

## Simple\_NFCIP\_peer\_to\_peer\_chat

Example of NFCIP P2P communication in [Nokia 6131 NFC](#). User can exchange messages with other phones by touching them. By default [MIDlet](#) is in reading state and when user wants to write something [MIDlet](#) goes to writing state.

*Note that [NFCIPConnection](#) class used in this example is proprietary implementation for [Nokia 6131 NFC](#) so this example might not work in any other phone.*

```
package com.nokia.nfc.sample.app;

import java.io.IOException;

import javax.microedition.io.Connector;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
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.microedition.midlet.MIDletStateChangeException;

import com.nokia.nfc.p2p.NFCIPConnection;

/* Example of P2P communication in Nokia 6131 NFC. User can exchange messages
 * with other phones by touching them. By default MIDlet is in reading state and
 * when user wants to write something MIDlet goes to writing state.
 */
public class P2PChat extends MIDlet implements Runnable, CommandListener {

    // Initiator URL, used to make the connection to other phone
    private static final String INITIATOR_URL="nfc:rf?type=nfcip;mode=initiator";
    // Target URL, used to listen for incoming connections
    private static final String TARGET_URL = "nfc:rf?type=nfcip;mode=target";

    // UI elements
    private Display display;
    private Form form;
    private TextBox writeTb;
    // Commands
    private Command writeCmd;
    private Command cancelCmd;
    private Command exitCmd;

    // Is the MIDlet in reading or writing state
    boolean reading = true;

    // MIDlet UI initialization and starting the thread
    protected void startApp() throws MIDletStateChangeException {

        // Initialize UI
        display = Display.getDisplay(this);
        form = new Form("Form");
        writeCmd = new Command("Write", Command.ITEM, 1);
        cancelCmd = new Command("Cancel", Command.CANCEL, 1);
        exitCmd = new Command("Exit", Command.EXIT, 1);
        form.addCommand(writeCmd);
        form.addCommand(exitCmd);
        form.setCommandListener(this);
    }
}
```

## Simple\_NFCIP\_peer\_to\_peer\_chat

```
display.setCurrent(form);

// Start the thread
Thread thread = new Thread(this);
thread.start();

}

// Thread to exchange data between phones
public void run() {
    NFCIPConnection conn = null;
    // Reading state
    if (reading) {
        // Read while in reading state
        while (reading) {
            try {
                // Start waiting for other phone to initiate connection,
                // blocks until connection made or another NFCIP connection
                // open called
                conn = (NFCIPConnection) Connector.open(TARGET_URL);
                // Receive data from connection
                byte[] data = conn.receive();
                // Send empty message to connection
                conn.send(null);
                // Append received string to form
                form.append(">> " + new String(data) + "\n");
                // Close connection
                conn.close();
            } catch (Exception ex) {
                try {
                    if (conn != null) {
                        conn.close();
                    }
                } catch (IOException e) {
                }
            }
        }
        // Writing state
    } else {
        // Write while in writing state. If writing fails MIDlet will do the
        // writing loop until it succeeds or changed back to reading state.
        while (!reading) {
            try {
                // Try to connect another phone, blocks until connection
                // made or another NFCIP connection open called
                conn = (NFCIPConnection) Connector.open(INITIATOR_URL);
                // Send string from TextBox to connection
                conn.send(writeTb.getString().getBytes());
                // Receive reply
                conn.receive();
                // Append sent message to form
                form.append("<< " + writeTb.getString() + "\n");
                // Close connection
                conn.close();
                display.setCurrent(form);
                read();
            } catch (Exception ex) {
                try {
                    if (conn != null) {
                        conn.close();
                    }
                } catch (IOException e) {
                }
            }
        }
    }
}
```

## Simple\_NFCIP\_peer\_to\_peer\_chat

```
        }
    }
}

public void commandAction(Command c, Displayable d) {
    if (c == writeCmd && d == form) {
        writeTb();
    } else if (c == writeCmd && d == writeTb) {
        write();
    } else if (c == cancelCmd) {
        read();
    } else if (c == exitCmd) {
        notifyDestroyed();
    }
}

// Go to read state
private void read() {
    reading = true;
    display.setCurrent(form);
    // Start new thread that kills the current connection
    Thread thread = new Thread(this);
    thread.start();
}

// Display TextBox
private void writeTb() {
    writeTb = new TextBox("Text to write", null, 255, TextField.ANY);
    writeTb.addCommand(writeCmd);
    writeTb.addCommand(cancelCmd);
    writeTb.setCommandListener(this);
    display.setCurrent(writeTb);
}

// Go to write state
private void write() {
    reading = false;
    // Show alert
    Alert a = new Alert("Writing", "Touch to write", null, AlertType.INFO);
    a.addCommand(cancelCmd);
    a.setTimeout(Alert.FOREVER);
    a.setCommandListener(this);
    display.setCurrent(a);
    // Start new thread that kills the current connection
    Thread thread = new Thread(this);
    thread.start();
}

protected void destroyApp(boolean unconditional)
    throws MIDletStateChangeException {
}

protected void pauseApp() {
}
}
```