

Wireless Messaging API (WMA) – JSR 120

Overview:

The Wireless Messaging API (WMA) is an optional package for the Java 2 Platform, Mobile Edition (J2ME) that provides platform-independent access to wireless communication resources like Short Message Service (SMS) and Cell Broadcast Service (CBS). Currently WMA can be used on top of CLDC and MIDP, but could also be implemented on CDC-based devices. Wireless Messaging – WMA supports SMS & CBS.

WMA is supported in Motorola MIDP2.0 handsets.

WM API - JSR 120:

1. Provide standard access to wireless messaging communication resources
2. WMA is designed to run on any configuration (CDC or CLDC) and enhance a J2ME Profile
3. Addresses the following technologies:
 - a. Short Message Service (SMS)
 - b. Cell Broadcast Service (CBS)
4. Based on Generic Connection Framework
5. Design of the messaging functionality is similar to the `UDPDatagramConnection` in MIDP 2.0
6. WMA interfaces are in `javax.wireless.messaging`.

Providing an Emulated Environment for WMA Simulation:

Within the “Motorola SDK v4.3 for J2ME”, there is WMA server. This allows a J2ME application developer to send and receive SMS/CBS based communications to and from a Motorola handset emulator, thus making development of SMS based applications easier and cheaper for the developer as they can be fully tested within the simulated environment provided by the SDK before deployment onto real handsets on an operator’s network.

How to Use WMA Server:

To send a message from the WMA Server, follow the steps below:

1. Create a 'send.txt' file with the following commands:
DestAddress: +1234567
DestPort: **10001**
SrcAddress: +12345678
TimeStamp: 000000F0EC65D783
PayloadType: 0

PayloadLength: 11
PayloadDataLength: 10
Payload: **C8329BFD66DDDF723619**

Note 1: The String that is to be sent from the WMA to the handset is placed in the payload field and is 7 bit character encoded for SMS from 8-bit ASCII. The example string is "Hello, World".

NOTE 2: The value in the DestPort field must match the port used in the SMS Receive MIDlet at the point of the MessageConnection instantiation or initialization.

Save the file 'send.txt' in: <SDK root directory>\AplixWmaTestTool\TestWmaServer

2. Start the WMA Test Server in Motorola SDK.
3. Start your MIDlet through Motorola SDK with the A630, C650, E1000, E398, T725, V80, V300/V400/V500, V600, and V180/V220 emulators, which will receive the message.
4. Type "TX SMS send.txt" from the server window command prompt and press <ENTER>.

A Complete Working Sample MIDlet:

***** THIS CODE IS USED FOR DEMONSTRATION PURPOSES ONLY*****

The following code illustrates how to SEND the message as a normal SMS

```
/*
 * SMS_TXMIDlet.java
 *
 * Created on 15 October 2004, 15:41
 */

package SMS_RX_TX;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.wireless.messaging.*;
import java.io.IOException;
import javax.microedition.io.*;

/**
 * An example MIDlet which demonstrates sending an SMS to the WMA
 * server.
```

```

    * This is a simple MIDlet which allows the user to enter a phone
number
    * with a minimum of 8 characters. Once entered it switches the UI to
a
    * TextBox to allow text entry for the SMS message payload. Finally
the
    * user can send the message to the WMA server.
**/

```

```

public class SMS_TXMIDlet extends MIDlet implements CommandListener {

    private Form form;                // The form UI object
    private TextBox tb;               // The SMS Text Entry UI object
    private TextField tf;             // The text field for phone number
    private Command exitCmd;          // The exit command object
    private Command composeCmd;       // The compose SMS command object
    private Command sendCmd;          // The send SMS command object
    private Display display;          // The display object
    private String txPort = "10000"; // The Send SMS port number

    public SMS_TXMIDlet() {

        // Initialise objects
        form = new Form("SMS Transmit");
        tf = new TextField("Enter Phone Number", "", 25,
TextField.PHONENUMBER);
        tb = new TextBox("Compose SMS", "", 100,
TextField.ANY);
        composeCmd = new Command("Compose", Command.SCREEN, 2);
        sendCmd = new Command("Send", Command.SCREEN, 2);
        exitCmd = new Command("Exit", Command.EXIT, 3);
        display = Display.getDisplay(this);

        // Build Form UI
        form.addCommand(exitCmd);
        form.addCommand(composeCmd);
        form.append(tf);
        form.setCommandListener(this);

        // Build TextBox UI
        tb.addCommand(exitCmd);
        tb.addCommand(sendCmd);

        // Associate display with form
        display.setCurrent(form);
    }

    public void startApp() {

    }

    public void pauseApp() {
    }
}

```

```

public void destroyApp(boolean unconditional) {
}

public void commandAction(Command c, Displayable s) {
    if (c == exitCmd) {
        destroyApp(false);
        notifyDestroyed();
    }

    if (c == composeCmd) {

        if (tf.getString().length() > 7) {

            // Switch UI to TextBox
            display.setCurrent(tb);
            tb.setCommandListener(this);
        }
        else {
            out("Phone Number Invalid Length");
            dbg("SMS_TXMIDlet.commandAction(): Phone Number Invalid
Length");
        }
    }

    if (c == sendCmd) {
        sendSMS(tb.getString());
    }
}

// Method which does the actual sending of the SMS to the WMA
server
private void sendSMS(String s) {

    try {

        String addr = "sms://" + tf.getString() + ":" + txPort;
        MessageConnection conn = (MessageConnection)
Connector.open(addr);
        TextMessage msg =

(TextMessage)conn.newMessage(MessageConnection.TEXT_MESSAGE);
        msg.setPayloadText(tb.getString());
        conn.send(msg);
        dbg("SMS_TXMIDlet.sendSMS(): " + tb.getString());
    } catch (IOException ioe) {
        dbg("SMS_TXMIDlet.sendSMS(): " + ioe.toString());
    }
}

private void out(String s) {

```

```

        form.append(s + "\n");
    }

    private void dbg(String s) {
        System.out.println(s);
    }
}

```

The following code illustrates how to RECEIVE the message as a normal SMS

```

/*
 * SMS_TXMIDlet.java
 *
 * Created on 15 October 2004, 15:41
 */

package SMS_RX_TX;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.wireless.messaging.*;
import javax.microedition.io.*;
import java.io.IOException;

/**
 * An example MIDlet which demonstrates receiving an SMS from the WMA
 * server.
 * This MIDlet is very basic. To use, simply run it in the Emulator
 * and then
 * send an SMS to it from the WMA server. When the MIDlet receives the
 * message
 * it will be displayed on the UI.
 * Remember when sending a string from the WMA, it must be converted
 * into 7bit
 * encoding before being sent.
 */

public class SMS_RXMIDlet extends MIDlet implements CommandListener,
MessageListener {

    private Form form; // The form UI object
    private Command exitCmd; // The exit command object
    private Display display; // The display object
    private String rxPort = "10001"; // The receive SMS port number
    private Reader reader; // The message reader object
    private MessageConnection messconn; // The MessageConnection object
    private boolean done; // The done reading flag

    public SMS_RXMIDlet() {

        // Initialise objects
        form = new Form("SMS Receive");
    }
}

```

```

        exitCmd = new Command("Exit", Command.EXIT, 2);
        display = Display.getDisplay(this);

        // Build UI
        form.addCommand(exitCmd);
        form.setCommandListener(this);

        // associate display with form
        display.setCurrent(form);

        // create message listener
        try {
            // create receiving port connection.
            messconn = (MessageConnection) Connector.open("sms://:" +
rxPort);

            // Register a listener for inbound messages.
            messconn.setMessageListener(this);

            // Start a message-reading thread.
            done = false;
            reader = new Reader();
            new Thread(reader).start();

        } catch (IOException ioe) {
            dbg("SMS_RXMIDlet.SMS_RXMIDlet(): " + ioe.toString());
        }
    }

    public void startApp() {
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

    public void commandAction(Command c, Displayable s) {
        if (c == exitCmd) {
            destroyApp(false);
            notifyDestroyed();
        }
    }
}

class Reader implements Runnable {

    private int pendingMessages = 0;

    // The run method performs the actual message reading.
    public void run() {

```

```

while (!done) {
    synchronized(this) {
        if (pendingMessages == 0) {
            try {
                wait();
            } catch (Exception e) {
                dbg("Reader.run(): " + e.toString());
            }
        }
        pendingMessages--;
    }

    try {
        Message mess = messconn.receive();
        TextMessage txtMess = (TextMessage)mess;
        String update = txtMess.getPayloadText();
        dbg("Reader.run(): " + txtMess.getPayloadText());

        // if a message has arrived, show message to the user
        if (update != null) {
            dbg("Reader.run(): Processed new SMS");
            dbg("Reader.run(): Message contents: " + update);
            form.deleteAll();
            form.append("New msg:\n" + update);

            update = null;

            } else { dbg("Reader.run(): No Update Info"); }
        } catch (IOException ioe) {
            dbg("Reader.run(): " + ioe.toString());
        }
    }
}

public synchronized void handleMessage() {
    pendingMessages++;
    dbg("Reader.handleMessage(): New message arrived");
    notify();
}

public void notifyIncomingMessage(MessageConnection conn) {
    if (conn == messconn) {
        reader.handleMessage();
    }
}

// General Debug method to output debug info to System.out
private void dbg(String s) {

    System.out.println("DEBUG: " + s);
}
}

```

Additional Sources of Information:

For additional information regarding functionality of a device you are targeting for development please refer to the J2ME Developer Guides available at <http://www.motocoder.com>

Also for additional information about the features described here then you can review the online Knowledgebase at MOTOCoder through the FAQ link and also use the JavaDocs that are supplied with the