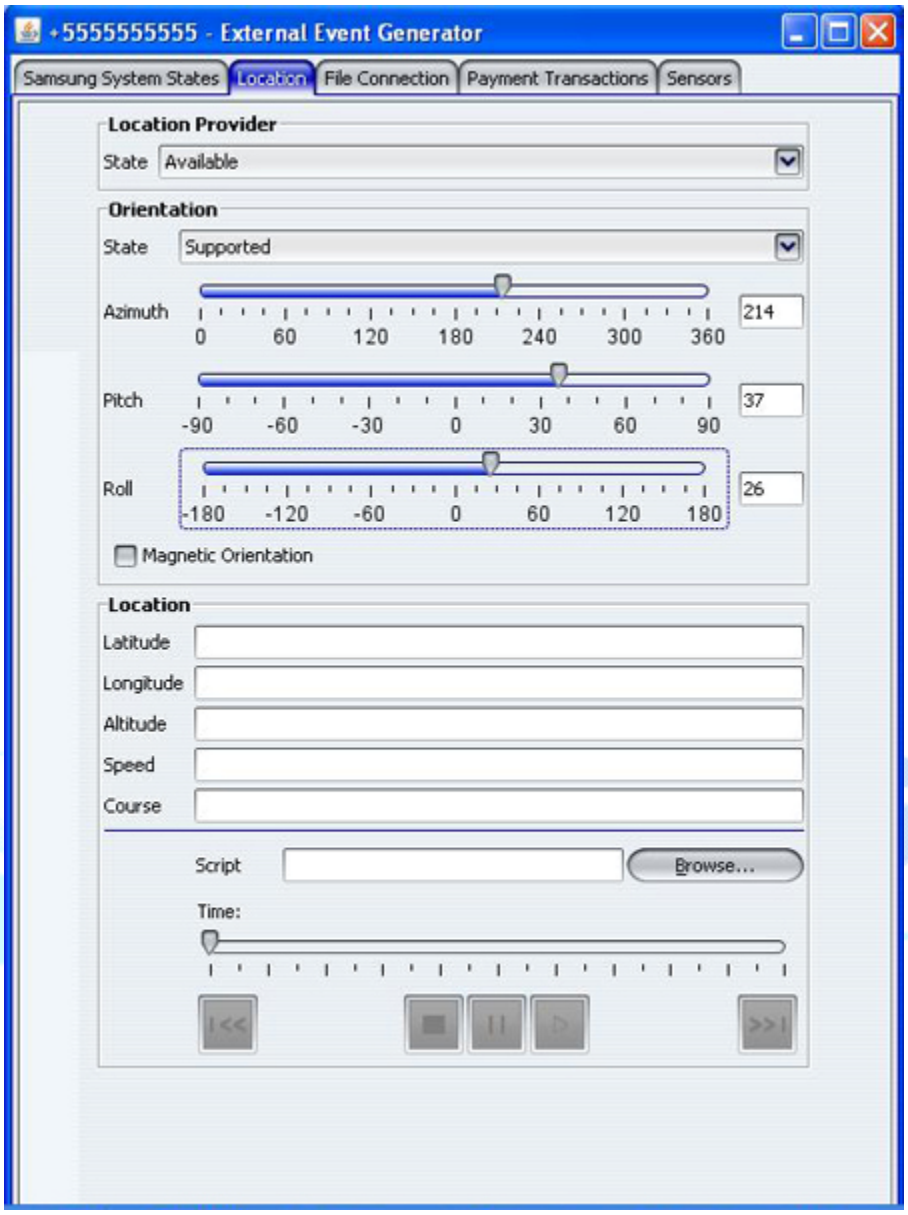

Figure 3: External Event Generator Window

Step 3: EEG window shows three sections Location Provider, Orientation and Location. One can set the state of Location Provider using the drop down box. The states that can be set are *Available*, *Temporary Available* and *Out Of Service*.

Similarly one can set the state of Orientation section using the drop down box provided. The states that can be set for orientation are *Unsupported*, *Available* and *Unavailable*. Orientation section also has **Azimuth**, **Pitch** and **Roll** slider through which one can set the values that would be reflected in the device or alternatively user can entered the desired value directly in the box provided beside each slider.

Similarly Location section has five different parts namely **Latitude**, **Longitude**, **Altitude**, **Speed** and **Course** to set the Location Information. Alternatively it also provides **script box** through which one can run the script that emulates the location information.

Information on how to use Orientation and how to use Script is explained later.

Set the desired location information in the Latitude, Longitude, Altitude, Speed and Course box as shown in Figure 4.

SAMSUNG mobile innovator

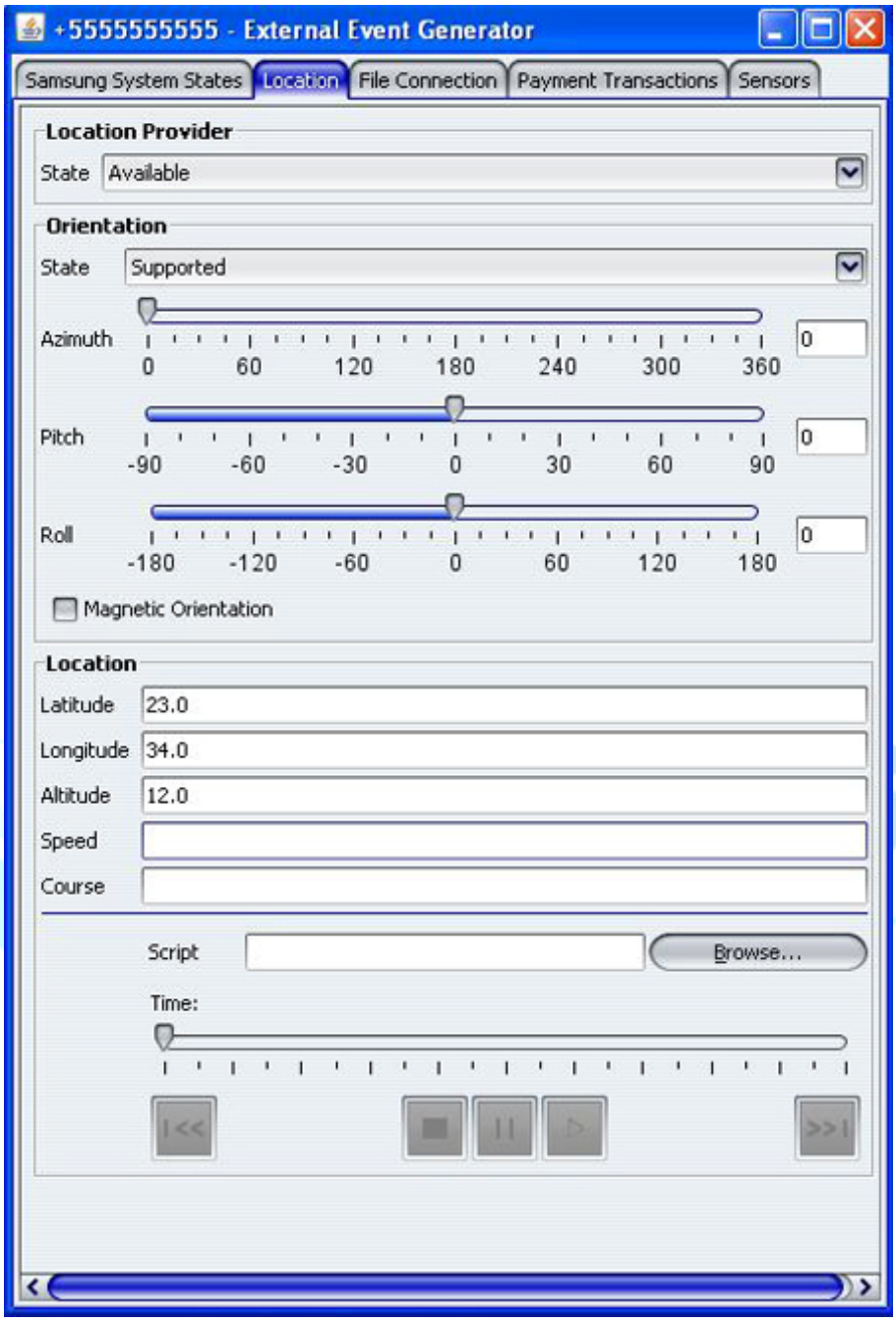

Figure 4: Adding Location Information

Step 4: Once the desired values are set, switch back to the emulator screen. Click on the locate command as shown in the Figure 5.

Figure 5: Location MIDlet Emulator

Clicking on Locate Command causes the Location MIDlet to collect the values from the EEG window creating the illusion as of coming from real device. The output is shown in Figure 6.

Figure 6: Location MIDlet Output

Running Scripts:

The EEG window can also be used to run a script. Script provides location information that changes the simulated geo-information provided to the emulator.

Scripts can be useful for MIDlets that have location listeners implemented. Location listener reacts to changes in location.

The EEG scripts are actually quite easy to write as they are written in XML and consist of a set of waypoints that are read and fed to the emulator, as you require. A small script example is provided below:

```
<waypoints>

<waypoint time="1200" latitude="10.238776738944633" longitude="50.19885522776621" altitude="100" />
```

```
<waypoint time="1500" latitude="14.238776738944633" longitude="54.19885522776621" altitude="200" />

<waypoint time="18500" latitude="18.238776738944633" longitude="58.19885522776621" altitude="300" />

<waypoint time="2100" latitude="22.238776738944633" longitude="62.19885522776621" altitude="400" />

</waypoints>
```

Samsung SDK provides you with a demo application called CityGuide that demonstrates the use of a location script with the EEG. CityGuide MIDlet has a script file "citywalk.xml". Click on Browse button and go to CityGuide demo application and select the *citywalk.xml* file as shown in Figure 7.

SAMSUNG mobile innovator

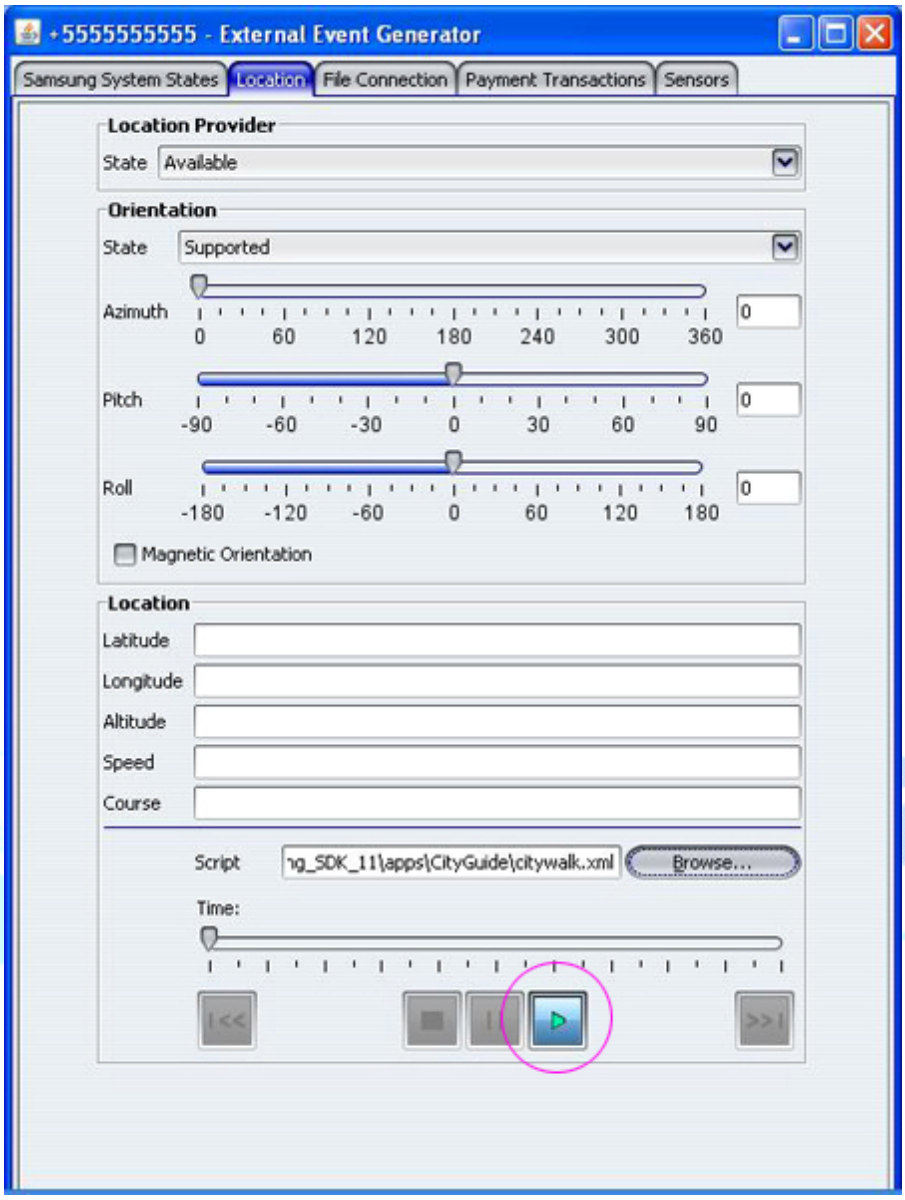

Figure 7: Script Playing

Once selected the script Player gets enabled. Simply press the play button to run the script. Switch back to emulator screen to see the output. Location information from the script file would continuously be provided to the CityGuide MIDlet.

Orientation:

To test the Orientation, Run the Orientation MIDlet provided at the end of this article. Follow the steps 1 and 2 as explained previously in Location MIDlet.

Step 3: Entered the desired values in Azimuth, Pitch and Roll box or slide the slider to set the desired values of Azimuth, Pitch and Roll as shown in Figure 8.

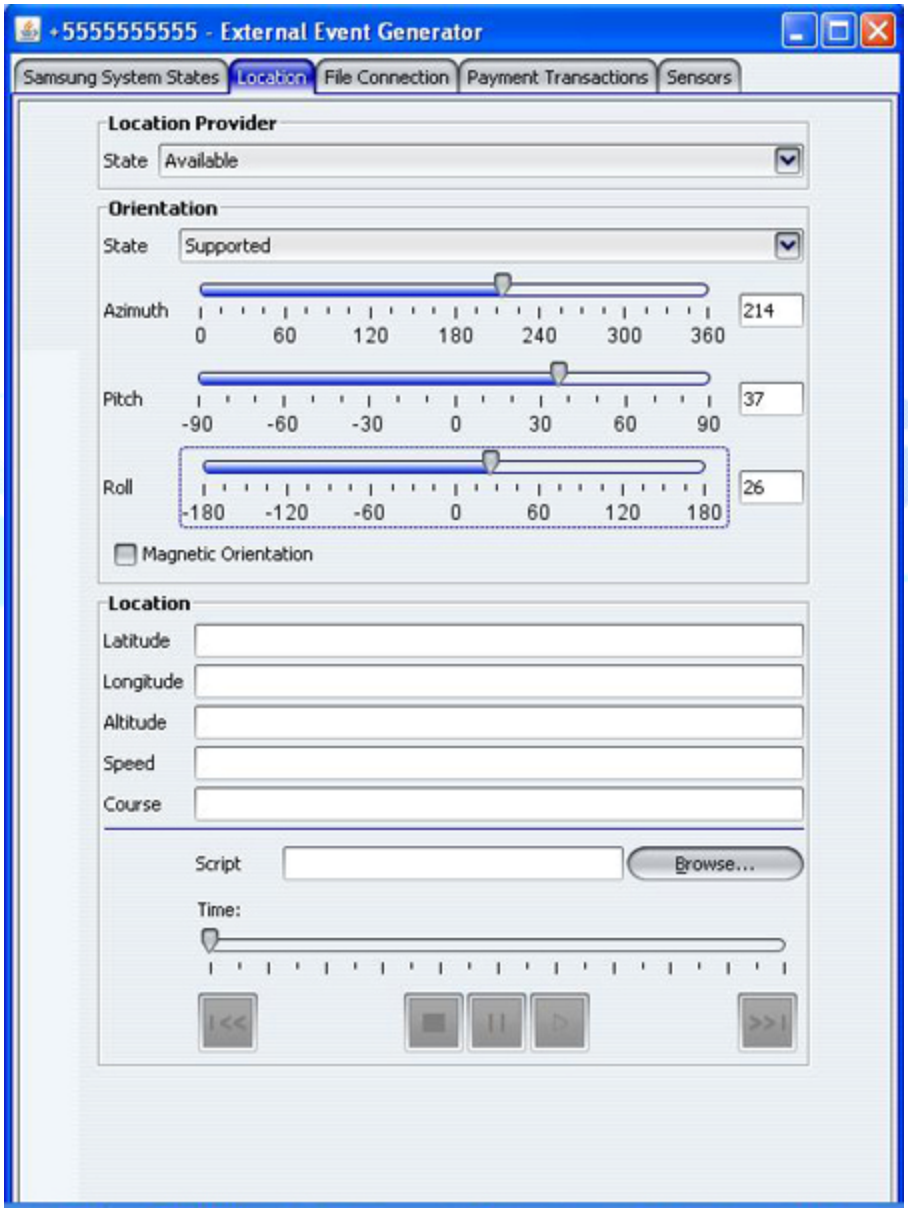

Figure 8: Setting Orientation Values

Step 4: Once the desired values are set, switch back to the emulator screen. Click on the Result command as shown in the Figure 9.

Figure 9: Orientation MIDlet

Clicking on Result Command causes the Orientation MIDlet to collect the values from the EEG window creating the illusion as of coming from real device. The output is shown in Figure 10.

Figure 10: Orientation MIDlet output

LocationMidlet Example

```
import javax.microedition.lcdui.Command;

import javax.microedition.lcdui.CommandListener;

import javax.microedition.lcdui.Display;

import javax.microedition.lcdui.Displayable;

import javax.microedition.lcdui.Form;

import javax.microedition.location.Criteria;

import javax.microedition.location.Location;
```

```
import javax.microedition.location.LocationException;

import javax.microedition.location.LocationProvider;

import javax.microedition.location.QualifiedCoordinates;

import javax.microedition.midlet.MIDlet;

public class LocationMidlet extends MIDlet implements CommandListener,Runnable{

   /* Define String Constants */

   private final String locate_String = "Locate";

   private final String exit_String = "Exit";

   private final String done_String = "Done";

   private final String locationForm_Title_String = "Location Demo";

   private final String locationInfo_String = "Press Locate Command to start “+

                                                “Application  Demo\n";

   private final String exitInfo_String = "Press Exit Command to exit Application  \n";

   /*LocationMidlet Display */

   private Display display;

   /*LocationMidlet Commands*/

   private Command cmd_Locate, cmd_Exit, cmd_Done;

   /*LocationMidlet Form*/

   private Form locationForm;



   /** location provider */

   private LocationProvider locationProvider = null;
```

```
private Location location;

private QualifiedCoordinates coordinates;

private Thread locationThread;

/*LocationMidlet Constructor*/

public LocationMidlet()    {

   /*Intialize Commands*/

   cmd_Locate = new Command(locate_String, Command.OK,1);

   cmd_Done   = new Command(done_String, Command.OK,1);

   cmd_Exit   = new Command(exit_String, Command.EXIT,2);

   /*Create Form*/

   locationForm = new Form(locationForm_Title_String);

   /*Appending location and Exit Information*/

   locationForm.append(locationInfo_String);

   locationForm.append(exitInfo_String);

   /*Adding commands to Location Form*/

   locationForm.addCommand(cmd_Exit);

   locationForm.addCommand(cmd_Locate);

   locationForm.setCommandListener(this);

   /*Intialize display*/

   display = Display.getDisplay(this);

   /*Create LocationProvider Instance*/

   locationProvider = null;
```

```
    createLocationProvider();
    if(locationProvider == null)      {
        displayError();
    }
}
public void startApp() {
    /*display Form*/
    display.setCurrent(locationForm);
}
public void pauseApp() {
    /*your code to handle interrupt*/
}
public void destroyApp(boolean unconditional) {
    /*your code before notifyDestroyed method*/
    notifyDestroyed();
}
private void exitMidlet()    {
    destroyApp(true);
}
private void displayError()    {
    locationForm.deleteAll();
    locationForm.append("Unable to create LocationProvider");
```

```
    locationForm.addCommand(cmd_Exit);
}
/*Initialize LocationProvider using Default Criteria*/
private void createLocationProvider() {
    if (locationProvider == null) {
        /*Constructs a Criteria object. All the fields are set to the default values*/
        Criteria criteria = new Criteria();
        try {
            locationProvider = LocationProvider.getInstance(criteria);
        } catch (LocationException le) {
            /*Unable to create Location Provider using this criteria*/
            le.printStackTrace();
        }
    }
}
private void createLocation()    {
    try {
        // Get a location fix for 30 seconds timeout
        location = locationProvider.getLocation(30);
    } catch (LocationException ex) {
        ex.printStackTrace();
    } catch (InterruptedException ex) {
```

```
        ex.printStackTrace();
    }
}
private void getLocationResults()    {
    coordinates = location.getQualifiedCoordinates();
}
private void displayLocationResults()    {
    locationForm.append("Altitude:"+coordinates.getAltitude() +"\n");
    locationForm.append("Latitude:"+coordinates.getLatitude() +"\n");
    locationForm.append("Longitude:"+coordinates.getLongitude() +"\n");
}
public void commandAction(Command cmd, Displayable disp) {
    if(cmd == cmd_Locate)       {
        /*Deleting previous data displayed*/
        locationForm.deleteAll();
        /*Removing exit and locate command*/
        locationForm.removeCommand(cmd_Exit);
        locationForm.removeCommand(cmd_Locate);
        /*Adding done command*/
        locationForm.addCommand(cmd_Done);
        /*Create Thread to avoid deadlock*/
        locationThread = new Thread(this);
```

```
        locationThread.start();
    }else
    if(cmd == cmd_Exit)       {
        exitMidlet();
    }else
    if(cmd == cmd_Done)       {
        locationForm.deleteAll();
        /*Remove done command*/
        locationForm.removeCommand(cmd_Done);
        /*Adding commands to Location Form*/
        locationForm.addCommand(cmd_Exit);
        locationForm.addCommand(cmd_Locate);
        /*Appending location and Exit Information*/
        locationForm.append(locationInfo_String);
        locationForm.append(exitInfo_String);
    }
}
public void run() {
    /*Get Location*/
    createLocation();
    /*Get Location Results*/
    getLocationResults();
```

```
        /*Display Results*/
        displayLocationResults();
    }
}
```

Orientation Midlet Example

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.location.Orientation;
public class OrientationMidlet extends MIDlet implements CommandListener {
    private Command exitCmd  = new Command("Exit", Command.EXIT, 1);
    private Command resultCmd  = new Command("Result", Command.OK, 1);
    private Display display;
    private Form mainForm;
   /*OrientationMidlet Constructor*/
    public OrientationMidlet(){
   }
    protected void destroyApp(boolean unconditional){
        notifyDestroyed();
    }
    protected void pauseApp(){
    }
```

```
protected void startApp()throws MIDletStateChangeException{
    display = Display.getDisplay(this);
    mainForm = new Form("OrientationMidlet");
    mainForm.append("Add values from External Event Generator “+
                        {(MIDlet > External Events > Location) \n “+
                        “Set Azimuth, Pitch and Roll values”+
                       “and Select Result Command");
    mainForm.addCommand(resultCmd);
    mainForm.addCommand(exitCmd);
    mainForm.setCommandListener(this);
    display.setCurrent(mainForm);
}
private void getOrientation()    {
        try{
                mainForm.deteleAll();
               Orientation orientation = Orientation.getOrientation();
                float azimuth = orientation.getCompassAzimuth();
                float pitch = orientation.getPitch();
                float roll = orientation.getRoll();
                boolean isOrientationMagnetic = orientation.isOrientationMagnetic();
                mainForm.append("azimuth "+azimuth+"\n");
                mainForm.append("pitch "+pitch+"\n");
```

```
                mainForm.append("roll "+roll+"\n");
        mainForm.append("isOrientationMagnetic "+isOrientationMagnetic+"\n");
        }catch(Exception e) {
                e.printStackTrace();
        }
  }
  public void commandAction(Command c, Displayable d) {
    if (c == exitCmd) {
      destroyApp(false);
      notifyDestroyed();
    }else
    if(c == resultCmd)      {
       getOrientation();
    }
  }
}
```