Fragmenting a binary message for sending over SMS using Java ME

Using IO classes such as Streams or Sockets is considered the standard way of transferring data over GPRS or other network protocols. But sometimes, especially in emerging markets, users may not have access to a data packet service.

Under this scenario, it is possible for MIDlets to exchange small binary messages by using the Wireless Messaging API, via SMSes. Please note that a typical SMS can not carry more than approximately 140 bytes. The data to be sent should be therefore fragmented by the sender and reconstructed by the receiving application, provided that the message exceeds the maximum binary load it can carry. The difference between sending a binary message instead of a text message from the Wireless Messaging API's point of view is in the construction of the message:

```
//Text Message
TextMessage message = (TextMessage)connection.newMessage(MessageConnection.TEXT_MESSAGE);
//Binary Message
BinaryMessage message = (BinaryMessage)connection.newMessage(MessageConnection.BINARY_MESSAGE);
```

Assuming that **data** is a byte array containing the binary message to be sent, **SMS_SIZE** is the maximum allowed binary payload and **number** is the number of the recipient, the following code snippet demonstrates how a binary message can be fragmented and sent over the operator's network as a sequence of SMSes:

```
    /**
     * Sends an SMS
     *
     * @param number where the SMS is sent to
     * @param data the binary message array to be sent
     */
    public void sendSMS(String number, byte[] data) {
        String port = "6553";
        //keeps track of how many parts the message is fragmented to
        int parts = data.length / (SMS_SIZE - 2) + 1;
        int offset = 0;
        for (int i = 0; i < parts; i++) {
            int length = Math.min(data.length - offset, SMS_SIZE - 2);
            byte[] dataToSend = new byte[2 + length];
            //the first bit indicates the ''i''th part
            dataToSend[0] = (byte) i;
            // out of a total of ''parts'' fragmented messages which is stored in the second bit of each message
            dataToSend[1] = (byte) parts;

            System.arraycopy(data, offset, dataToSend, 2, length);
            offset += length;
            try {
                //Opens the message connection to a specific port
                MessageConnection msgConn = (MessageConnection) Connector.open("sms://" + number + ":" + port);
                //sets the connection to binary type
                BinaryMessage msg = (BinaryMessage) msgConn.newMessage(MessageConnection.BINARY_MESSAGE, "sms://" + number + ":" + p
                //adds the fragmented payload
                msg.setPayloadData(dataToSend);
                msgConn.send(msg);
                msgConn.close();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (SecurityException e) {
                e.printStackTrace();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

}
```

See also

- Listening asynchronously for incoming SMS messages in Java ME
- Listening synchronously for incoming SMS messages in Java ME
- How to send a message to a given port as SMS with Java ME
- How to launch a MIDlet remotely via SMS or locally via an Alarm with PushRegistry in Java ME

(C) Copyright Nokia 2013. All rights reserved.

 (C) Copyright Nokia 2013. All rights reserved.