Mobile Phone Programming

J2ME – Wireless Messaging API

Dr. Christelle Scharff
Pace University, USA

http://atlantis.seidenberg.pace.edu/wiki/mobile2008

Objective

- Provide the necessary knowledge to write MIDlets that can send and receive messages

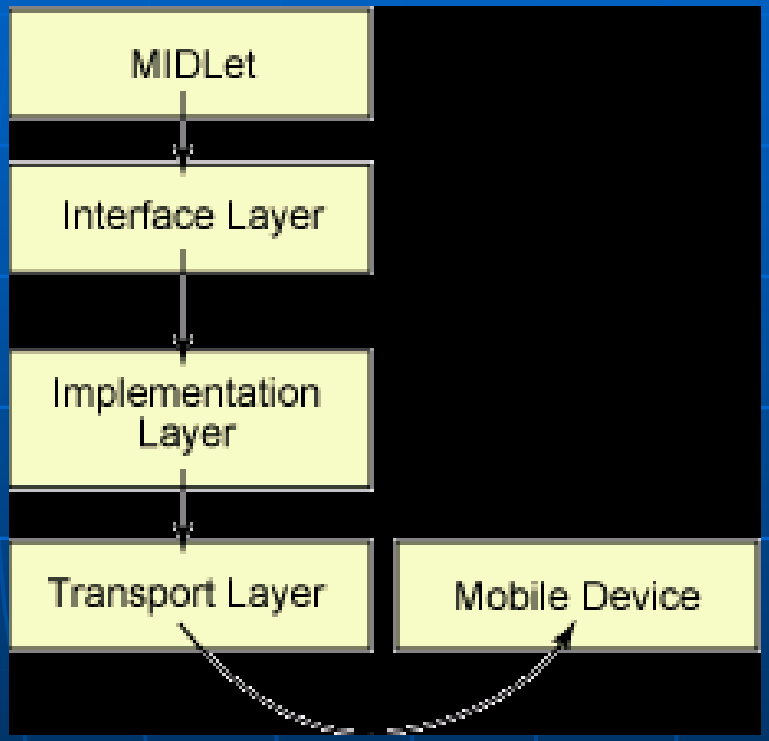
Wireless Messaging API

- WMA is an optional API on the top of CLDC (1.0 or 1.1) and CDC that enables MIDP applications to use the Short Message Service (SMS) and Cell Broadcast Service (CBS) protocols
- SMS messages are sent through a store-and-forward network
- SMS messages are up to 160 characters in most cases
- WMA 2.0 adds support for Multipart and Multimedia Message Service (MMS) messages (e.g., videos, images)
- Examples of applications: Chats, interactive games, event reminders, queries of corporate data

Wireless Messaging API

- WMA 1.1 - JSR 120
  - http://java.sun.com/products/wma/
  - http://paginas.fe.up.pt/~apm/JavaME/api/WirelessMessaging_1.1/javax/wireless/messaging/package-summary.html
- WMA 2.0 - JSR 205
  - http://developers.sun.com/mobility/midp/articles/wma2
  - http://www.forum.nokia.com/document/Java_Developers_Library_v2/GUID-71D2FC38-3A7C-4F69-9A2C-C307ABF33D6C/overview-summary.html

3-tier Wireless Messaging System Architecture

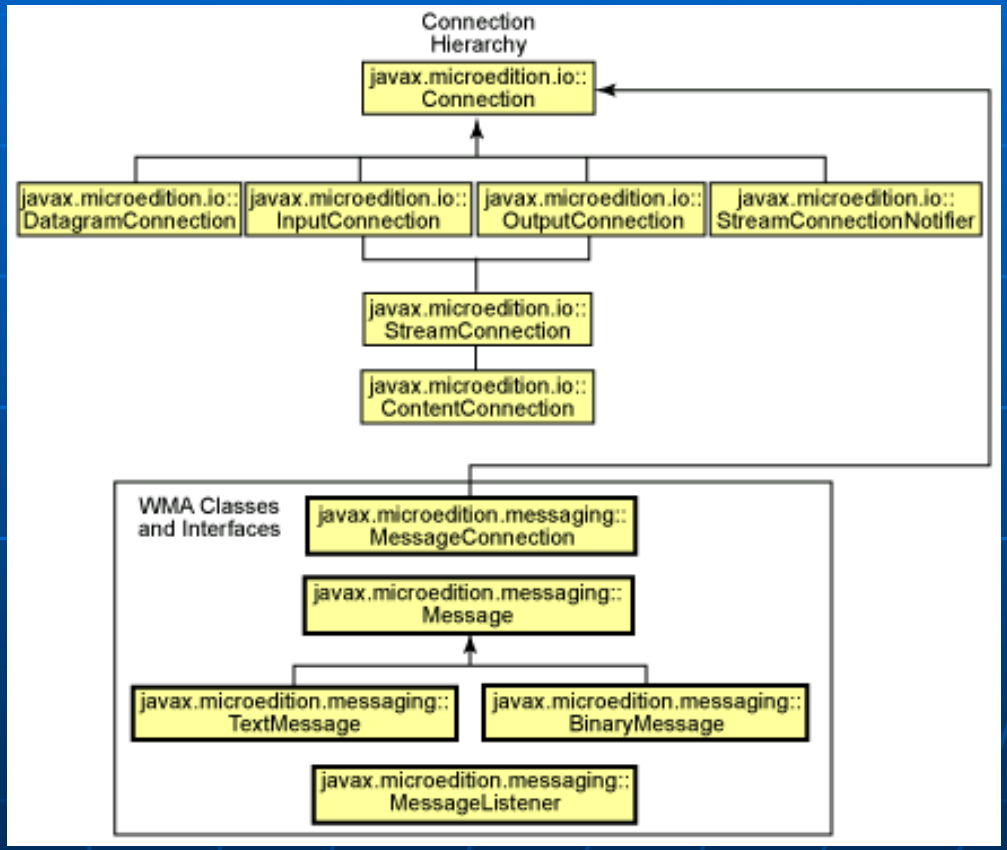


http://www.ibm.com/developerworks/library/wi-extendj2me/gif/

Connection Framework Extended with WMA APIs

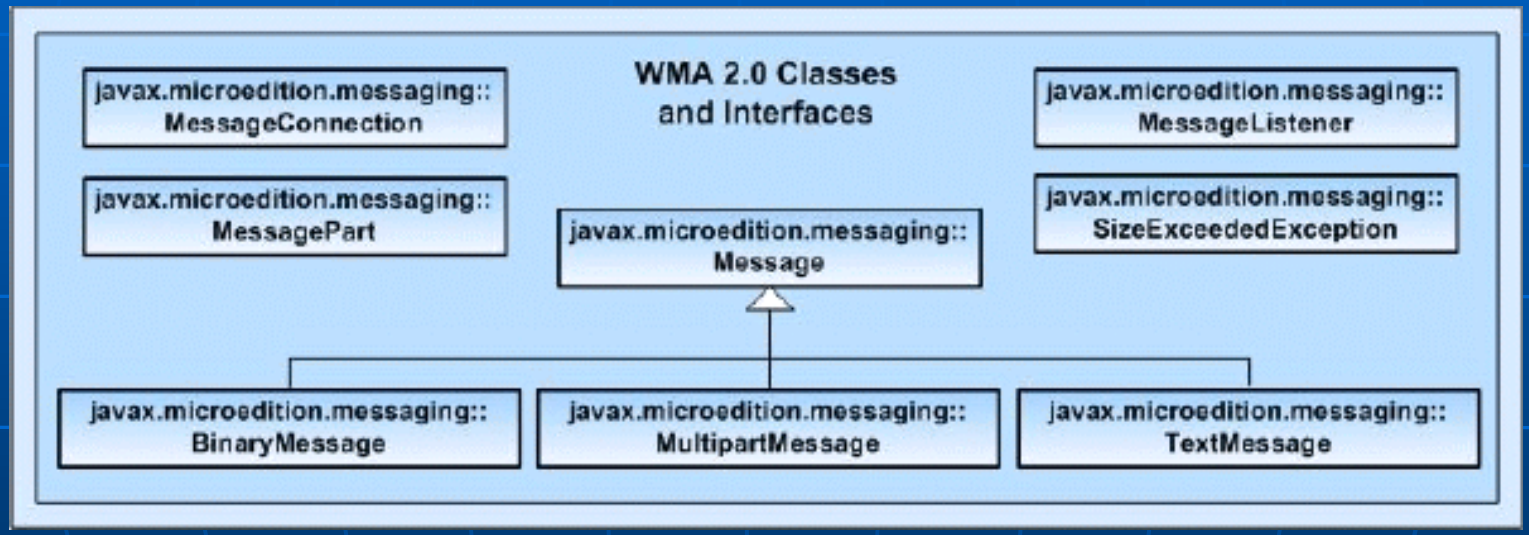
Java General Connection Framework (GCF)

javax.wireless.messaging defines messages, how to send/receive messages and message notification



http://www.ibm.com/developerworks/wireless/library/wi-prep/fig11-wma-gcf.gif/

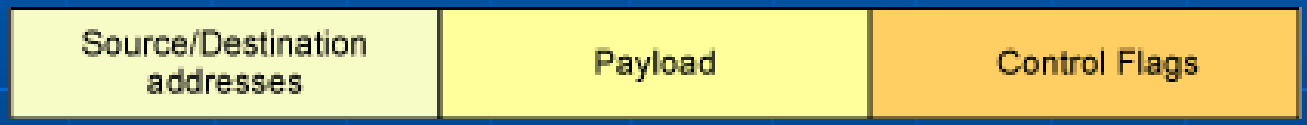
WMA 2.0



http://developers.sun.com/mobility/midp/articles/wma2/

Message Structure

- A message is composed of an address, a payload and control flags to send and block for message



http://www.ibm.com/developerworks/library/wi-extendj2me

URLs and Message Connections

- sms://+phone number – Client connection that sends messages to a phone number
- sms://+phone number:port number – Client connection that sends message to a port number at a phone number
- sms://:port number – Server connection that receives messages on a port . It can also send messages.
- cbs://:port number – Server connection that listens for inbound CBS messages on a port. It cannot send messages.

Opening a MessageConnection

- `MessageConnections` are used to send and receive messages, but also to create messages and obtain segmentation information on messages
- Example:
  - Client connection

```
MessageConnection clientConn =
(MessageConnection)
Connector.open("sms://5550000");
```

  - Server connection

```
MessageConnection serverConn =
(MessageConnection)Connector.open("sms:
/:1234");
```

Creating Messages

- Messages are created from the `MessageConnection` as empty messages using the `newMessage` method
  - `public Message newMessage(java.lang.String type)`
  - `public Message newMessage(java.lang.String type, java.lang.String address)`
  - `Types` are TEXT_MESSAGE, BINARY_MESSAGE and MULTIPART_MESSAGE
  - The payload of the message needs to be set

Message Properties

- Recipient/sender of the message if the message was received/sent
  - `public String getAddress()`
- Set the destination address for a message. The destination is set automatically when using a client mode `MessageConnection` to create the message
  - `public void setAddress(String addr)`
- Return when the message has been sent
  - `public Date getTimestamp()`

Sending a Text SMS Message

- Use of the `send` method of `MessageConnection`
  - `public void send(Message msg) throws java.io.IOException, java.io.InterruptedIOException`
- The method should be called in a new thread
- Example:

```
public void sendText(String address, String text) {
MessageConnection mc;
try {
   mc = (MessageConnection) Connector.open("sms://" + address);
   TextMessage tm = (TextMessage)
   mc.newMessage(MessageConnection.TEXT_MESSAGE);
   tm.setPayloadText(text);
   mc.send(tm);
} catch (IOException e) {e.printStackTrace();}}
```

Receiving a Text SMS Message

- Use the `receive` method of `MessageConnection`
  - `public Message` **`receive`**`() throws java.io.IOException, java.io.InterruptedIOException`
  - If there are no Message waiting, this method will block until either a message is received or the `MessageConnection` is closed
- Messages can be received using:
  - a blocking `receive` method – `receive` is called in its own thread
  - a non-blocking `receive` method – `receive` is managed using a `MessageListener` associated with the `Messageconnection` and should be called in a separate thread

Non-blocking Reception of SMS Messages

- The steps are to:
  - Make the required class implement the `MessageListener` interface
  - Register a `MessageListener` with the `Messageconnection`
    - `public void setMessageListener(MessageListener l) throws java.io.IOException`
  - Handle the callback in a thread in the `notifyIncomingMessage` method of the `MessageListener` interface
    - `public void notifyIncomingMessage(MessageConnection conn)`

Example

```
try {
   msg = mc.receive();
} catch (Exception e) {…}
// Process the received message
if (msg instanceof TextMessage) { // Text message
   TextMessage tmsg = (TextMessage) msg;
   msgReceived = tmsg.getPayloadText();
} else { if (msg instanceof BinaryMessage) { // Binary message
   BinaryMessage bmsg = (BinaryMessage) msg;
   Byte[] data = bmsg.getPayloadData();
    msgReceived = data.toString();
    } else {
      System.out.println("Other message");
      msgReceived = “Other";
}}
```

Multi-Threading

- Multi-threading allows a program to run concurrently
- Concurrent programming in Java is done with threads that exist within a process
- The code of a thread can be executed through a `Runnable` object that defines a single `run` method
- `Thread.start` is used to start the new thread
- `Thread.sleep` causes the current thread to suspend execution for a specified period
- The `join` method allows one thread to wait for the termination of another

Example

- Using an anonymous inner class:

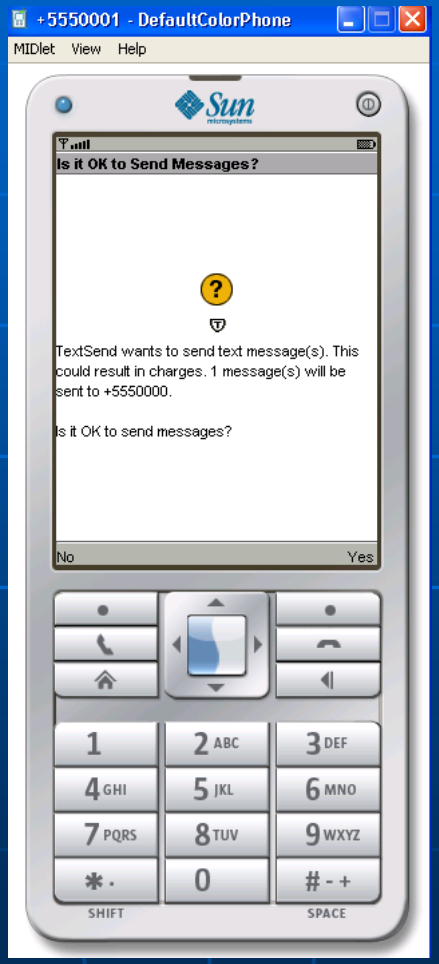
```
new Thread(new Runnable() {

 public void run() {
    …
}}).start();
```

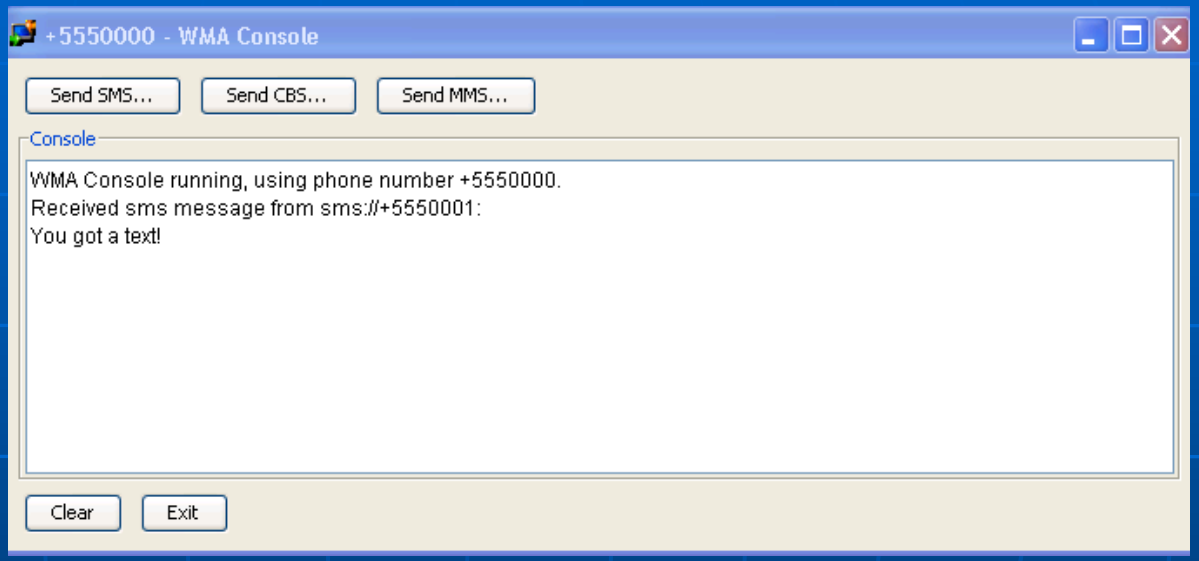
Testing WMA Applications with WTK 2.5

- Use the WMA console of WTK 2.5 to send SMS and monitor SMS receptions
  - Select `File / Utilities / WMA console`
  - Emulator phone numbers are of the form +5550000, +5550001…
  - When testing SMS answer `yes` to the security question
  - When testing the reception of SMS set the MIDlet permissions for SMS
    - Select the `Settings` of the project
    - Go to `Permissions`
    - Go to `Add` to add `javax.microedition.io.Connector.sms`

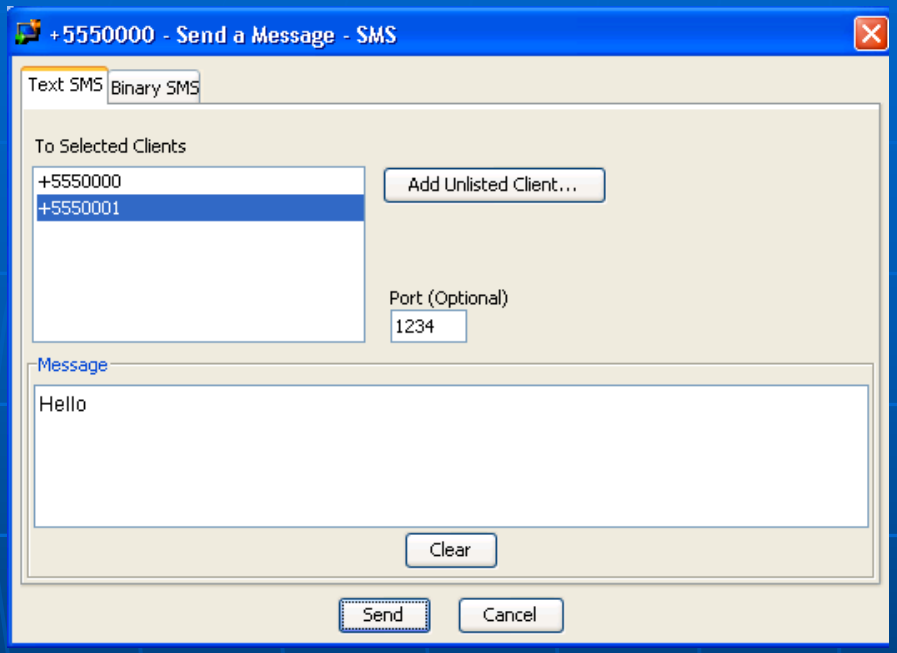
Security Question

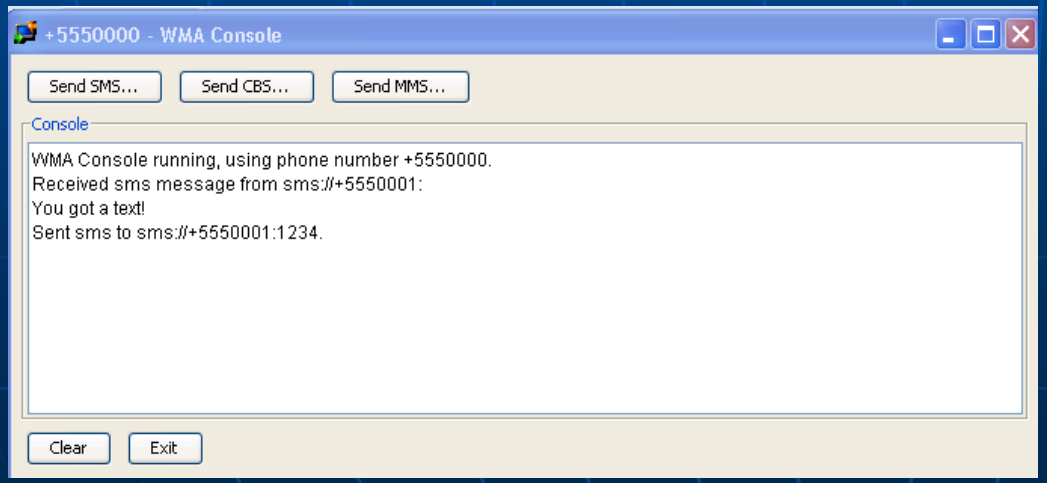


Sending a SMS



Receiving a SMS





References

- MIDP Profile API
  - http://java.sun.com/javame/reference/apis/jsr118
- JSR 120
  - http://jcp.org/jsr/details/120.jsp
- JSR 205
  - http://jcp.org/jsr/details/205.jsp
- The Wireless API 2.0
  - http://developers.sun.com/mobility/midp/articles/wma2

References

- Mobile P2P messaging, Part 1
  - http://www.ibm.com/developerworks/wireless/library/wi-p2pmsg
- Wireless Messaging API (WMA); JSR 120, JSR 205
  - http://java.sun.com/products/wma
- The J2ME Wireless Toolkit WMA Console
  - http://developers.sun.com/mobility/midp/ttips/wmaconsole
- Extend J2ME to Wireless Messaging
  - http://www.ibm.com/developerworks/library/wi-extendj2me

References

- Sun Java™ Wireless Toolkit for CLDC - Version 2.5.1
  - http://www.cs.kent.ac.uk/teaching/this_year/modules/CO/8/79/JavaME/wtkdocs/docs/UserGuide.pdf
- The Generic Connection Framework
  - http://developers.sun.com/mobility/midp/articles/genericframework
- Java Concurrency
  - http://java.sun.com/docs/books/tutorial/essential/concurrency

References

- To send SMS messages applications to phone
  - http://web21c.bt.com/