The code shows a basic MIDlet that writes an NDEF (EXTERNAL_RTD) to a tag, in order to start an application. The application to be started requires the following line in the JAD-File:

```
MIDlet-Push-1: ndef:external_rtd?name=urn:nfc:ext:nfc-research.at:pushreg,at.nfcresearch.wima.exa
```

This example does NOT work in the Emulator of the Nokia 6131, but only on the phone! You can download ZIP with two NetBeans Project to try out the functionality: File:NDEFPush.zip.

```
package at.nfcresearch.wima.examples;

import javax.microedition.contactless.ContactlessException;
import javax.microedition.contactless.DiscoveryManager;
import javax.microedition.contactless.TargetListener;
import javax.microedition.contactless.TargetProperties;
import javax.microedition.contactless.TargetType;
import javax.microedition.contactless.ndef.NDEFMessage;
import javax.microedition.contactless.ndef.NDEFRecord;
import javax.microedition.contactless.ndef.NDEFRecordType;
import javax.microedition.contactless.ndef.NDEFTagConnection;
import javax.microedition.io.Connector;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class PushRegTagWriter extends MIDlet implements TargetListener, CommandListener {

    // Display Items
    private Command exitCommand;
    private StringItem text;
    private Form form;

    public PushRegTagWriter() {

        // Display Items
        exitCommand = new Command("Exit", Command.EXIT, 1);
        form = new Form("NFC-Research.at: PushRegistry Tag Writer");

        form.addCommand(exitCommand);
        form.append("Touch Tag to write Ext-NDEF for PushRegistry start.");
        form.setCommandListener(this);


        // Registration of the Targetlistener
        // Look for ANY Type of RFID Tag that can be
        // read by the phone.
        try {
            DiscoveryManager dm = DiscoveryManager.getInstance();
            dm.addTargetListener(this, TargetType.RFID_TAG);
        } catch (ContactlessException ce) {
            displayAlert("Unable to Register Targetlistener:" + ce.toString(), AlertType.ERROR);
            Display.getDisplay(this).setCurrent(form);
        }
    }

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

    protected void pauseApp() {
    }

    protected void destroyApp(boolean bool) {
```

```
}

public void commandAction(Command cmd, Displayable disp) {
    if (cmd == exitCommand) {
        DiscoveryManager dm = DiscoveryManager.getInstance();
        dm.removeTargetListener(this, TargetType.RFID_TAG);
        destroyApp(false);
        notifyDestroyed();
    }


}

/**
 * Method called by the TargetListener in case a tag
 * was detected.
 * @param prop contains the array of tags found.
 */
public void targetDetected(TargetProperties[] prop) {

    // if there is no information available on the tags
    // found, exit Method.
    if (prop.length == 0) {
        return;
    }

    // Create a new NDEF Connection in order
    // to write an NDEF EXT to the tag.
    NDEFTagConnection ndconn = null;
    try {
        // create an ndef tag connection.
        String ur = prop[0].getUrl(Class.forName("javax.microedition.contactless.ndef.NDEFTag

        // open the NDEF Connection
        ndconn = (NDEFTagConnection) Connector.open(ur);

        // now create the NDEFRecord
        NDEFRecordType myType = new NDEFRecordType(NDEFRecordType.EXTERNAL_RTD, "urn:nfc:ext:
        NDEFRecord myRec;

        // create Data-Container for Tag
        myRec = new NDEFRecord(myType, null, null);

        NDEFRecord[] myRecArray = new NDEFRecord[]{myRec};
        NDEFMessage myMessage = new NDEFMessage(myRecArray);

        // write Information to Tag
        ndconn.writeNDEF(myMessage);

        displayAlert("Writing succeeded!", AlertType.INFO);
    } catch (Exception e) {
        displayAlert("Error Writing to Tag! " + e.toString(), AlertType.ERROR);
    }
    try {
        ndconn.close();
    } catch (Exception e) {
    }
}

private void displayAlert(String error, AlertType type) {
    Alert err = new Alert(form.getTitle(), error, null, type);
```

```
            Display.getDisplay(this).setCurrent(err, form);
        }
    }
```