Running Push Registry MIDlet using Samsung SDK

Version 0.9, Draft

INFORMATION GUIDE

COPYRIGHT

Trademarks and Service Marks

About This Document

This document describes how to run sample Push Registry MIDlet using Samsung SDK.

Scope

This document is intended for MIDP developers who wish to develop push registry based mobile Java applications.

Document History:

Date	Version	Comment
17/10/09	0.9	Draft

Abbreviations:

MIDP	Mobile Information Device Profile
AMS	Application Management Software
SMS	Short Message Service
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
GCF	Generic Connection Framework

Table of Contents

Table of Figures

Introduction

A MIDlet is a MID Profile application which is implemented and controlled by the Application Management Software (AMS). MIDP 2.0 came up with many new features for developers to build innovative applications one of them being PushRegistry. MIDP 2.0 PushRegistry feature provides a way for a MIDlet to respond to an inbound connection irrespective of whether the MIDlet is running or not. If the MIDlet is not open then MIDlet will be launched automatically to an incoming event.

PushRegistry can be used to respond to the following:

- Inbound wireless messaging connection such as SMS.
- Inbound network connection such as stream based TCP socket or packet based UDP datagram.
- Timer initiated MIDlet activation.

For Example:

PushRegistry can be used to notify the user when a work item has been created against his/her name, and user can respond to the work item as soon as possible.

Java ME's Push Registry easily pushes a message to a Java ME application and automatically launches the application.

You can also set events for appointments that have been scheduled or you can set timer-based activation to schedule your MIDlet.

Overview

javax.microedition.io.PushRegistry is the component of the AMS that exposes the Push API and keeps track of Push Registration.

Following steps show how PushRegistry works:

1. Connections like messaging (SMS) or Timer or network (socket, datagram) are needed to register a MIDlet application.
2. Push Registry maintains list of inbound connections associated with the application. Java ME application in the mobile device is registered for an event.
3. AMS monitors activity associated with the application.
4. When AMS detects an incoming connection associated with the MIDlet, AMS starts the MIDlet if it is not opened or delivers the response to the running MIDlet.

5. MIDlet now takes over responsibility for the connection and performs the steps necessary to handle the incoming connection.

For more information on PushRegistry please refer to the PushRegistry document in Developing section under Knowledge Base.

Running PushRegistry sample MIDlet using Samsung SDK.

PushMidlet shows two features:

1. How to dynamically register port for invoking MIDlet on incoming SMS.
2. How to register alarm.

Port information is stored in JAD file with key name "**SMS-Port**" and value as "16252". Port from 16000 to 16999 is available to user. Following section shows the Push Registry MIDlet JAD file.

PushMidlet JAD File:

```
MIDlet-1: PushMidlet, PushMidlet.png, PushMidlet
MIDlet-Jar-Size: 3512
MIDlet-Jar-URL: PushMidlet.jar
MIDlet-Name: PushMidlet
MIDlet-Vendor: Unknown
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.1
MicroEdition-Profile: MIDP-2.0
SMS-Port: 16252
```

Steps to run PushMidlet in Samsung SDK:

1. Create PushRegistry Project in Samsung SDK 1.1.1 for the Java ME Platform.

2. Select Samsung device that supports MIDP 2.0 and JSR 120 example a867.

3. Build the Project and then Run via OTA as shown in figure 1.

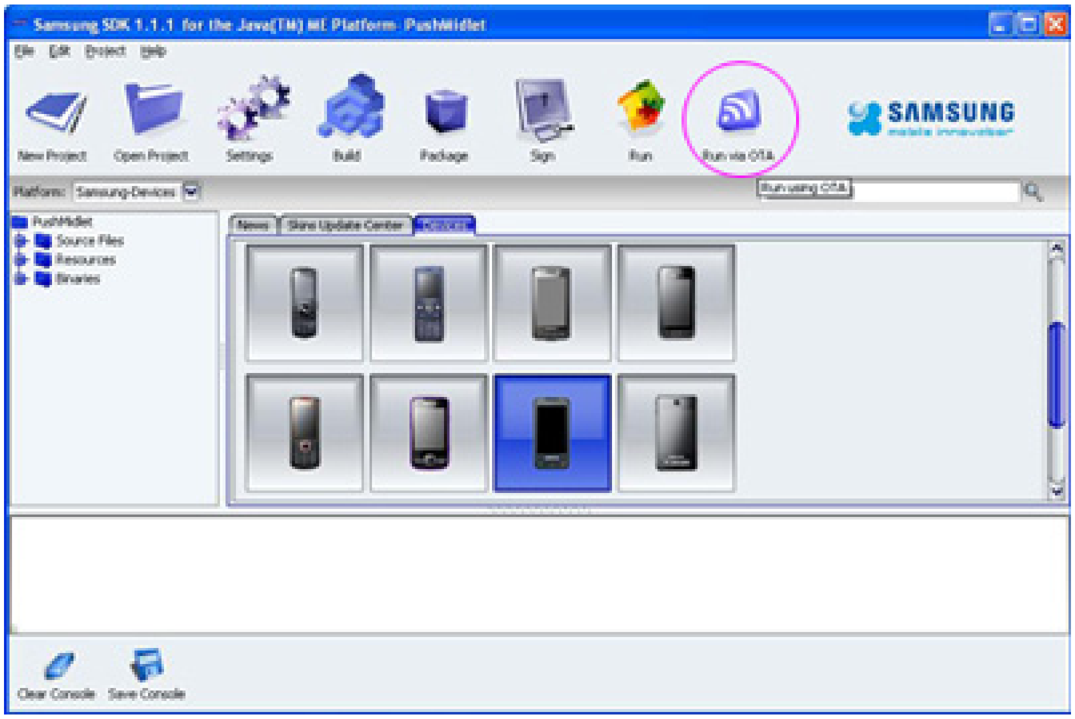


Figure 1: PushMidlet Run via OTA

4. PushMidlet executes in OTA mode as shown in figure 2.

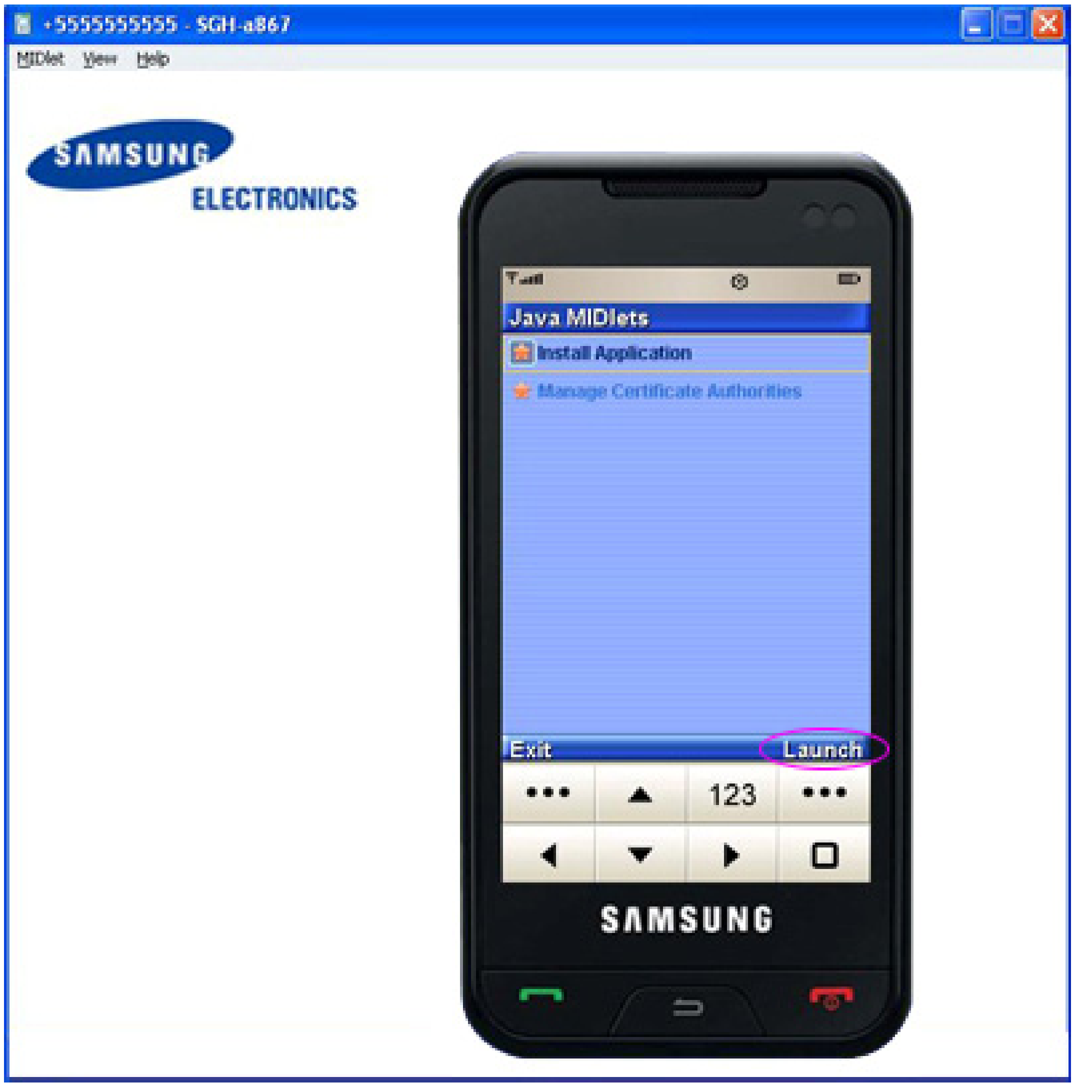


Figure 2: PushMidlet in OTA Mode

5. Select "Install Application" from the list and select **Launch** Command.

6. Installation path appears on screen. Select **Menu > Go to Command** as shown in figure 3.

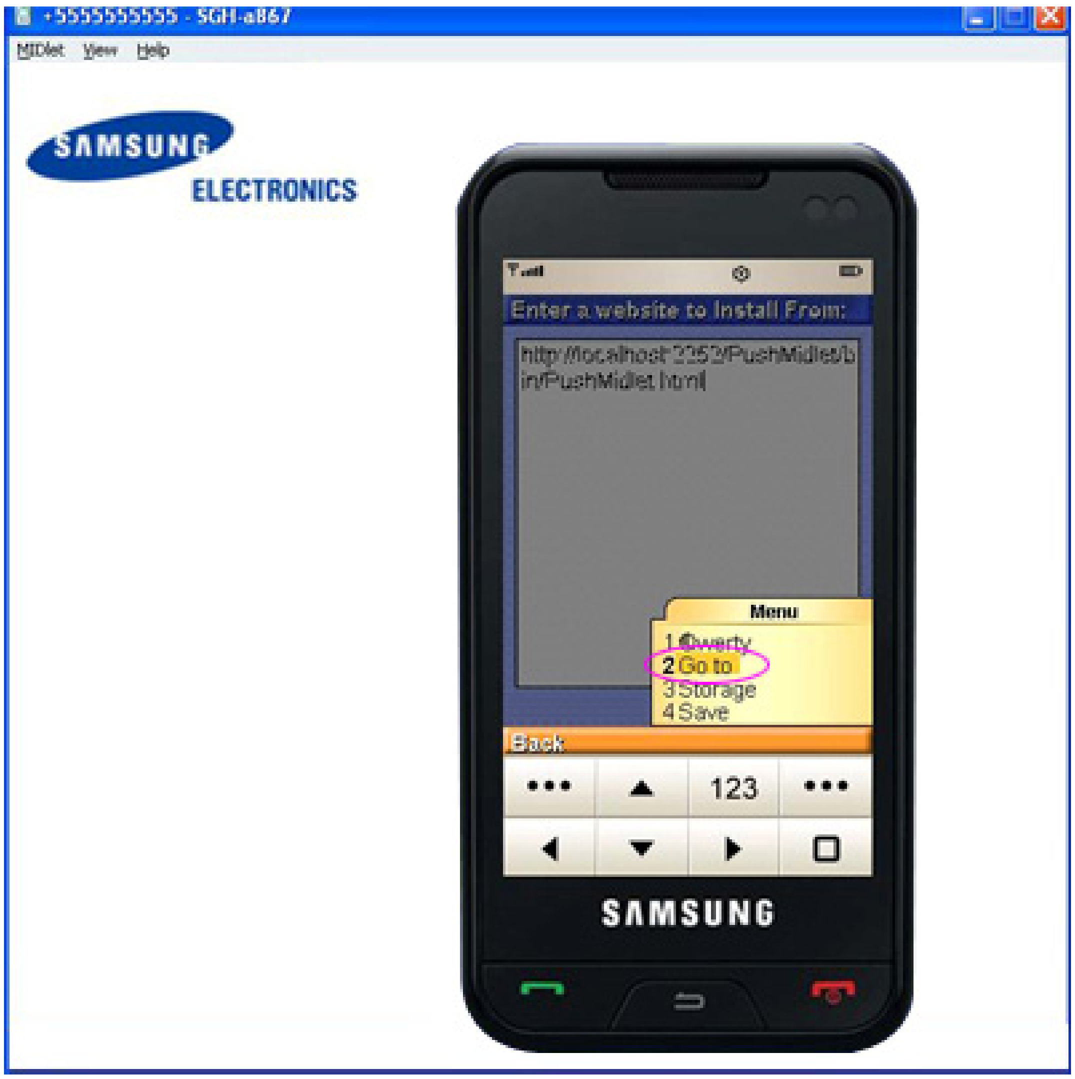
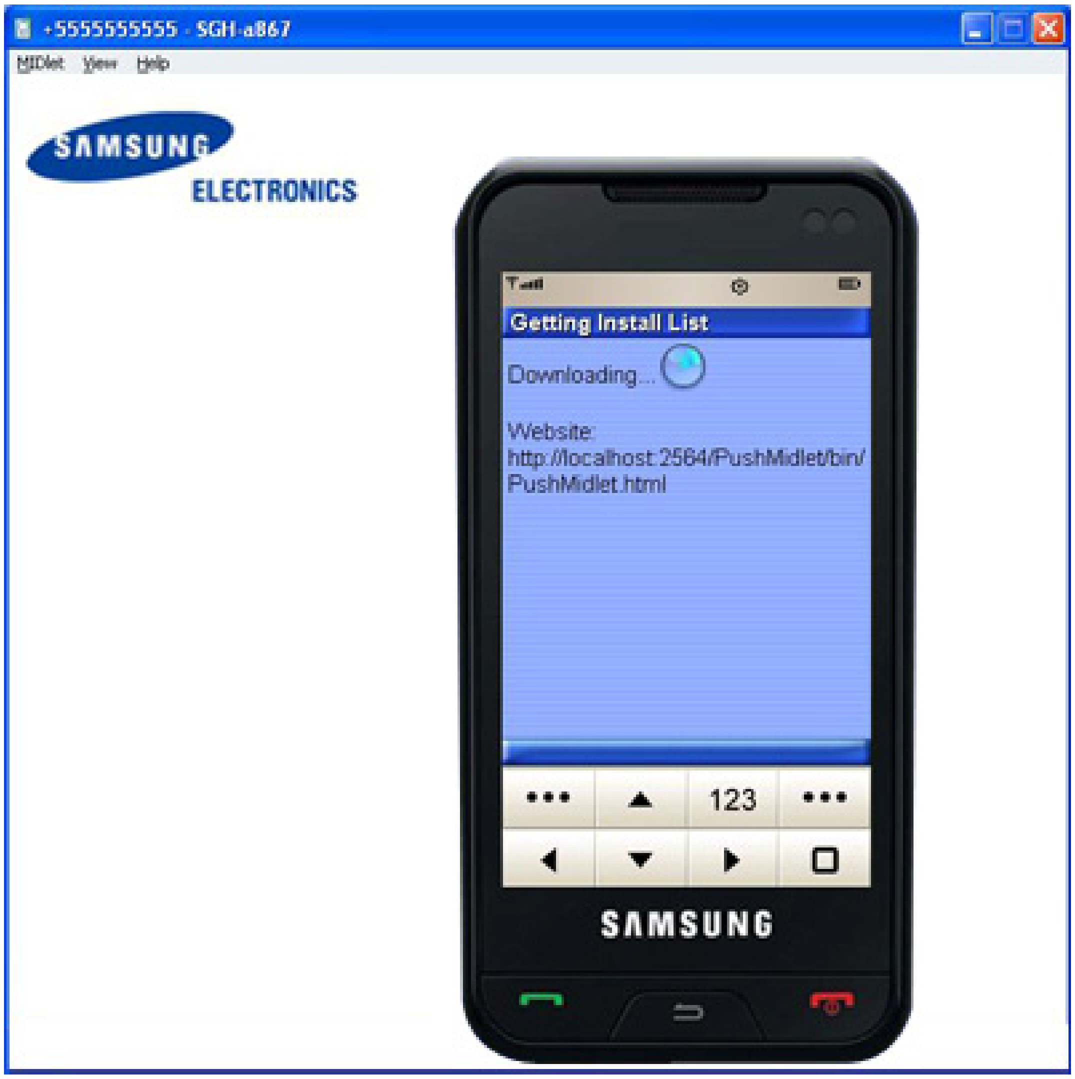


Figure 3: PushMidlet Installation Path

7. Downloading takes place as shown in Figure 4.



Getting Install List

Downloading...

Website:
http://localhost:2564/PushMidlet/bin/PushMidlet.html

Figure 4: PushMidlet Download

8. Once downloaded it will ask for the installation. Install the PushMidlet.jad by selecting the **Install** command as shown in figure 5.

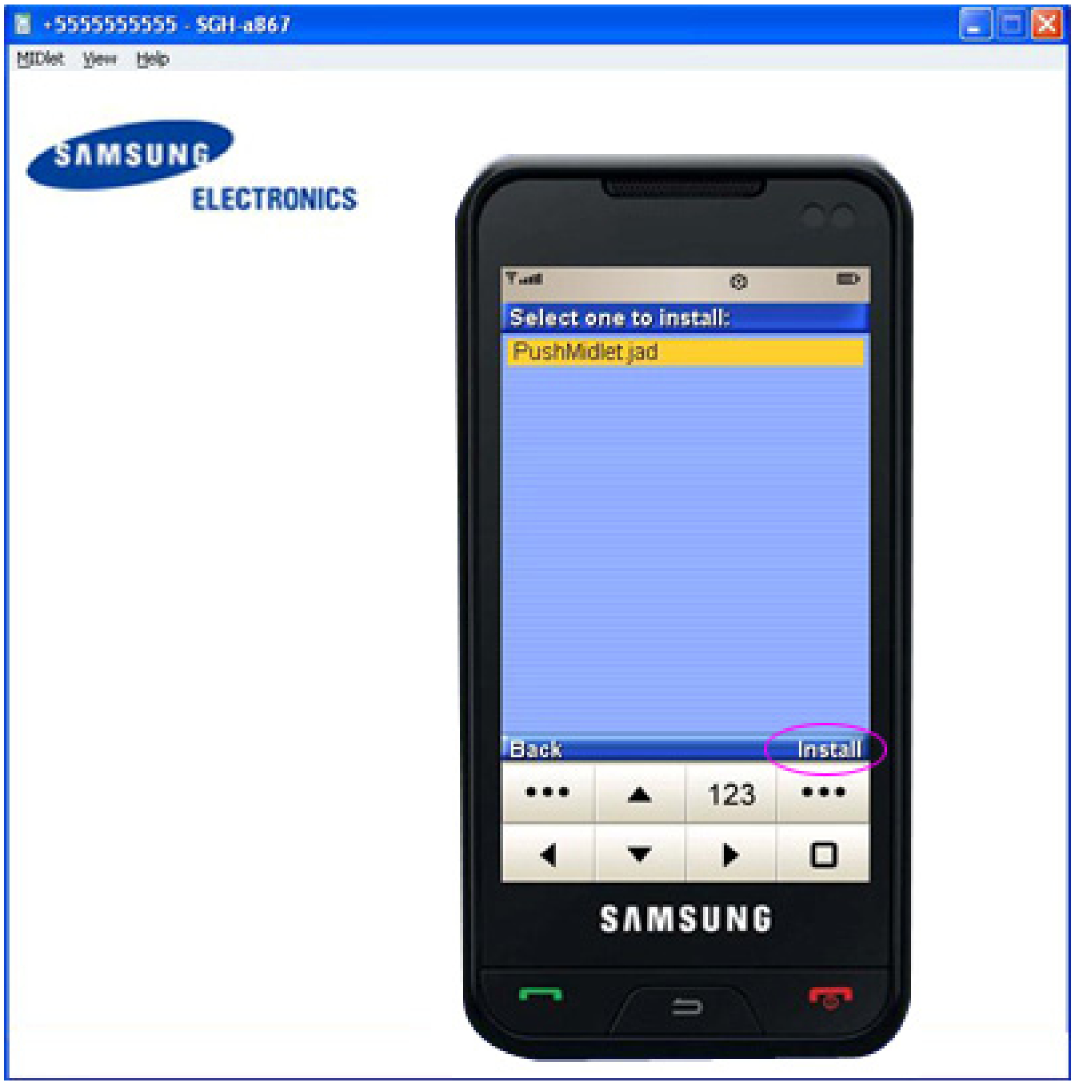
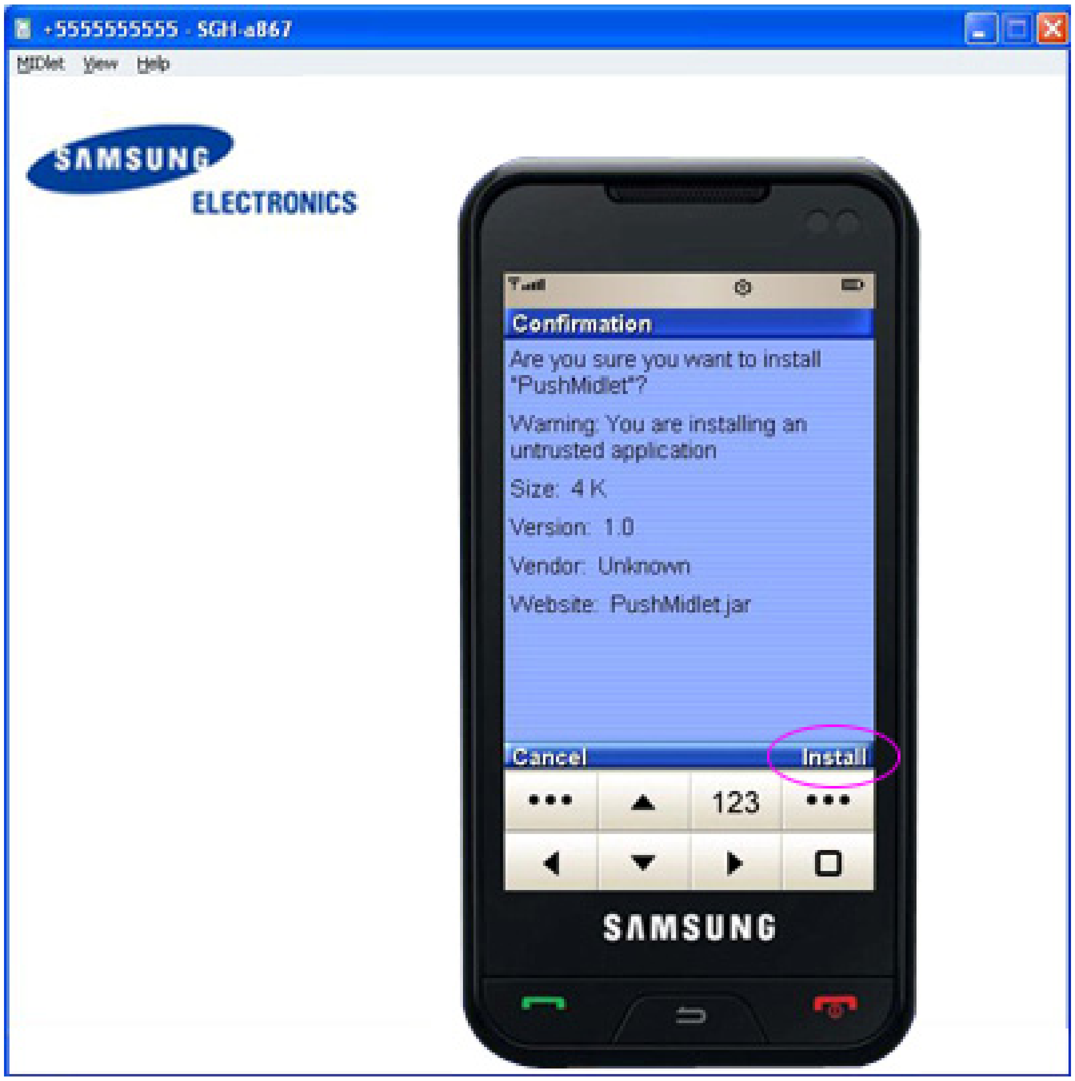


Figure 5: PushMidlet Installation

9. Confirmation screen appears as shown in Figure 6. Select **Install** Command to install the Midlet.



Confirmation

Are you sure you want to install "PushMidlet"?

Warning: You are installing an untrusted application

Size: 4 K

Version: 1.0

Vendor: Unknown

Website: PushMidlet.jar

Figure 6: Confirmation Screen

10. On successful installation it shows the Launch screen as show in Figure 7. Select **Yes** Command

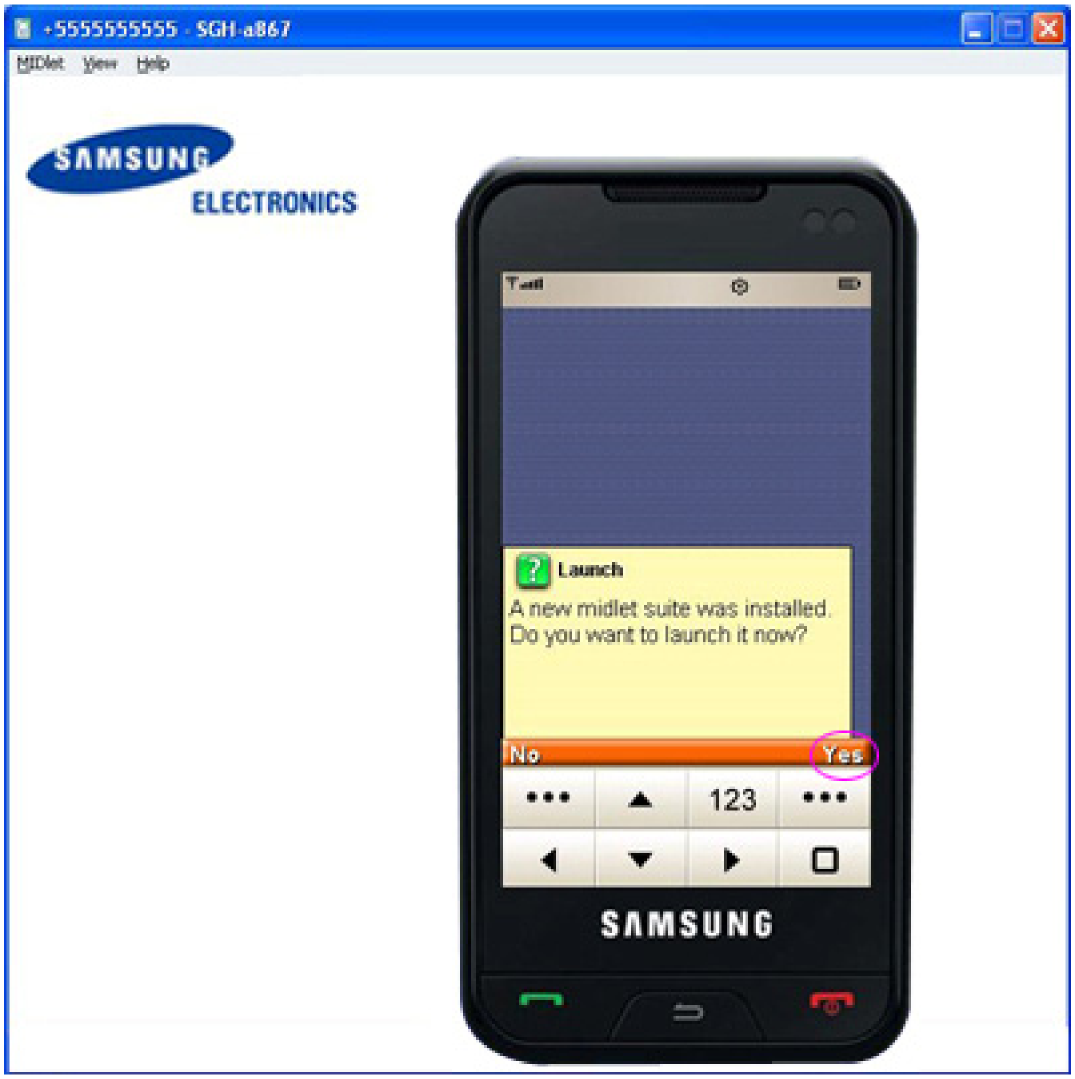
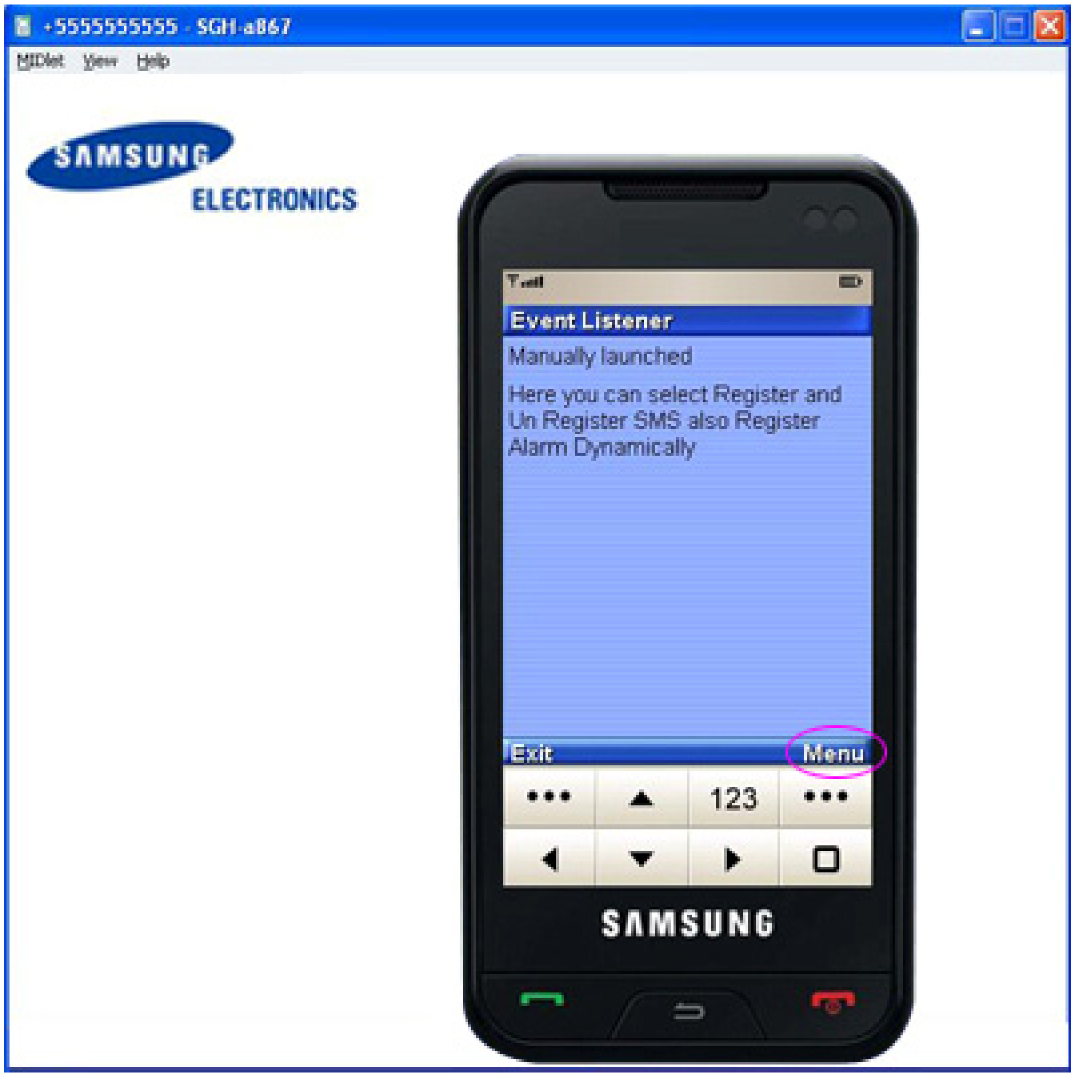


Figure 7: PushMidlet Launch

11. PushMidlet executes as shown in Figure 8. Select **Menu > Register SMS** Command as shown in Figure 9.



Manually launched

Here you can select Register and Un Register SMS also Register Alarm Dynamically

Figure 8: PushMidlet Screen

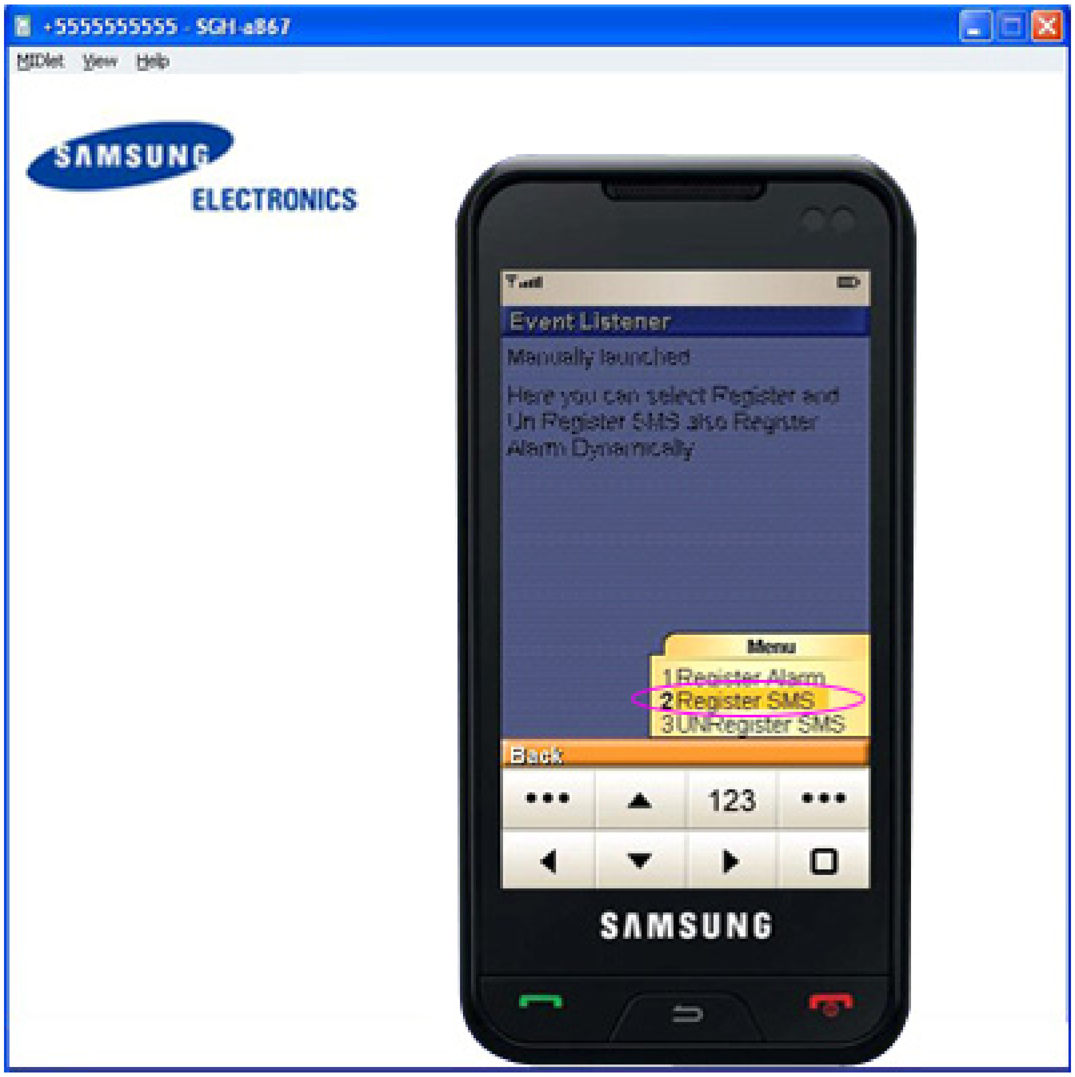
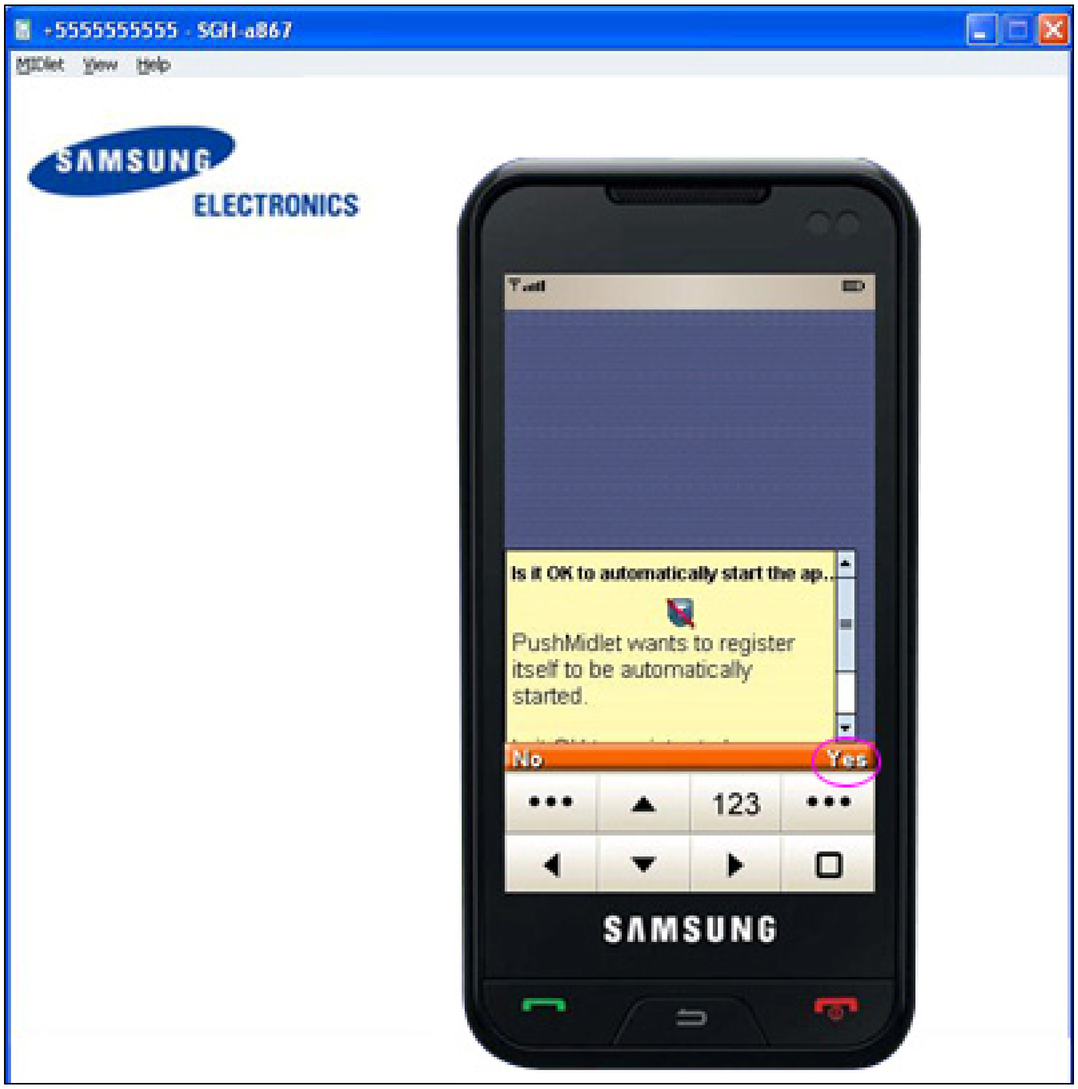


Figure 9: Register SMS

12. Permission Alert appears as shown in Figure 10. Select **Yes** Command



Is it OK to automatically start the ap...

PushMidlet wants to register itself to be automatically started.

Figure 10: Permissions

13. Once all permissions are done it registers the SMS port. Now exit the PushMidlet by selecting the Exit Command. On Exit it shows you the PushMidlet in the list as shown in Figure 11. Do not close the emulator.

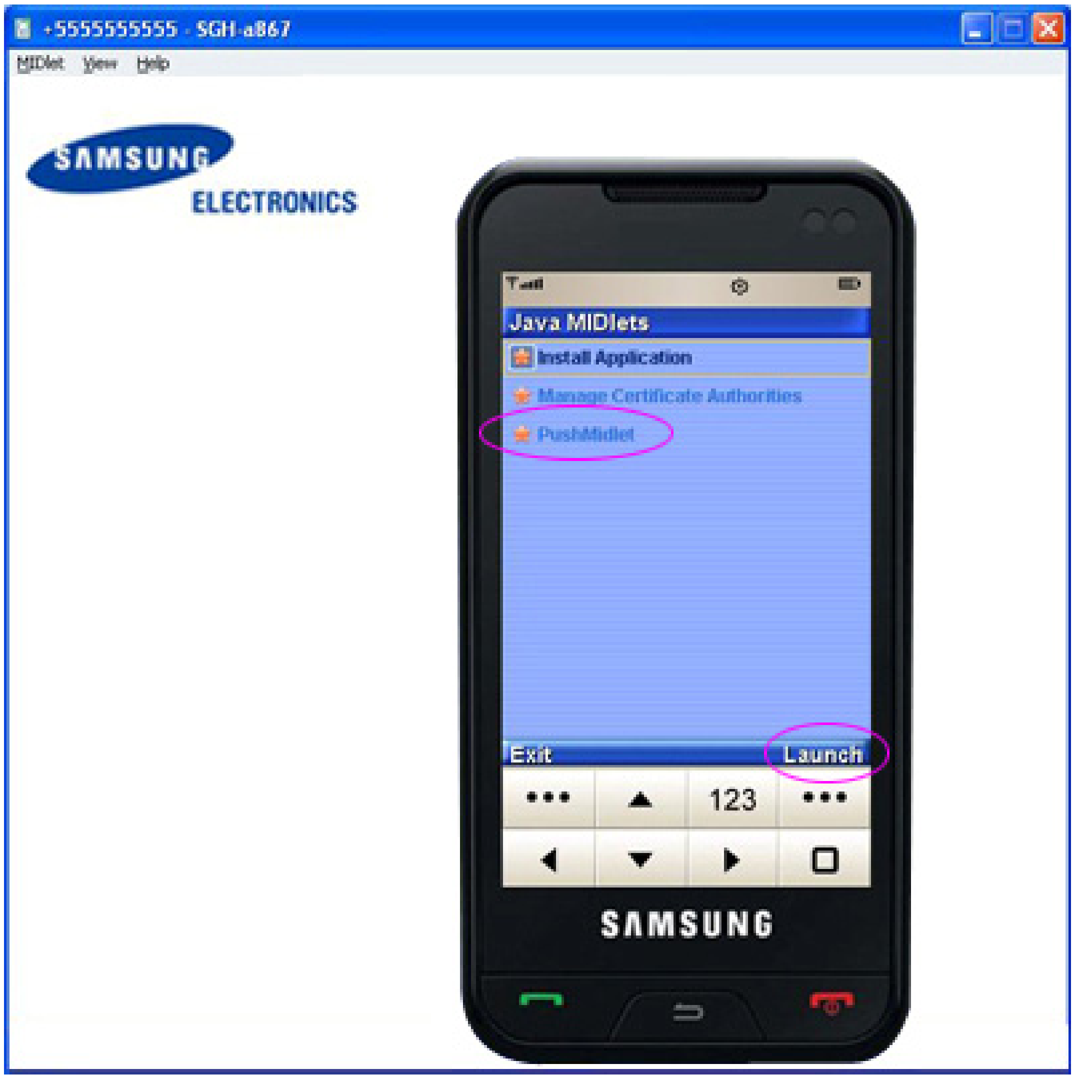


Figure 11: PushMidlet Installed

14. Now select **File > Utilities** from Samsung SDK menu as shown in Figure 12. Utilities screen appears as shown in Figure 13. Select WMA Console and Press Launch button.

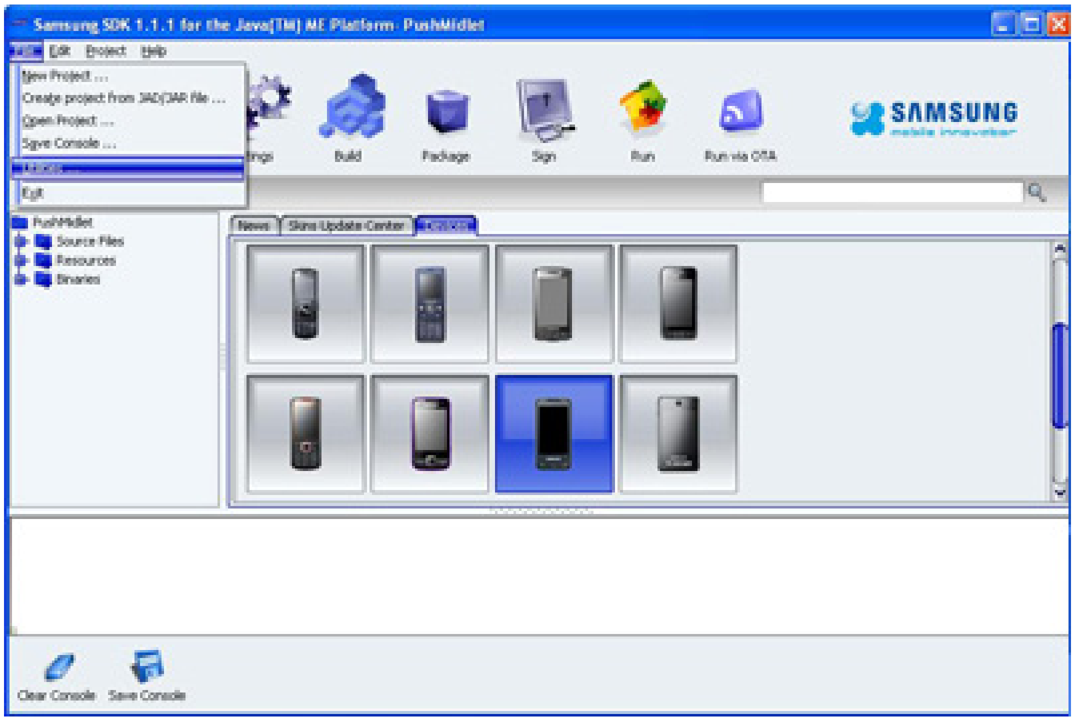


Figure 12: Launching WMA Console

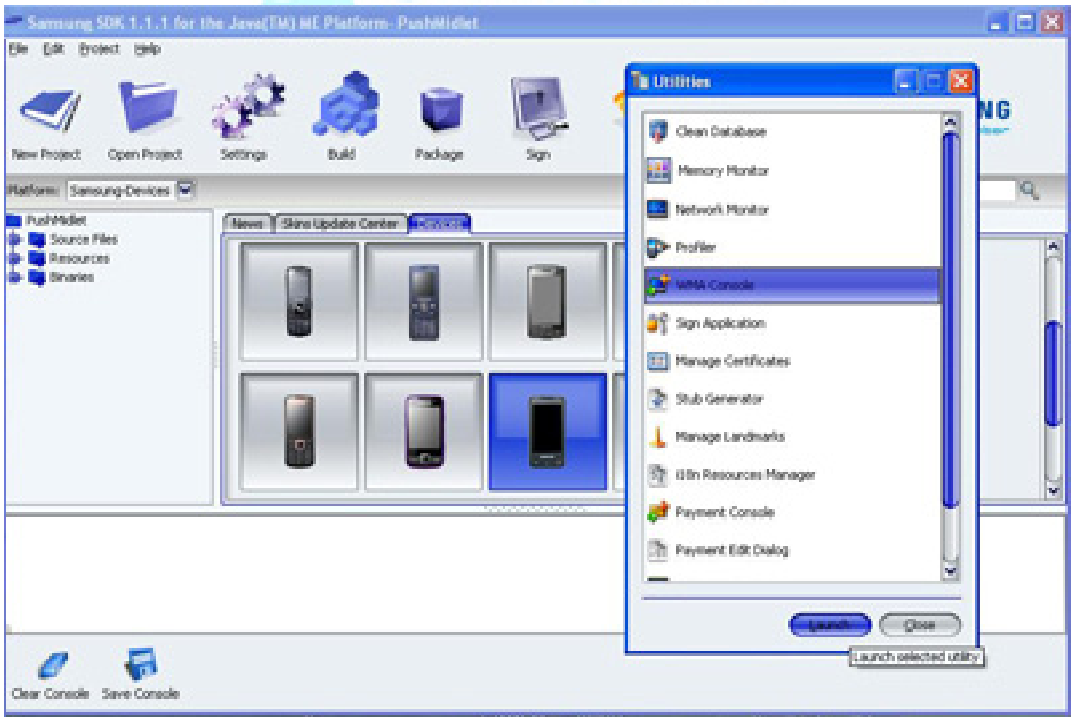
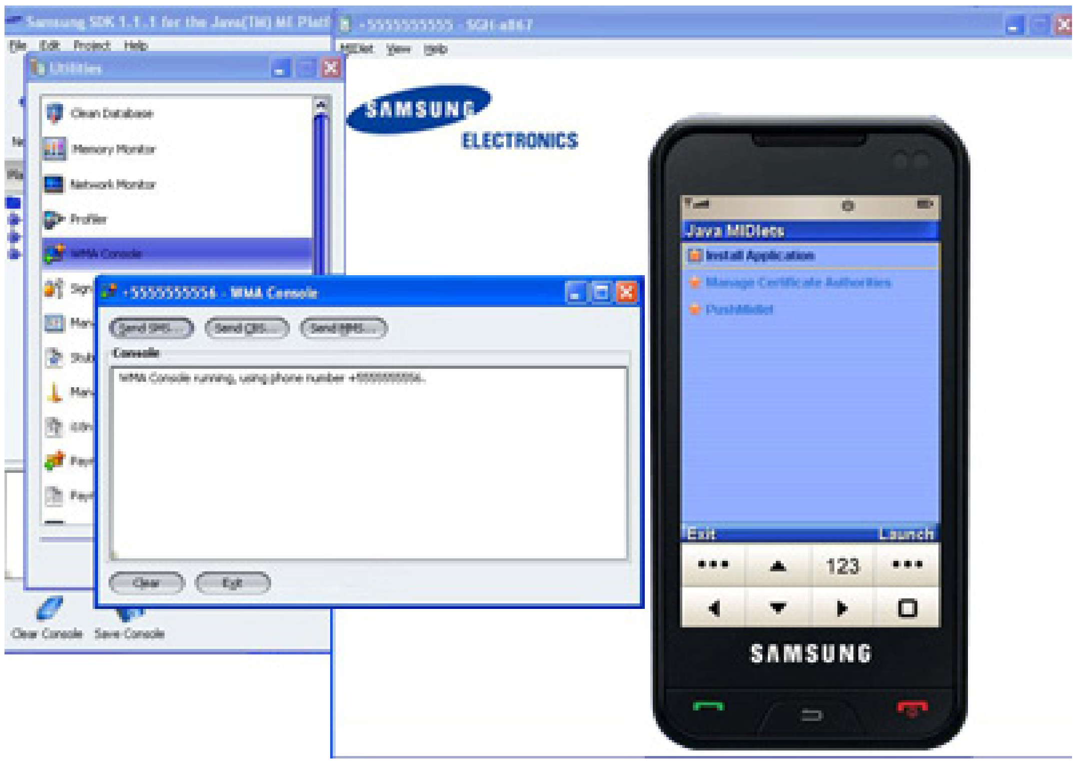


Figure 13: Utilities Screen

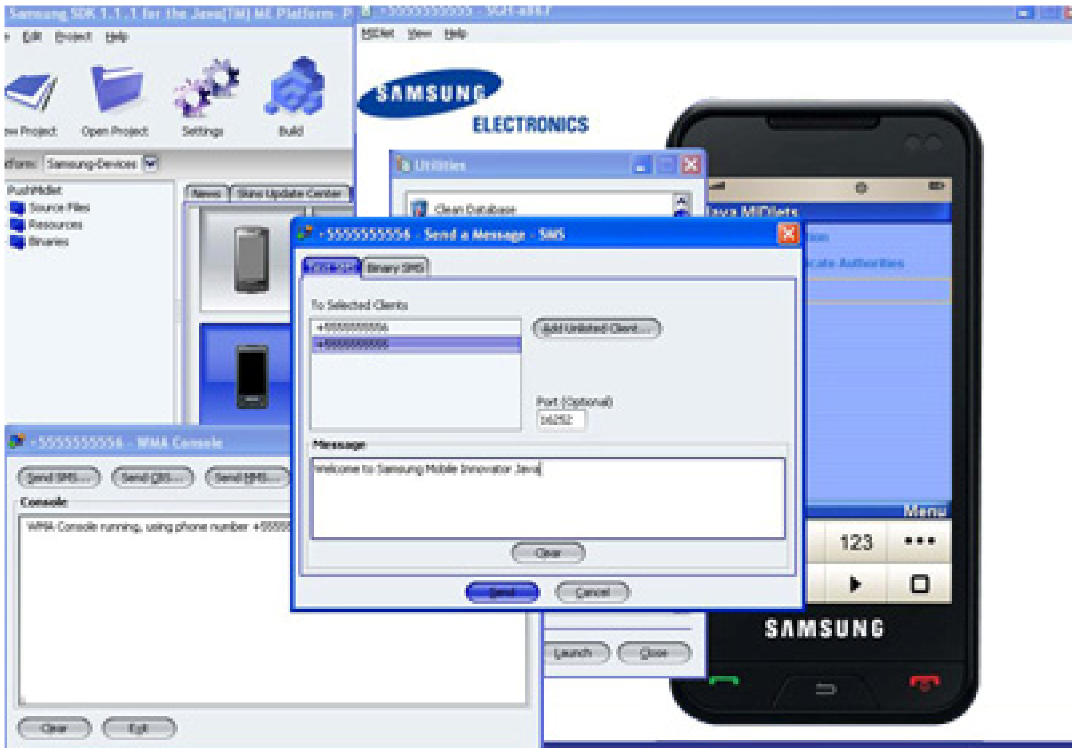
15. WMA Console opens up as shown in Figure 14. It shows unique indentifier number. Each console and Emulator has a unique address that signifies the mobile number. E.g.”55 555 5555” is the address of Emulator and “555 555 5556” is the address of the WMA console.



```
WMA Console running, using phone number +5555555556.
```

Figure 14: WMA Console

16. Select SendSMS button to send message to emulator with address “555 555 5555”. On selecting SendSMS button SMS Editor appears as shown in Figure 15. Select “555 555 5555” number from the client list and enter message your desired message and press Send button.



To Selected Clients: +5555555554, +5555555555

Port (Optional): 16252

Message: Welcome to Samsung Mobile Innovator Java

WMA Console running, using phone number +5555

Figure 15: SMS Editor

17. PushMidlet gets activated on receiving message as shown in figure 16 and figure 17. It asks for the permissions. Select **Yes** command to proceed.

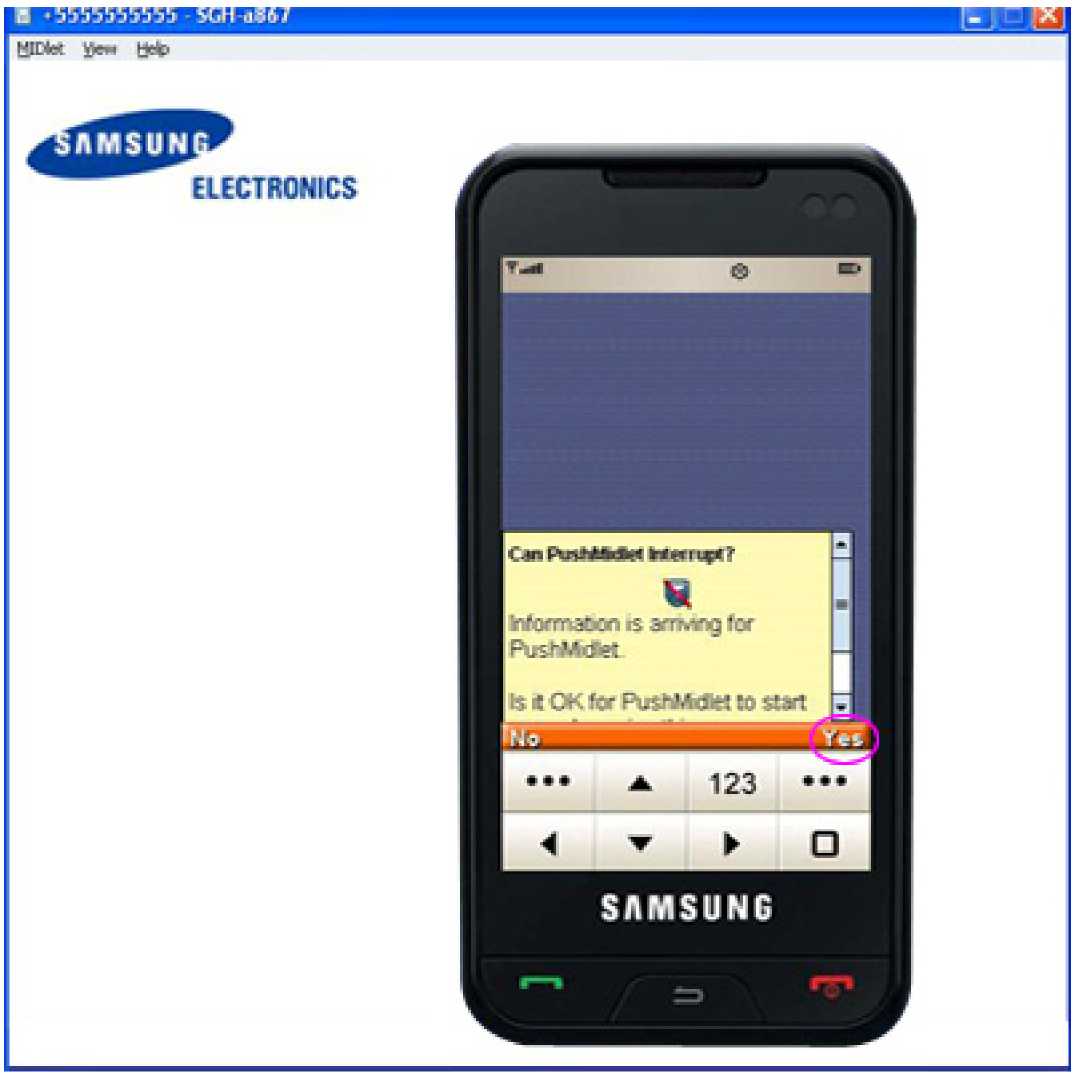
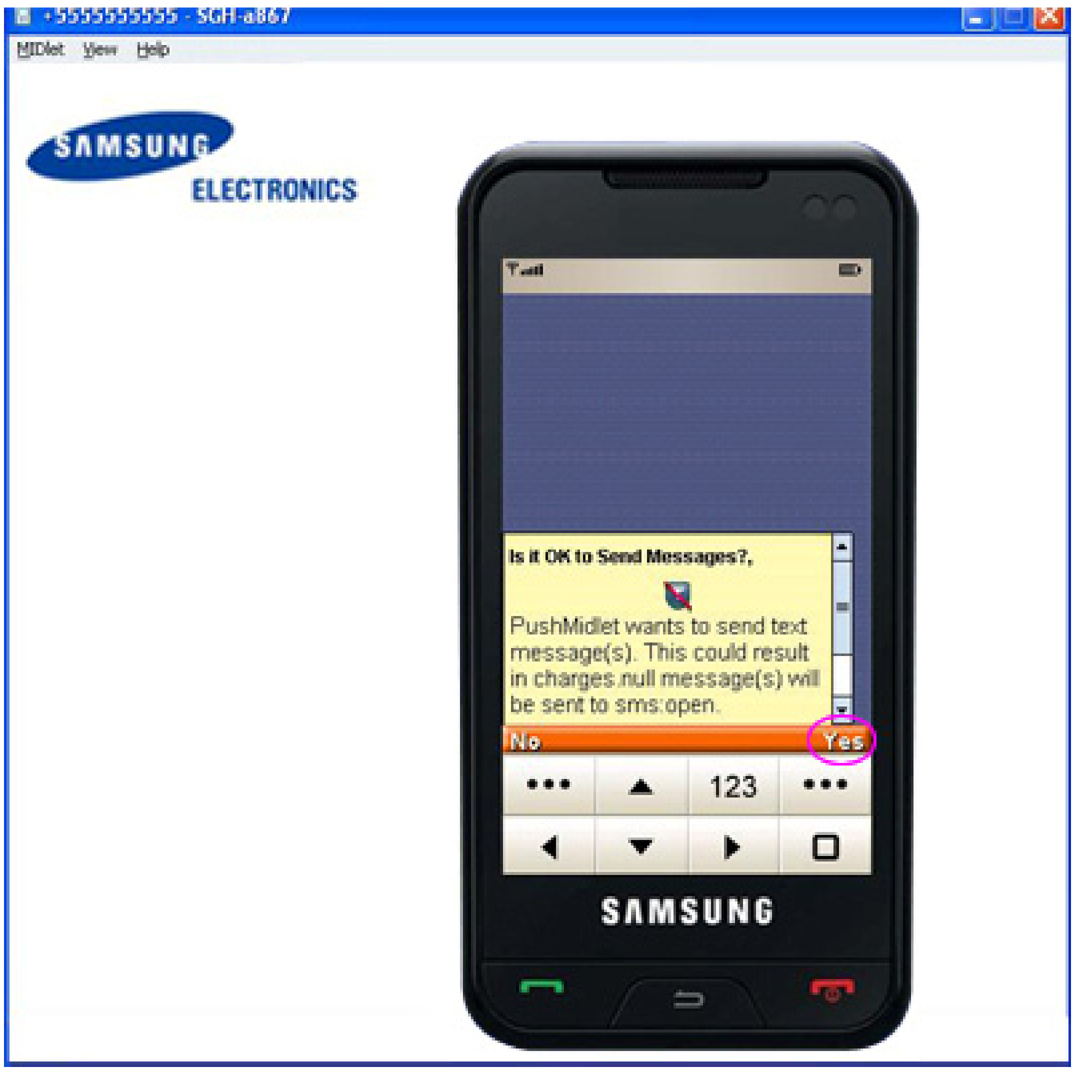

+5555555555 - SGH-a867
MIDlet View Help
SAMSUNG ELECTRONICS
Can PushMidlet Interrupt?
Information is arriving for PushMidlet.
Is it OK for PushMidlet to start
No
Yes
123
SAMSUNG

Figure 16: PushMidlet Launch Permissions

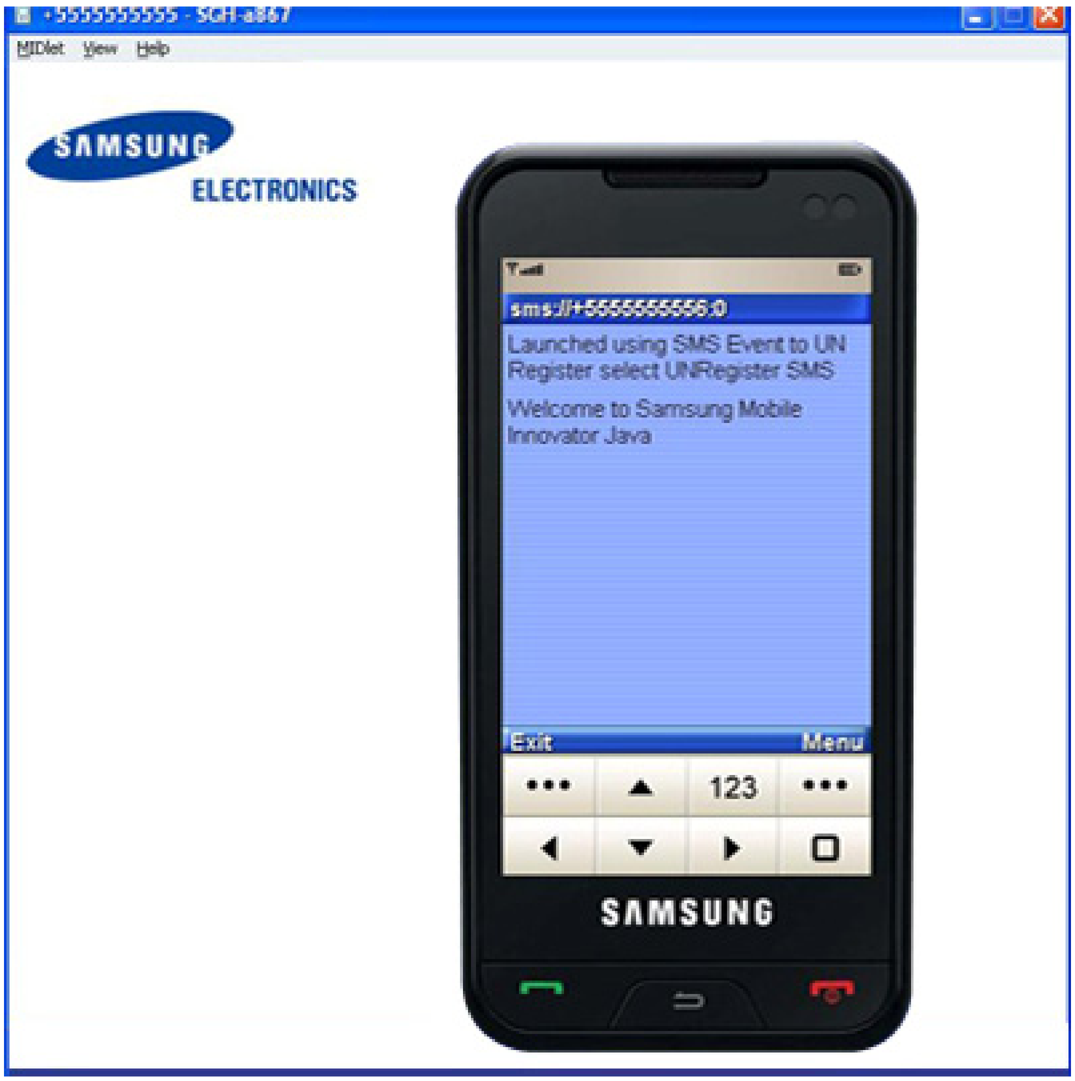


Is it OK to Send Messages?,

PushMidlet wants to send text message(s). This could result in charges.null message(s) will be sent to sms:open.

Figure 17: PushMidlet SMS Permissions

18. "Welcome to Samsung Mobile Innovator Java" message appears as shown in figure 18.



sms://+5555555556:0

Launched using SMS Event to UN Register select UNRegister SMS

Welcome to Samsung Mobile Innovator Java

Figure 18: Displaying Received Message

Code Sample

Class: PushMidlet

```
import java.io.IOException;
import java.util.Vector;
import javax.microedition.io.ConnectionNotFoundException;
import javax.microedition.io.Connector;
import javax.microedition.io.PushRegistry;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.wireless.messaging.MessageConnection;
import javax.wireless.messaging.MessageListener;
import javax.wireless.messaging.TextMessage;

public class PushMidlet extends MIDlet implements CommandListener, MessageListener,
Runnable {

    Display display;
    Form form;

    /*MIDlet class name.*/
    private String midletName = this.getClass().getName();
    Command cmdExit = new Command("Exit", Command.EXIT, 1);
    /*Command for registering alarm*/
    Command cmdAlarm = new Command("Register Alarm", Command.ITEM, 0);
    /*Command for registering SMS Port*/
    Command cmdReg = new Command("Register SMS", Command.ITEM, 0);
    /*Command for unregistering SMS Port*/
    Command cmdUnreg = new Command("UNRegister SMS", Command.ITEM, 0);
    Vector allConn = new Vector();
    Thread thread;
    String smsPort;
    boolean firstTime;

    public PushMidlet() {
        display = Display.getDisplay(this);
        smsPort = getAppProperty("SMS-Port");
        firstTime = true;
        form = new Form("Event Listener");
        form.addCommand(cmdExit);
        form.addCommand(cmdAlarm);
        form.addCommand(cmdReg);
        form.addCommand(cmdUnreg);
        form.setCommandListener(this);
    }

    public void startApp() {
        getListConnections();
        display.setCurrent(form);
    }

    public void pauseApp() {
    }
```

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

  /*Check for list of connections recieved*/
  public void getListConnections() {
    String regConnections[];

    regConnections = PushRegistry.listConnections(true);

    if (regConnections.length != 0) {
      form.append("Launched using SMS Event to UN Register select UNRegister SMS");

      for (int i = 0; i < regConnections.length; i++) {
        try {
          MessageConnection msgconn = (MessageConnection)
Connector.open(regConnections[i]);
          msgconn.setMessageListener(this);
          allConn.addElement(msgconn);
        } catch (SecurityException exSec) {
          form.append("SecurityException=" + exSec);
        } catch (IOException exIO) {
          form.append("IOException==" + exIO);
        }
      }

    } else {
      if (firstTime) {
        form.append("Manually launched");
        form.append("Here you can select Register and Un Register SMS also Register Alarm
Dynamically");
        firstTime = false;
      }
      regConnections = PushRegistry.listConnections(false);
    }


  }

  /*Register the Alarm for certain period for auto launch*/
  private void registerAlarm(final long duration) {
    new Thread() {

      public void run() {
            long alarmTiming = System.currentTimeMillis() + duration;

            try {
                /*to register MIDlet for a time period*/
                PushRegistry.registerAlarm(midletName, alarmTiming);
            } catch (ClassNotFoundException ex) {
                    form.append("\n ClassNotFoundException");
            } catch (ConnectionNotFoundException ex) {
                    form.append("\n ConnectionNotFoundException");
            }
        }
    }.start();

}

public void commandAction(Command cmd, Displayable disp) {
    if (cmd == cmdExit) {
        exitMidlet();

    } else if (cmd == cmdAlarm) {
        registerAlarm(4000);
    } else if (cmd == cmdReg) {

        RegisterSMSConn();
    } else if (cmd == cmdUnreg) {
        UnRegisterSMSConn();
    }
}

public void exitMidlet() {
    closeConnections();
    destroyApp(true);
    notifyDestroyed();
}

/*Make dynamic connection registered for a specific sms port*/
public void RegisterSMSConn() {
    thread = new Thread(this);
    thread.start();

}
```

```
public void run() {
    try {
        /*To register Midlet for a port number*/
        PushRegistry.registerConnection("sms://:" + smsPort, midletName, "*");
        closeConnections();
        getListConnections();
    } catch (ClassNotFoundException exe) {
    } catch (IOException ex) {
    }
}

/*To unregister the port dynamically*/
public void UnRegisterSMSConn() {
    PushRegistry.unregisterConnection("sms://:" + smsPort);
}

public void closeConnections() {
    if (allConn != null) {
        while (allConn.isEmpty() == false) {
            MessageConnection msgConn =
                    (MessageConnection) allConn.firstElement();
            if (msgConn != null) {
                try {
                    msgConn.setMessageListener(null);
                    msgConn.close();
                } catch (Exception exp) {
                }
            }
            allConn.removeElementAt(0);
        }
    }
}

/* Invokes when recieved message on specify port*/
public void notifyIncomingMessage(MessageConnection msgConn) {

    TextMessage message = null;
    try {
        message = (TextMessage) msgConn.receive();
        String sendAddr = message.getAddress();
        form.setTitle(sendAddr);
        String messageText = message.getPayloadText();
        form.append(messageText);
        } catch (IOException ex) {
            form.append("Exception here" + ex);
        }
    }
}
```