

NFC_Send_UID_from_a_tag_to_a_Server

The following example consists out of two projects. One Java ME MIDlet, that uses a Nokia 6131 to read the UID of a tag and send it to a socket server. The second is very simple implementation of a Java SE socket server, that receives the UID and sends back an ok-message.

The Java ME MIDlet

Important: Change the content of the variable "ip" according to your setup:

```
package at.nfcresearch.examples.uiddemo;

// Packages for contactless Communcation
import javax.microedition.contactless.ContactlessException;
import javax.microedition.contactless.DiscoveryManager;
import javax.microedition.contactless.TargetProperties;
import javax.microedition.contactless.TargetListener;
import javax.microedition.contactless.TargetType;

// Packages for GUI Stuff
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Form;

// Data Connection
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import javax.microedition.io.ConnectionNotFoundException;
import javax.microedition.io.Connector;
import javax.microedition.io.SocketConnection;

import javax.microedition.midlet.*;

public class UIDClient extends MIDlet implements TargetListener, CommandListener {
    // Display Stuff
    private Command exitCommand;
    private Form form;

    // change this IP Address according to your setup!!!
    private final static String ip = "socket://194.187.178.164:49997";

    public UIDClient() {

        // Create the GUI
        exitCommand = new Command("Exit", Command.EXIT, 1);
        form = new Form("UIDClient");

        form.addCommand(exitCommand);
        form.append("Touch Tag to read ID.");
        form.setCommandListener(this);

        // Registration of the TargetListener for external contactless
```

NFC_Send_UID_from_a_tag_to_a_Server

```
// Targets (in this RFID_TAG).
try {
    DiscoveryManager dm = DiscoveryManager.getInstance();
    dm.addTargetListener(this, TargetType.NDEF_TAG);

} catch (ContactlessException ce) {
    displayAlert("Unable to register TargetListener: " + ce.toString(), AlertType.ERROR);
}

}

public void startApp() {
    Display.getDisplay(this).setCurrent(form);
}

public void pauseApp() {
}

public void destroyApp(boolean unconditional) {
}

/**
 * Implementation of the Call-Back Function of the TargetListener
 * @param targetProperties: Array of Targets found by the Phone
 *
 */
public void targetDetected(TargetProperties[] targetProperties) {

    // in case no targets found, exit the method
    if (targetProperties.length == 0) {
        return;
    }

    // show the UID of the first tag found
    TargetProperties tmp = targetProperties[0];
    displayAlert("UID read: " + tmp.getUid(), AlertType.INFO);

    // send data to server
    try {
        byte[] result = postViaSocketConnection(ip, tmp.getUid().getBytes());
        form.append("\nServer says: " + arrayToHex(result));
    } catch (Exception e) {
        form.append("Error: " + e.toString());
    }

}

/**
 * Implementation of the Call-Back function of the CommandListener
 * @param command: command key pressed
 * @param displayable: associated displayable Object
 */
public void commandAction(Command command, Displayable displayable) {
    if (command == exitCommand) {
        DiscoveryManager dm = DiscoveryManager.getInstance();
        dm.removeTargetListener(this, TargetType.NDEF_TAG);
        destroyApp(false);
        notifyDestroyed();
    }

}

}
```

NFC_Send_UID_from_a_tag_to_a_Server

```
private void showAlert(String error, AlertType type) {
    Alert err = new Alert(form.getTitle(), error, null, type);
    Display.getDisplay(this).setCurrent(err, form);
}

// Establish a connectoin to a server.
private byte[] postViaSocketConnection(String connectURL, byte[] inData) throws IOException,

    SocketConnection sc = (SocketConnection) Connector.open(connectURL);
    sc.setSocketOption(SocketConnection.LINGER, 5);

    InputStream is = sc.openInputStream();
    OutputStream os = sc.openOutputStream();

    os.write(inData);
    int ch = 0;

    byte[] readin = new byte[0xFF];
    ch = is.read(readin);

    byte[] readReally = new byte[ch];
    System.arraycopy(readin, 0, readReally, 0, ch);

    if (is != null) {
        is.close();
    }
    if (os != null) {
        os.close();
    }
    if (sc != null) {
        sc.close();
    }
    return readReally;
}

private static String arrayToHex(byte[] data) {
    StringBuffer sb = new StringBuffer();

    for (int i = 0; i < data.length; i++) {
        String bs = Integer.toHexString(data[i] & 0xFF);
        if (bs.length() == 1) {
            sb.append(0);
        }
        sb.append(bs);
        sb.append((i + 1) % 4 == 0 ? '\n' : ' ');
    }

    return sb.toString();
}
}
```

Java SE socket server implementation

The second application in the socket server (J2SE!!), that receives the data and sends back a 90 00 in order to confirm that the data was received.

Make sure that your firewall is not blocking the port. On a Windows machine you can use `netstat -a in`

NFC_Send_UID_from_a_tag_to_a_Server

order to see all the open ports on your machine at the moment.

```
package at.nfcresearch.examples.uiddemo;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.UnknownHostException;

public class Server implements Runnable {

    private Socket m_socket;

    public Server(Socket socket) {
        m_socket = socket;
        new Thread(this).start();
    }

    public void run() {
        System.out.println("Thread opened -- Waiting for requests; ");

        // hier beginnen wir etwas einzulesen.
        try {

            InputStream input_stream;
            OutputStream output_stream;

            // getting the pointer to the streams
            input_stream = m_socket.getInputStream();
            output_stream = m_socket.getOutputStream();

            // reading data;
            byte bufferin[] = new byte[0xff];
            int len_cli = input_stream.read(bufferin);
            byte[] cli_buffer = new byte[len_cli];
            System.arraycopy(bufferin, 0, cli_buffer, 0, len_cli);

            System.out.println(len_cli + " Bytes read");

            // print the read bbuffer
            System.out.println(new String(cli_buffer));

            // construct the return value
            byte bufferout[] = new byte[2];
            bufferout[0] = (byte) 0x90;
            bufferout[1] = (byte) 0x00;

            System.out.println("Information sent back;");

            output_stream.write(bufferout);
            output_stream.flush();

            input_stream.close();
            output_stream.close();

            // wait a bit, so that the data is sent for sure (!)
            Thread.sleep(100);

            m_socket.close();
        }
    }
}
```

NFC_Send_UID_from_a_tag_to_a_Server

```
} catch (InterruptedException e) {
    System.out.println("Error on Seloop Sleep: " + e.toString());
} catch (IOException ie) {
    System.out.println("Error on opening the socket " + ie.toString());
}

System.out.println("connection to Tcp:/" + m_socket.getRemoteSocketAddress().toString()+
}

public static void main(String[] args) {

    System.out.println("Server Started .....");
    ServerSocket tcp_server = null;
    InetAddress localhost = null;
    int serverPort = 49997;

    try {

        localhost = InetAddress.getLocalHost();

        // Öffnen eines neuen ServerSockets
        tcp_server = new ServerSocket(serverPort, 10, localhost);
        System.out.println("TcpEchoServer waiting for a connection at " + tcp_server.getLocal

    } catch (UnknownHostException e) {
        System.out.println("Error on opening the Socket Server: " + e.toString());
        System.exit(-1);
    } catch (IOException e) {
        System.out.println("OpenServerSocket faild: " + e.toString());
        System.exit(-1);
    }

    // start endless loop
    while (true) {

        // Open Server Socket
        Socket tcp_socket = null;
        try {

            // wait for incoming connection; this Method is blocking!
            tcp_socket = tcp_server.accept();
            System.out.println("-----\nConnection to Tcp:/" + tcp_socket.getRemoteSocketAddre

            // start a new thread to process this information
            new Server(tcp_socket);

        } catch (IOException e) {
            System.out.println("Error on trying to establish the connection : " + e.toString());
            System.exit(-1);
        }
    }
}
}
```