How to get System.out output from a MIDlet and save it to a file in S60 devices

Overview

When developing a MIDlet, it is very useful to get the output from the MIDlet somehow visible. This can be done for example simply by adding `System.out.println()` lines to the code. However, this method normally works only, when you use an emulator or use on-device debugging features of the S60 SDKs.

In S60 devices (S60 3rd Edition and newer) there is also a redirect method, which has been previously undocumented. It is possible to use URI scheme name "redirect", create an InputStream and read the other MIDlet's output from the stream for processing. For example, it is possible to create a String out of the output and show it to user in a TextBox, as it is done in the example MIDlet below. This output can be saved to a file (if FileConnection API is available) or to a RecordStore for viewing it later. In this updated example the output can also be sent as MMS message to a recipient.

The exact form of the URL varies from each other a little bit, depending on which S60 platform version is in use. In case of S60 3rd Edition FP1 or older the URL to be used is "redirect://" and in case of S60 3rd Edition FP2 or newer it is "redirect://test".

The redirecting input stream can be opened in different S60 platform devices as follows:

```
// for S60 3rd Edition FP1 devices or older
InputConnection inputConnection = (InputConnection) Connector.open("redirect://",
    Connector.READ);
inputStream = inputConnection.openInputStream();

// for S60 3rd Edition FP2 devices or newer
InputConnection inputConnection = (InputConnection) Connector.open("redirect://test",
    Connector.READ);
inputStream = inputConnection.openInputStream();
```

When the connection is open, the stream can be read as follows:

```
while (keepLogging) {
    try { // if there is no data this will block
        len = inputStream.read(data);
    } catch (IOException ioe) {
        ioe.printStackTrace();
    }
    if(len > 0){
        String str = new String(data, 0, len);
 // do something with the string
    }
}
```

Here is a full working sample MIDlet code below. The MIDPLogger MIDlet is launched first and then the MIDlet, which is to be followed. All the System.out.println(), System.err.println() and Throwable.printStackTrace() output should be shown in the TextBox of the MIDPLogger MIDlet. The output can then be saved to a file or send as MMS message. In this application the output is saved by default to C:/Data/Images/log.txt file. There is a Settings command for opening a Form for editing the MMS message address and the output file path.

Note: This is known to be not working in some devices. For example, currently no output is coming from a MIDlet run in N78 (sw 12.046).

Source code: MIDPLogger.java

```
/**
* Title: MIDPLogger
* Description: Redirect MIDlet for S60 devices
* Copyright: Copyright (c) 2008
*
* @author Nokia
* @version 1.10
*/

import java.io.IOException;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.TextBox;
import javax.microedition.lcdui.TextField;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
import javax.microedition.io.Connector;
import javax.microedition.io.InputConnection;
import java.io.InputStream;
import java.io.OutputStream;
import javax.microedition.io.file.FileConnection;
import javax.wireless.messaging.MessageConnection;
import javax.wireless.messaging.SizeExceededException;
import javax.wireless.messaging.MessagePart;
import javax.wireless.messaging.MultipartMessage;

public class MIDPLogger extends MIDlet implements Runnable, CommandListener {
    private static final int READ_BUFFER = 256;
    private static final String URL_S60_3_2 = "redirect://test";
```


(c) Copyright Nokia 2013. All rights reserved.

```
    private static final String URL_S60_3_1 = "redirect://";
    private boolean isAppStarted = false;
    private boolean keepLogging = true;
    private Command clearCommand;
    private Command saveCommand;
    private Command sendCommand;
    private Command settingsCommand;
    private Command exitCommand;
    private TextBox logTextBox;
    private SettingsForm form;
    private InputStream inputStream;
    private WriteData writeData;
    private Alert messageAlert;
    private Display display;
    private boolean s60_3_2 = true;
    private String platform;
    protected String address = "mms://";
    protected String filepath = "file:///C:/Data/Images/log.txt";
    private MessagePart textPart;
    private MultipartMessage message;
    private MessageConnection connection;
    private SettingsForm settingsForm;

    /**
    * startApp
    *
    * @throws MIDletStateChangeException
    */
    protected void startApp() throws MIDletStateChangeException {
        if (!isAppStarted) {
            Thread t = new Thread(this);
            t.start();
            isAppStarted = true;
        }
    }

    /**
    * pauseApp Method is called when the midlet is paused
    */
    protected void pauseApp() {
    }

    /**
    * destroyApp
    *
    * @param b
    * @throws MIDletStateChangeException
    */
    protected void destroyApp(boolean b)throws MIDletStateChangeException {
    }

    /**
     * Initializes the MIDlet UI: creates and adds the commands and sets the
     * logTextBox as current Displayable.
     */
    private void initUI() {
        display = Display.getDisplay(this);
        logTextBox = new TextBox("MIDPLogger", "", 10000, TextField.ANY);
        saveCommand = new Command("Save in file", Command.SCREEN, 1);
        logTextBox.addCommand(saveCommand);
        sendCommand = new Command("Send log as MMS", Command.SCREEN, 2);
        logTextBox.addCommand(sendCommand);
        settingsCommand = new Command("Settings", Command.SCREEN, 3);
        logTextBox.addCommand(settingsCommand);
        clearCommand = new Command("Clear", Command.CANCEL, 1);
        logTextBox.addCommand(clearCommand);
        exitCommand = new Command("Exit", Command.EXIT, 1);
        logTextBox.addCommand(exitCommand);
        logTextBox.setCommandListener(this);
        display.setCurrent(logTextBox);
        writeData = new WriteData();
    }

    /**
     * Creates the input stream and also checks, if the device is S60 3rd Ed. FP2
     * device or newer. Note: "java_build_version" information was added to the
     * "microedition.platform" property value in S60 3rd Ed. FP2 devices.
     */
    private void createInputStream() {
        platform = System.getProperty("microedition.platform");
        if(platform != null && platform.indexOf("java_build_version") > 0){
            s60_3_2 = true;     // S60 3.2 or higher
        }
        else if (platform != null) {
            s60_3_2 = false;    // S60 3.1 or older version
        }
        else return;
        createRedirectInputStreamCon();
    }

    /**
     * This method selects the url to be used and informs user about the
     * S60 platform release of the device.
     */
    private void createRedirectInputStreamCon() {
        try {
            if (s60_3_2) {
                InputConnection inputConnection = (InputConnection) Connector.open(URL_S60_3_2,
                    Connector.READ);
                inputStream = inputConnection.openInputStream();
                logTextBox.insert("Platform S60 3rd Ed. FP2 or newer.\n", logTextBox.size());
            }
            else {
                InputConnection inputConnection = (InputConnection) Connector.open(URL_S60_3_1,
                    Connector.READ);
                inputStream = inputConnection.openInputStream();
                logTextBox.insert("Platform S60 3rd Ed. FP1 or older.\n", logTextBox.size());
            }
        } catch (IOException ioe) {
            logTextBox.insert("IOException: " + ioe.getMessage() + "\n", logTextBox.size());
        }
    }

    public void commandAction(Command command, Displayable displayable) {
        if (command == clearCommand) {
            logTextBox.setString("");
        }
        else if (command == saveCommand) {
            Thread thread = new Thread(writeData);
            thread.start();
```


(c) Copyright Nokia 2013. All rights reserved.

```
        }
        else if (command == sendCommand) {
            try {
                connection = (MessageConnection)Connector.open(address);
                String text = logTextBox.getString();   // Message text from TextBox
                String textContentId = "text";          // Mandatory contentId
                textPart = new MessagePart(text.getBytes(), 0, text.length(), "text/plain",
                    textContentId, null, null);
                message = (MultipartMessage)connection.newMessage("multipart");
                // "multipart" equals MessageConnection.MULTIPART_MESSAGE
                message.setSubject("Log from MIDPLogger");
                message.setAddress(address);
                message.addMessagePart(textPart);
                sendMMS(message);
            } catch (SizeExceededException see) {
                logTextBox.insert("SizeExceededException: " + see.getMessage() + "\n",
                    logTextBox.size());
            } catch (IOException ioe) {
                logTextBox.insert("IOException: " + ioe.getMessage() + "\n", logTextBox.size());
            }
        }
        else if (command == settingsCommand) {
            settingsForm = new SettingsForm("Settings", this);
            Display.getDisplay(this).setCurrent(settingsForm);
        }
        else if (command == exitCommand) {
            keepLogging = false;
            this.notifyDestroyed();
        }
    }

    private void sendMMS(final MultipartMessage message) {
        // Send the message on its own thread of execution
        logTextBox.insert("sending MMS message to: " + address + "\n", logTextBox.size());
        Thread messageThread = new Thread() {
            public void run() {
                try {
                    connection.send(message);
                } catch (IOException ioe) {
                    logTextBox.insert("IOException: " + ioe.getMessage() + "\n",
                        logTextBox.size());
                } catch (IllegalArgumentException iae) {
                    logTextBox.insert("IllegalArgumentException: " + iae.getMessage() + "\n",
                        logTextBox.size());
                } catch (SecurityException se) {
                    logTextBox.insert("SecurityException: " + se.getMessage() + "\n",
                        logTextBox.size());
                }
            }
        };
        messageThread.start();
    }

    public void run() {
        initUI();
        createInputStream();
        readData();
    }

    /**
     * Reads data from the input stream and displays it
     */
    private void readData() {
        if(inputStream == null){
            if (platform == null) showAlert("Error", "microedition.platform = null!",
                AlertType.ERROR);
            else showAlert("Error", "The url redirect:// is not supported", AlertType.ERROR);
            return;
        }
        readDataAndDisplay();
    }

    /**
     * Reads data MIDlet system out and error stream and shows it to user
     */
    private void readDataAndDisplay() {
        // read data from stream and display it in text box
        byte[] data = new byte[READ_BUFFER];
        int len = -1;
        while (keepLogging) {
            try { // if there is no data this will block
                len = inputStream.read(data);
            } catch (IOException ioe) {
                logTextBox.insert("IOException: " + ioe.getMessage() + "\n", logTextBox.size());
            }
            if(len > 0){
                String str = new String(data, 0, len);
                int maxSize = logTextBox.getMaxSize();
                if (logTextBox.size() >= maxSize) {
                    logTextBox.setMaxSize(maxSize + READ_BUFFER * 10);
                }
                logTextBox.insert(str, logTextBox.size());
            }
        }
    }

    private void showAlert(String title, String message, AlertType alertType){
        if(null == messageAlert){
            messageAlert = new Alert(message);
        }
        messageAlert.setTitle(title);
        messageAlert.setString(message);
        messageAlert.setType(alertType);

        if(alertType == AlertType.ERROR) {
            messageAlert.setTimeout(Alert.FOREVER);
        }
        else {
            messageAlert.setTimeout(3000);
        }
        display.setCurrent(messageAlert);
    }

    protected void showTextBox() {
        logTextBox.insert("Address is now: " + address + "\n", logTextBox.size());
        logTextBox.insert("Log file path is now: " + filepath + "\n", logTextBox.size());
        Display.getDisplay(this).setCurrent(logTextBox);
        if (form != null) form = null;
    }
```


(c) Copyright Nokia 2013. All rights reserved.

```
    class WriteData implements Runnable {
        WriteData() {}

        public void run() {
            saveData();
        }

        /**
         * Saves data to file system
         */
        private void saveData() {
            try {
                String data = logTextBox.getString();
                FileConnection fileOutputConnection = (FileConnection) Connector.open(filepath);
                // Create file specified in url if it does not exist
                if (!fileOutputConnection.exists()) {
                    // create the directory if it does not exist
                    String path = "file://" + fileOutputConnection.getPath();
                    FileConnection fc = (FileConnection) Connector.open(path);
                    if (!fc.exists()) {
                        fc.mkdir();
                    }
                    fc.close();
                    // create the file if it doesn't exist
                    fileOutputConnection.create();
                }
                OutputStream outputStream = fileOutputConnection.openOutputStream();
                outputStream.write(data.getBytes());
                outputStream.close();
                fileOutputConnection.close();
                logTextBox.insert("Log saved to " + filepath.substring(8) + "\n",
                    logTextBox.size());
            } catch (IOException ioe) {
                logTextBox.insert("IOException: " + ioe.getMessage() + "\n",
                    logTextBox.size());
            } catch (SecurityException se) {
                logTextBox.insert("SecurityException: " + se.getMessage() + "\n",
                    logTextBox.size());
            }
        }
    }
}
```

Source code: SettingsForm.java

```
/**
* Title: SettingsForm
* Description: A Form for editing the log file path and the phone number for the MMS message.
* Copyright: Copyright (c) 2008
*
* @author Nokia
* @version 1.10
*/

import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.TextField;

public class SettingsForm extends Form implements CommandListener {
    private MIDPLogger midlet;
    private Command backCommand;
    private Command exitCommand;
    private TextField mmsField;
    private TextField fileField;

    public SettingsForm(String title, MIDPLogger midlet) {
 super(title);
 this.midlet = midlet;
 backCommand = new Command("Back", Command.BACK, 1);
 exitCommand = new Command("Exit", Command.EXIT, 1);
 this.addCommand(backCommand);
 this.addCommand(exitCommand);
 this.setCommandListener(this);
        mmsField = new TextField("Phone number", "+358", 50, TextField.PHONENUMBER);
        this.append(mmsField);
        fileField = new TextField("Log file path", "file:///C:/Data/Images/log.txt",
            100, TextField.ANY);
        this.append(fileField);
    }

    public void commandAction( Command c, Displayable d) {
 if (c == backCommand) {
            midlet.address = "mms://" + mmsField.getString();
            midlet.filepath = fileField.getString();
     midlet.showTextBox();
 }
 if (c == exitCommand) {
     midlet.notifyDestroyed();
 }
    }
}
```

There are also the MIDPLogger.jad and MIDPLogger.jar files available here, they are signed to trusted 3rd party domain. Because of this it is possible to switch off the security prompt, which is shown, when the log is written to a file. The security setting can be changed in Application Manager (MIDPLogger -> Open -> Set "Read user data" and "Edit user data" to "Always allowed").

Note: The log send as MMS message can be saved to a file as follows: In Messaging application open the MMS message. Select "Objects" from Options menu. The log text should be shown there as something like "midp_wma20_attachment1". The message's text part can then be saved to Notes or be sent forward.

Example application

(c) Copyright Nokia 2013. All rights reserved.

- MIDPLogger_v1_10_1_src.zip containing sources, MIDPLogger.jad and MIDPLogger.jar files

 (c) Copyright Nokia 2013. All rights reserved.

(c) Copyright Nokia 2013. All rights reserved.