ScanFiles.java

```
// ScanFiles
// Demonstration of scanning a directory for EFs within an GSM Smart Card (SIM)
// using the OCF 1.2 with PC/SC to transmit the data
// File: ScanFiles.java
//
// This program uses the PassThruCardService from OCF opencard.opt.util package.
// Additional informations and reference implementation: www.opencard.org.
// Testet with Towitoko Chipdrive micro on USB port and the appropriate PC/SC dri
// from Towitoko on Windows 98. Testet with Omnikey CardMan Dongle 6020 on USB po
// and the appropriate PC/SC drivers from Omnikey on Windows 2000. Testet with
// JBuilder 6 and JDK 1.3.1 and a GSM 11.11 Smart Card.
//
// Uses OCFPCSC1.DLL, OCFPCSC1.DLL, pcsc-wrapper-src.jar, base-core.jar, base-opt
// from OCF 1.2
// The content of the file "opencard.properties" must be changed to:
// OpenCard.terminals = com.ibm.opencard.terminal.pcsc10.Pcsc10CardTerminalFactor
// For understanding the mechanisms of OCF it is useful to look first into the
// SimpleOCF example.
//
// Performance: The scan needs about 50 sec for 1000 FIDs.
// Remark:      If there is a by FID selectable DF within the search area it coul
//              be that this DF will be select by the search process. This will
//              cause an unpredictable behaviour of the search process. This prob
//              can avoid with a reset before every selection by FID, but this sl
//              down the performance.
//
// 18. Dez. 2003 - V 8, rw: minor changes concerning the user interface,
//                          4th published version
// 15. Dez. 2003 - V 7, rw: minor clarifications, add GPL statement
// 23. Jan. 2003 - V 6, rw: simplifications byte to int conversion, test with new
//                          card reader, review of the source code, 3rd published
// 31. Oct. 2002 - V 5, rw: include decoding of the selected file parameters,
//                          2nd published version
// 29. Oct. 2002 - V 4, rw: description of the possible jump out of the selected
//  9. Oct. 2002 - V 3, rw: improved documentation, 1st published version
//  8. Oct. 2002 - V 2, rw: improved UI, bug fix concerning selection by FID > 25
//  2. Oct. 2002 - V 1, rw: initial runnable version
//------------------------------------------------------------------------------

import java.awt.*;
import java.awt.event.*;
import java.util.*;
import opencard.core.service.*;
import opencard.core.terminal.*;
import opencard.opt.util.*;

public class ScanFiles implements ActionListener {
  static final int IFD_TIMEOUT = 0;      // unit: seconds
  static final int MAX_APDU_SIZE = 250;  // unit: byte
  static final int MIN_FID = 0x6e00;     // start value for FID scan
  static final int MAX_FID = 0x72FF;     // stop value for FID scan
  static final int CAPDU_LE = 4;         // pointer to Le within a command APDU
  static final int RAPDU_SW2 = 1;        // pointer to P1 within a command APDU

  // pointer to type of a file within the response of a SELECT FILE
  static final int R_TYPE_OF_FILE = 7 - 1;
  // pointer to structure of a file within the response of a SELECT FILE
  static final int R_FILE_STRUCTURE = 14 - 1;
  // pointer to the number of records (1 byte) within the response of a SELECT FI
  static final int R_SELECTFILE_NO_OF_RECORDS = 15-1;
  // pointer to the file size (2 byte) within the response of a SELECT FILE
```

```
static final int R_SELECTFILE_FILE_SIZE = 3-1;

// command SELECT FILE (select MF), according to GSM 11.11
// CLA || INS || P1 || P2 || Lc || DATA 1 (= FID high) || DATA 2 (= FID low)
static final byte[] CMD_SELECT_BY_FID = {(byte)0xA0, (byte)0xA4,
                                         (byte)0x00, (byte)0x00, (byte)0x02,
                                         (byte)0x00, (byte)0x00 };

// command GET RESPONSE, according to GSM 11.11
// CLA || INS || P1 || P2 || Le
static final byte[] CMD_GET_RESPONSE = {(byte)0xA0, (byte)0xC0,
                                        (byte)0x00, (byte)0x00, (byte)0x00 };

// FID of DF Telecom
static final byte[] FID_DF_TELECOM = {(byte)0x7F, (byte)0x10};

// Returncode '9Fxx': command propper executed, xx byte data available
static final byte RC_CMD_PROPPER_EXEC = (byte) 0x9F;

static private int n, i_fid;
static private byte[] i;
private PassThruCardService ptcs;
private Properties sysProps;
private Frame f;
private Panel p;
private static TextArea t;
private Button b;
static private ResponseAPDU response;
static private CommandAPDU command;

public ScanFiles() {
  // build the UI
  f = new Frame();
  f.setTitle("Smart Card - Terminal Communication");
  p = new Panel();
  p.setLayout(new FlowLayout(FlowLayout.CENTER, 0, 0));
  t = new TextArea(30, 120);
  t.setFont(new Font("Monospaced", 0, 12));   // Courier, 12 point size
  t.setBackground(new Color(0, 0, 0));        // black
  t.setForeground(new Color(200, 255, 100));  // yellowgreen
  p.add(t);
  b = new Button("Exit");
  b.addActionListener(this);
  p.add(b);
  f.add(p);
  f.pack();
  f.show();
} // ScanFiles

public void actionPerformed(ActionEvent evt) {
// exit programm if exit button is pressed
  if (evt.getSource() == b) System.exit(0);
} // actionPerformed

public static String IntTo5CharHexString(int x) {
  // converts a integer to the format "xx xx", x = hex digit
  String s, FID;
  s = Integer.toHexString(x/256).toUpperCase();   // convert high byte
  if (s.length() == 1) s = "0" + s;
  FID = s + " ";
  s = Integer.toHexString(x%256).toUpperCase();   // convert low byte
  if (s.length() == 1) s = "0" + s;
  FID = FID + s;
  return FID;
```

```
  } // IntTo5CharHexString

  public static void showText(String Text) {
    t.append(Text);
  } // showText

  public void showFID(String Text) {
    // show the "running" FID number in the UI
    t.replaceRange(Text, t.getCaretPosition()-6, t.getCaretPosition());
  } // showFID

  public void showATR(byte[] ATR) {
    int n, x;
    String s = new String();

    t.append("ATR:      ");
    for (n=0; n<i.length; n++) {
      x = (int) (0x000000FF & ATR[n]); // byte to int conversion
      s = Integer.toHexString(x).toUpperCase();
      if (s.length()== 1) s = "0" + s;
      t.append(s + " ");
    } // for
    t.append("\n");
  } // showATR

  public void showCmdAPDU(byte[] CmdAPDU, int LenCmdAPDU) {
    int n, x;
    String s = new String();

    t.append("IFD->ICC: ");
    for (n=0; n<LenCmdAPDU; n++) {
      x = (int) (0x000000FF & CmdAPDU[n]); // byte to int conversion
      s = Integer.toHexString(x).toUpperCase();
      if (s.length() == 1) s = "0" + s;
      t.append(s + " ");
    } // for
    t.append("\n");
   } // showCmdAPDU

  public void showRspAPDU(byte[] RspAPDU, int LenRspAPDU) {
    int n, x;
    String s = new String();

    t.append("ICC->IFD: ");
    for (n=0; n<LenRspAPDU; n++) {
      x = (int) (0x000000FF & RspAPDU[n]); // byte to int conversion
      s = Integer.toHexString(x).toUpperCase();
      if (s.length()== 1) s = "0" + s;
      t.append(s + " ");
    } // for
    t.append("\n");
  } // showRspAPDU

  public String showFileInfo(byte[] RspAPDU, int LenRspAPDU) {
    int x;
    int FileSize, RecordLength;
    String s = new String();

    s = "filetype: ";
    if (RspAPDU[R_TYPE_OF_FILE] == 00) s = s + "RFU";
    else if (RspAPDU[R_TYPE_OF_FILE] == 01) s = s + "MF";
    else if (RspAPDU[R_TYPE_OF_FILE] == 02) s = s + "DF";
    else if (RspAPDU[R_TYPE_OF_FILE] == 04) s = s + "EF";
    else s = s + "unknown (= " + RspAPDU[R_TYPE_OF_FILE] + ")";
```

```
    if (RspAPDU[R_TYPE_OF_FILE] == 04) {  // it is a EF
      x = (int) (0x000000FF & RspAPDU[R_SELECTFILE_FILE_SIZE]); // byte to int co
      FileSize = x*256;
      x = (int) (0x000000FF & RspAPDU[R_SELECTFILE_FILE_SIZE+1]); // byte to int
      FileSize = FileSize + x;
      s = s + "; file size: " + FileSize + " bytes";

      s = s + "; file structure: ";
      if (RspAPDU[R_FILE_STRUCTURE] == 00) s = s + "transparent";
      else if (RspAPDU[R_FILE_STRUCTURE] == 01) s = s + "linear fixed";
      else if (RspAPDU[R_FILE_STRUCTURE] == 03) s = s + "cyclic";
      else s = s + "unknown (= " + RspAPDU[R_FILE_STRUCTURE] + ")";

      if (RspAPDU[R_FILE_STRUCTURE] == 01) {  // the file have a linear fixed str
        RecordLength = (int) (0x000000FF & RspAPDU[R_SELECTFILE_NO_OF_RECORDS]);
        s = s + ("; record length: " + RecordLength + " bytes");
      } // if

      if (RspAPDU[R_FILE_STRUCTURE] == 03) {  // the file have a cyclic structure
        RecordLength = (int) (0x000000FF & RspAPDU[R_SELECTFILE_NO_OF_RECORDS]);
        s = s + ("; record length: " + RecordLength + " bytes");
      } // if
    } // if

     return s;
  } // showFileInfo

  public static void main( String [] args ){
    ScanFiles sf = new ScanFiles();
    sf.showText("Start Program: ScanFiles\n\n");
    // get and set system properties for OCF, PC/SC and PassThruCardServce
    Properties sysProps = System.getProperties();
    sysProps.put ("OpenCard.terminals", "com.ibm.opencard.terminal.pcsc10.Pcsc10C
    sysProps.put ("OpenCard.services", "opencard.opt.util.PassThruCardServiceFact

    try {
      SmartCard.start();
      CardRequest cr = new CardRequest(CardRequest.ANYCARD, null, PassThruCardSer
      cr.setTimeout(IFD_TIMEOUT);  // set timeout for IFD
      SmartCard sc = SmartCard.waitForCard(cr); // wait  for ICC in the IFD
      sf.showText("INFO:     activate smart card\n\n");

      if (sc != null) {  // no error occur, a smart card is in the terminal
        PassThruCardService ptcs = (PassThruCardService) sc.getCardService(PassTh
        command = new CommandAPDU(MAX_APDU_SIZE); // set APDU buffer size

        // get ATR
        CardID cardID = sc.getCardID();
        i = cardID.getATR();
        sf.showATR(i);

        // select a DF in the SIM (DF Telecom)
        sf.showText("\nINFO:     SELECT DF Telecom\n");
        command.append(CMD_SELECT_BY_FID);
        command.setByte(5, FID_DF_TELECOM[0]);   // set FID high
        command.setByte(6, FID_DF_TELECOM[1]);   // set FID low
        sf.showCmdAPDU(command.getBytes(), command.getLength());
        response = ptcs.sendCommandAPDU(command);
        sf.showRspAPDU(response.getBytes(),response.getLength());
        // send GET RESPONSE and fetch the data from the previous SELECT FILE com
        sf.showText("\nINFO:     GET RESPONSE\n");
        command.setLength(0);
        command.append(CMD_GET_RESPONSE);
```

```
        command.setByte(4, response.getByte(1));   // set number of byte to fetch
        sf.showCmdAPDU(command.getBytes(), command.getLength());
        response = ptcs.sendCommandAPDU(command);
        sf.showRspAPDU(response.getBytes(),response.getLength());
        sf.showText("\n");

        // this is the main loop for scanning all the files within the preselecte
        sf.showText("try to select file with FID =  xx xx");
        for (i_fid =MIN_FID; i_fid < MAX_FID; i_fid++) {
          sf.showFID(IntTo5CharHexString(i_fid));
          command.setLength(0);
          command.append(CMD_SELECT_BY_FID);
          command.setByte(5, i_fid/256);   // set FID high
          command.setByte(6, i_fid%256);   // set FID low
          response = ptcs.sendCommandAPDU(command);
          if ((byte) response.getByte(0) == (byte) RC_CMD_PROPPER_EXEC) {
            // EF found
            sf.showText(" -> found file: ");
            // send GET RESPONSE and fetch the data from the previous SELECT FILE
            command.setLength(0);
            command.append(CMD_GET_RESPONSE);
            command.setByte(CAPDU_LE, response.getByte(RAPDU_SW2));   // set numb
            response = ptcs.sendCommandAPDU(command);
            sf.showText(sf.showFileInfo(response.getBytes(),response.getLength())
            if (i_fid < MAX_FID-1) sf.showText("try to select file with FID = xx
          } // if
          sf.showText("\n");
        } // for

        sf.showText("\nINFO:     deactivate smart card\n");
        SmartCard.shutdown();
      } // if
      else
        sf.showText("\n\ncould not create smart card object (e.g. no ICC in IFD)\
    } // try
    catch (Exception except) {
        sf.showText("\n\nCaught exception '" + except.getClass() + "' - " + exce
    } // catch
  } // main
} // class
```

ScanFiles.java