# Wireless Messaging API (WMA)

**版本 0.9，初稿**




# API 指南

## 版权声明



## 商标和服务标识

# 关于本文档

本文概要介绍了 WMA（Wireless MessagingAPI），并演示了如何创建生成一个三星手机 WMA 程序。文档包括 WMA 架构和 API 的介绍以及一段关于如何收发短信的代码。

**读者对象：**

本文面向想要开发 Java ME 应用的 Java ME 程序员，读者应具备良好的 java 编程知识。

**文档版本：**

| 日期 | 版本 | 备注 |
|---|---|---|
| 05/02/09 | 0.9 | 初稿 |

**参考资料：**

1. WMA 1.1 规范：

   http://jcp.org/en/jsr/detail?id=120

2. WMA

   http://developers.sun.com/mobility/midp/articles/wma/

**缩略术语：**

| | |
|---|---|
| Java ME | Java 平台微型版( Java Platform Micro Edition) |
| CLDC | 有限连接设备配置( Connection Limited Device Configuration) |
| WMA | 无线信息 API( Wireless Messaging API) |
| CBS | 蜂窝广播服务( Cell Broadcast Service) |
| SMS | 短信息服务( Short Messaging Service) |
| CGF | 通用链接框架( Generic Connection Framework) |
| URL | 统一资源定位符( Uniform Resource Locator) |
| API | 应用编程接口( Application programming Interface) |

# 目录



## 图表目录

# 引言

Wireless MessagingAPI (WMAPI)用跨平台访问无线通讯资源，如 SMS（短信服务）和 CBS（蜂窝广播服务）。messaging API 基于有限连接设备配置(CLDC)规范中定义的通用链接框架(GCF)。
Wireless MessagingAPI(WMA)是 Java ME 的可选包。

# 概览

文档阐释了如下信息协议：

- 蜂窝广播服务(CBS)
- 短消息服务(SMS)

**蜂窝广播服务(CBS)**

- 使用 GSM 蜂窝广播服务，信息可以被发送到每一个移动终端(MS)。如：移动电话和传真机。
- cell 的广播消息周期性的重复，因此当 MS 无论何时进入到某小区时，都能接收到这些广播信息
- 二进制数据和 ASCII 码是发送数据的两种方式。在 ASCII 码方式下，文本的长度最大是 15 页，每页 93 个字符。文本设置只支持 ASCII 码信息。

**短消息服务(SMS)**

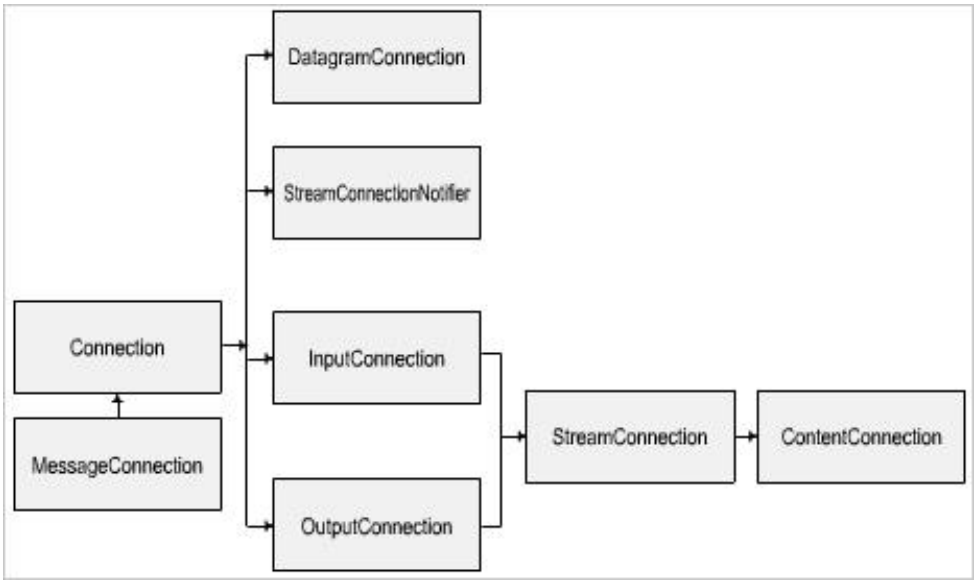
- 短消息服务是一个用于手机文本/二进制短信息收发传输的通讯协议，该协议最初定义为 GSM 标准的一部分。
- 每条信息的最大长度为 160 个字符。

本文概要介绍了 Wireless MessagingAPI。欲获取更多信息，请参考 Java Community Process (JCP)定义的规范。*javax.wireless.messaging.Message 是所有 messages 要实现的 interface*。它提供了地址和时间戳的方法。*Message* 接口对所有 messages 提供通用方法。

信息的数据部分由文本信息和二进制信息组成，由从 *Message* 继承的 *TextMessage* 和 *BinaryMessage* 接口表示。

如图 1 所示，信息收发功能是由 *MessageConnection* (从通用连接框架(GCF)接口继承)实现的。图 1 描述了 WMA 与 GCF 的关系。

如果程序无权执行这些操作，收发短信的方法会抛出 *java.lang.SecurityException* 异常。



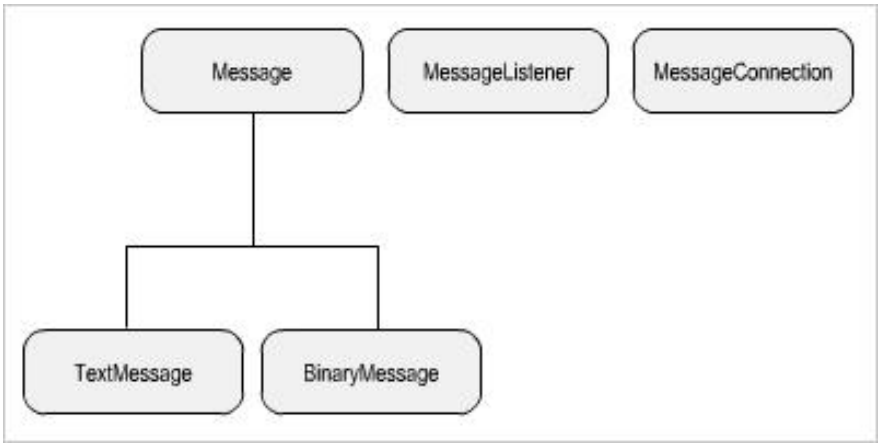

**图表 1：通用连接框架**

## API 描述：

WMA API 由一个单独的包 *javax.wireless.messaging* 构成，这个包定义了所有需要收发无线文本和二进制信息的 API。下面给出 WMA 指定的接口：

*Message*
*BinaryMessage*
*TestMessage*
*MessageConnection*
*MessageListener*

图 2 显示的是 Wireless MessagingAPI 组件：




**图表 2：Wireless MessagingAPI 组件**

## 接口

*Message:*

这是基础 Interface。*TextMessage* 和 *BinaryMessage* 是从 Message 接口派生的。它作为一个容器存储了信息的地址和内容。Message 具备地址和时间戳的方法，这些方法是 *getAddress()*，*getTimeStamp()*，*setAddress()*。

*BinaryMessage:*

这是 Message 的 *subinterface 子接口*。该接口用于收发二进制信息。设置和获取二进制内容数据的方法是 *setPayloadData()*，*getPayloadData()*。

*TextMessage:*

这是 Message 的 *subinterface 子接口*。该接口用于收发文本信息。设置和获取文本内容数据的方法是 *setPayloadText()*，*getPayloadText()*。

*MessageConnection:*

这是 GCF Connection 的 *subinterface 子接口*。该接口包含了一个创建新 *Message* 对象的工厂方法。*MessageConnection* 包括方法 *numberOfSegments()*，返回需要发送指定 *Message* 的下层协议的 segments 数量。该接口的其它方法是 *newMessage()*，*receive()*，*send()*，
*setMessageListener()*。

*MessageListener:*

这是个监听器接口，通过 *notifyIncomingMessage()*方法监听新短信通知。

## 使用 WMA 收发信息

使用 Wireless MessagingAPI 收发信息的代码一览：

**发送文本信息**

客户端模式连接，通过给 *Connector.open()*方法传入一个标识目标地址的字符串来建立。该方法返回一个 *MessageConnection* 对象，如下所示：

```
/*full destination address*/
String addr = “sms://+5555555500:5432”;
MessageConnection msgconn = (MessageConnection) Connector.open(addr);
```

地址部分包括由带国家区号的电话号码和规范指定的端口号。3GPP（The 3rd Generation Partnership Project）指定 SMS 端口号在 16000-16999 区间是应用程序可用的。一些手

机为系统保留了端口（受限端口）。Java ME 程序可能也无法使用这些系统保留端口。推荐开发 Java ME 应用使用指定范围内的非保留端口。

然后，给 *newMessage()*方法传递 *MessageConnection.TEXT_MESSAGE* 常量。该方法 *TextMessage* 返回对象，如下所示：

```
TextMessage txtmsg =
    (TextMessage)msgconn.newMessage(MessageConnection.TEXT_MESSAGE);
```

使用 *Textmessage* 对象提供文本内容。

```
txtmsg.setPayloadText("Samsung Mobile Innovator");
```

使用 *MessageConnection* 对象发送文本。

```
msgconn.send(msg);
```

**接收文本信息**

在 Java ME 中信息接收主要是绑定在一个端口上。要接收信息，须给 *Connector.open()* 方法传入接入点（协议标识，如端口号）的字符串参数，建立一个 server 模式连接。方法返回 *MessageConnection* 对象，如下所示：

```
/* address part*/
String addr = "sms://:5432";
MessageConnection msgconn = (MessageConnection) Connector.open(addr);
```

表示地址的字符串-URL 的格式是 messaging 协议规定的。

使用 *receive()*方法接收一条信息，该方法返回一个 Message 对象。

```
msg = conn.receive();
```

使用 *Textmessage* 检查是否收到信息，并获取内容文本。
确认收到的消息是否为 Textmessage 并获取文本内容

```
if (msg instanceof TextMessage) {
    TextMessage tmsg = (TextMessage)msg;
    String receivedText = tmsg.getPayloadText();
}
```

# 发送文本信息代码范例

下面的代码范例演示了如何发送一条文本信息。

代码假定在 JAD 中已加入 *portNo* 属性，如：*portNo: 5000*。

类: SendSMS

```
import javax.microedition.io.Connector;
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.lcdui.TextBox;
import javax.microedition.lcdui.TextField;
import javax.microedition.midlet.MIDlet;
import javax.wireless.messaging.MessageConnection;
import javax.wireless.messaging.TextMessage;

public class SendSMS extends MIDlet implements CommandListener, Runnable {

    private static final String EXIT = "Exit";
    private static final String OK = "Ok";
    private static final String BACK = "Back";
    private static final String SEND = "Send";
    private Form form;
    private TextField textfield;
    private TextBox textbox;
    private Display display;
    private Thread thread;

    /*this command is used to exit from the current application*/
    Command cmdExit = new Command(EXIT, Command.EXIT, 2);
    /*command for acceptance to display next screen*/
    Command cmdOk = new Command(OK, Command.OK, 1);
    /*command which allows to move to previous screen*/
    Command cmdBack = new Command(BACK, Command.BACK, 2);
    /*command to send the entered message to destination*/
    Command cmdSend = new Command(SEND, Command.OK, 1);
    private String getPort = null;

    public SendSMS() {
        /*gets the port number from JAD*/
        getPort = this.getAppProperty("portNo");
        display = Display.getDisplay(this);
        form = new Form("Enter Destination address");
        textfield = new TextField("Enter Destination address:", "", 20,
                TextField.PHONENUMBER);
        form.append(textfield);
        form.addCommand(cmdExit);
        form.addCommand(cmdOk);
        form.setCommandListener(this);
    }

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

    public void pauseApp() {
        /*all interupts can be handled here*/
    }

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

```
    public void commandAction(Command cmd, Displayable disp) {
        if (cmd == cmdExit && disp == form) {
            exitMidlet();
        } else if (cmd == cmdOk && disp == form) {
            textbox = new TextBox("Enter Message", null, 256, TextField.ANY);
            textbox.addCommand(cmdBack);
            textbox.addCommand(cmdSend);
            textbox.setCommandListener(this);
            display.setCurrent(textbox);
        } else if (cmd == cmdBack && disp == textbox) {
            display.setCurrent(form);
        } else if (cmd == cmdSend && disp == textbox) {
            sendSMS();
        }
    }

    public void sendSMS() {
        /*recomeneded to start the players in a separate thread*/
        thread = new Thread(this);
        thread.start();
    }

    public void run() {
        MessageConnection msgconn = null;
        try {
            /*full destination address*/
            String address = "sms://" + textfield.getString() + ":" + getPort;
            /*to open message connection*/
            msgconn = (MessageConnection) Connector.open(address);
            /*to send text message*/
            TextMessage textmsg = (TextMessage)
                    msgconn.newMessage(MessageConnection.TEXT_MESSAGE);
            /*pay load text passed here*/
            textmsg.setPayloadText(textbox.getString());
            /*send the message*/
            msgconn.send(textmsg);
        } catch (Exception exe) {
            exe.printStackTrace();
        } finally {
            try {
                msgconn.close();
            } catch (Exception io) {
                io.printStackTrace();
            }
            exitMidlet();
        }
    }

    public void exitMidlet() {
        destroyApp(false);
        notifyDestroyed();
    }
}
```

# 接收文本信息代码范例

下面的代码范例演示了如何接收一条文本信息。

代码假定在 JAD 中已加入 *portNo* 属性，如：*portNo: 5000*。

**类: ReceiveSMS**

```
import javax.microedition.io.Connector;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.midlet.MIDlet;
import javax.wireless.messaging.BinaryMessage;
import javax.wireless.messaging.Message;
import javax.wireless.messaging.MessageConnection;
import javax.wireless.messaging.MessageListener;
import javax.wireless.messaging.TextMessage;

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

    private static final String EXIT = "Exit";
    private Display display;
    private Thread thread;
    private MessageConnection msgconn;
    private Message msg;
    private String senderAddress;
    private Alert recMess;

    /*this command is used to exit from the current application*/
    Command cmdExit = new Command(EXIT, Command.EXIT, 2);
    private String getPort = null;

    public ReceiveSMS() {
        /*gets the port number from JAD*/
        getPort = this.getAppProperty("portNo");
        display = Display.getDisplay(this);
        recMess = new Alert("Message");
        recMess.setString("Waiting for SMS");
        recMess.setTimeout(Alert.FOREVER);
        recMess.addCommand(cmdExit);
        recMess.setCommandListener(this);
    }

    public void startApp() {
        display.setCurrent(recMess);
        try {
            /*port on which to receive sms*/
            String address = "sms://:" + getPort;
            /*open connection*/
```

```
            msgconn = (MessageConnection) Connector.open(address);
            /*set listening mode*/
            msgconn.setMessageListener(this);
            /*need to start the players in a separate thread to
            avoid dead lock*/
            thread = new Thread(this);
            thread.start();
        } catch (Exception exe) {
            exe.printStackTrace();
        }

    }

    public void pauseApp() {
        thread = null;
    /*all interupts can be handled here*/
    }

    public void destroyApp(boolean unconditional) {
        thread = null;

        if (msgconn != null) {
            try {
                msgconn.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

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

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

    public void notifyIncomingMessage(MessageConnection msgconn) {
        thread = new Thread(this);
        thread.start();
    }

    public void run() {
        try {
            /*gets message object*/
            msg = msgconn.receive();
            if (msg != null) {
                /*gets address of the message received*/
                senderAddress = msg.getAddress();
                recMess.setTitle("From: " + senderAddress);
                /*check for the type of message received*/
```

```
            if (msg instanceof TextMessage) {
                recMess.setString("Recieved Messsage: \n" + ((TextMessage)
msg).getPayloadText());
            } else {
                /*get the binary data of the message*/
                byte[] data = ((BinaryMessage) msg).getPayloadData();

            /*wite here code to convert binary to text message*/

            }
          }
       } catch (Exception exe) {
          exe.printStackTrace();
       }
    }
}
```