TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

D I P L O M A R B E I T

ISIM Investigation and Specification - a sample implementation on the IBM JavaCard (JC/OP)

Ausgeführt am Institut für
Softwaretechnik und Interaktive Systeme
der Technischen Universität Wien

Unter der Anleitung von
Univ.Doz. Dipl.-Ing. Dr.techn. Ernst Piller

durch

Schröttner Robert
Leonard Bernstein Str. 4-6/2/142
1220 Wien

Wien, 28.07.2002

ISIM Investigation and Specification - a sample implementation on the IBM JavaCard (JC/OP)

Schröttner Robert

July, 28th 2002

Contents

List of Figures

List of Tables

Abstract

This Document gives an introduction to the ISIM specification and provides a sample implementation on the IBM JavaCard (JC/OP).

ISIM is a specification by 3GPP which describes how to store public and private identities for authentication onto a Universal Integrated Circuit Card (UICC) for operating in IP-Multimedia networks (IMS). This specification also defines an application protocol for some security functions. The ISIM Application can be used on a UICC standalone (e.g. for authentication in a wireless local area network) or as addition to SIM's for GSM or UMTS networks.

Uses in GSM or UMTS Environments and for Wireless-LAN Authentication are not covered by this Document. Therefore only usage of ISIM in card reader and a simple exploring tool for this application is inspected in this paper.

Usage in IP-Multimedia Networks requires support for the Session Initiaton Protocol (SIP) which lacks available tools presently.

Chapter 1

Definitions, symbols and abbreviations

1.1 Definitions

UICC Integrated Curcuit installed on a Plasic Card. The Integrated Curcuit includes Processor (CPU) and volatile and non-volatile Memory (ROM, EEPROM, RAM) and can be used for general purpose.

ISIM In this document the ISIM is a term that indicates the collection of IMS security data and functions for a UICC. The ISIM may be a distinct application on the UICC.

1.2 Abbreviations

3GPP 3rg Generation Partnership Project[1]

AAA Authentication Authorisation and Accounting

ADF Application Dedicated File

DF Dedicated File

EF Elementary File

GSM Global System for Mobile communications

HTTP Hyper Text Transfer Protocol, see RFC2068[16]

IM IP Multimedia

IMPI IM Private Identity

[1] see 3gpp Home at `http://www.3gpp.org`

IMPU IP Public Identity

IMS IP Multimedia Core Network System, see TS22.228[2]

ISIM IM Subscriber Identity Module

PKI Public Key Infrastructure[2]

SIM Subscriber Identity Module

SIP Session Initiated Protocol

UE User Equipment

UICC Universal Integrated Circuit Card

UMTS Universal Mobile Telecommunications System

USIM Universal Subscriber Identity Module

[2]see PKI-Home at `http://www.pki-page.org`

Chapter 2

Architecture Overview

This document relates to 3GPP system evolution, release 5 and upcoming release 6. Authentication takes place with the help of the ISIM Application, which is on the UICC in the UE's.

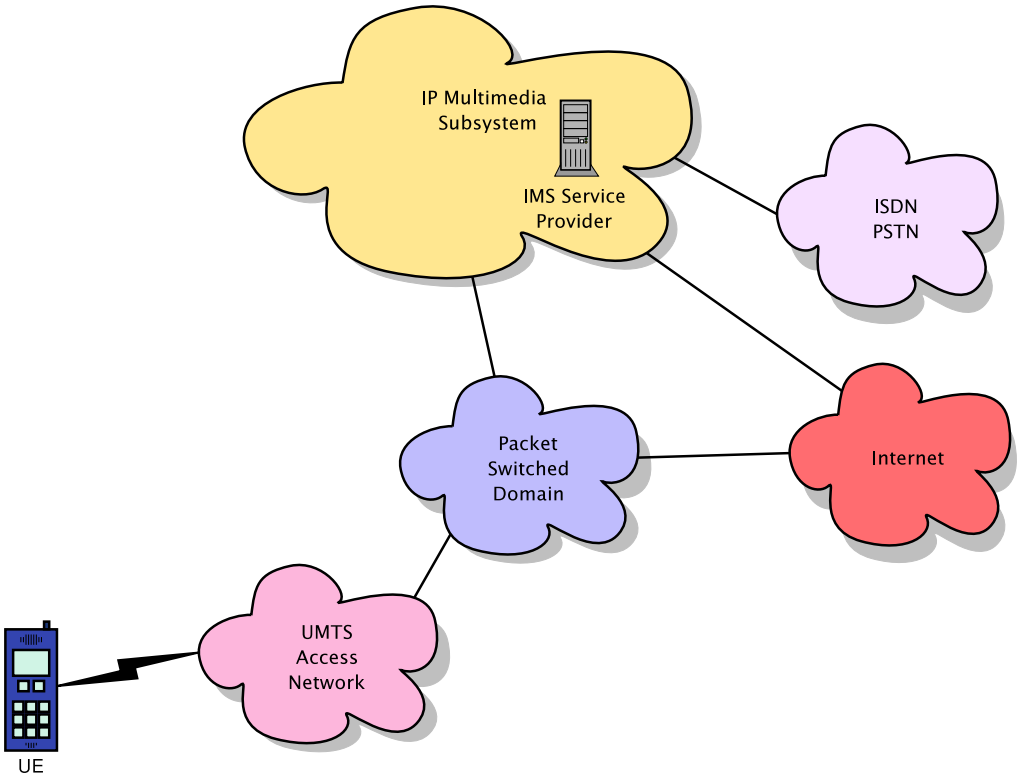

Figure 2.1: 3GPP System Evolution, REL-5

Figure 2.1 shows Release-5. This Release is a convergence to the all-IP world. In this stage ISM provides multimedia functions and the access network is UTRAN[4].

In Figure 2.2 you see the upcoming Release-6. There, IMS[2]is the service plat-

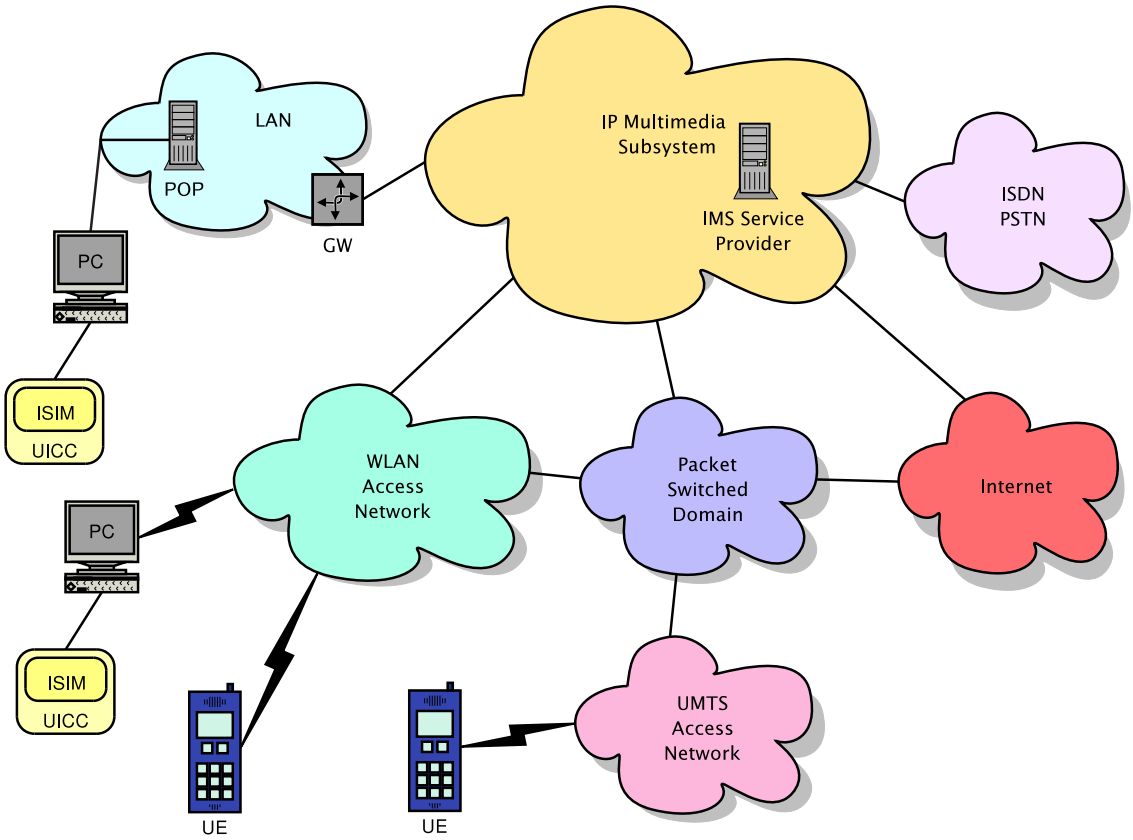

Figure 2.2: 3GPP System Evolution, REL-6

form in an all-IP world and the IMS can be accessed from multiple different access points.

In this environments the ISIM Application is responsible for authentication in the IMS. IMS connection establishments are based on SIP[17]. This protocol is responsible for registration of user and services, and manages searching (for user and services), dynamic routing, connection establishment and connection termination. Due to the functionality of SIP users can change their access points (For example: a user switches from UMTS access to WLAN) and SIP manages the changed routing without terminating users connections.

2.1 Overview of the IMS procedures

In the real world all available services are registered in the IMS onto a SIP-Registrar[17]. So the service ist known and can be easily found by all users. Without this registration users would need to know the exact location of the service (host-name or ip-address and port number of the service).

The UE or the PC (and so the ISIM) may then register itself to the IMS for

callback of some services. For example, the Server may connect back to the UE for security reasons.

2.2 Overview of the ISIM implementation

The sample implementation runs on the IBM JavaCard (JC/OP)[22] as single applet. This card is a GlobalPlatform[20] card which implements the Card Specification 2.0.1[21]. This specification conflicts in some score with ETSI TS 102.221[10]. So, in this sample implementation some functions deviate from the specification.

The Card provides some functions for uploading, selecting and deleting applets. This functions are given through a subset of ISO7816[12]. So some functions cannot be coded like ETSI TS 102.221[10] since those likewise is a subset of ISO7816[12] too. The implemetation uses therefore for some commands other class-bytes. (see code documentation for more details)

Chapter 3

PKI Systems

The keys stored on the ISIM and the authentication services are based on public key cryptography. So there is following a short introduction in the mode of operation of such systems.

3.1 Cryptography Basics

Consider a siutaltion where Alice, a user from company A, is electronically communicating with Bob, a user from company B. If Alice uses an insecure communications method such as standard e-mail, then typically the message can be read by anyone and is termed to be in cleartext or plaintext. If Alice wishes to secure the communication to ensure privacy then she can encrypt the message, i.e. encoding the message to render it unreadable to those not on the intended recipient list. Intended recipients (Bob in this example) receive the encrypted message, known as cyphertext, and decrypt it, i.e. convert it from cyphertext back to cleartext. Encryption and decryption typically occur using complex mathematical algorithms with the use of a key. There are two classes of key-based encryption algorithms, *symmetric* (private key) and *asymmetric* (public key) - these algorithms are explained in the following sections.

Since the key forms the basis of the encryption, its strength against attack is an important feature. An indication of a key's strength can be obtained from its length - for a given encryption algorithm the longer the key, the stronger the key. The most basic level of attack is known as a brute force attack, in which every possible key combination is tried in sequence. As the key length grows, brute force attacks exponentially grow in difficulty. Current symmetric encryption technologies typically use 128-bit length keys - this means that there are 2128 different key combinations, a number that is generally assumed to be unbreakable by brute force with current levels of computing power. Current asymmetric encryption technologies typically use 1024-bit length keys. The difference in key lengths occurs because not every combination is suitable for use in asymmetric encryption due to the specific mathematical methods on which the encryption technologies are based. Therefore,

the larger key length is needed to provide a similar level of strength as 128-bit private keys. Finally, note that the key strength becomes weaker as computing power increases, and the key length has to be extended to provide the same level of security as technology improves.

3.2 Symmetric (Private Key) Encryption

Symmetric encryption is the simpler of the two classes of key-based encryption algorithms. In this class, the same key is used to encrypt and decrypt the message. Taking our example, Alice would encrypt her message using a key, and then send the message to Bob. Alice would separately communicate the key to Bob to allow him to decrypt the message. To maintain security and privacy, Alice and Bob need to ensure that the key remains private to them. This scenario is illustrated in Figure 3.1

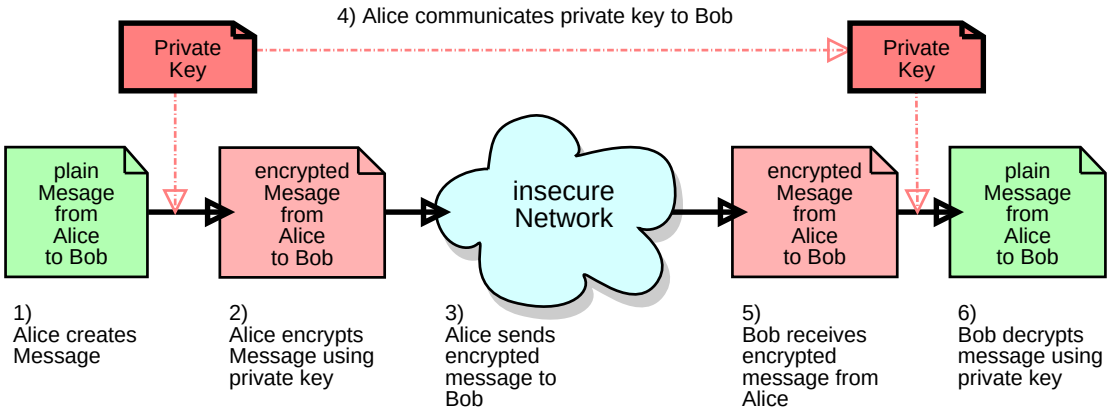

Figure 3.1: Symmetric encryption example

A simple example of this is the use of a password as the key. Alice could send a password protected message to Bob. Alice would then communicate the password to Bob separately, e.g. by a direct telephone call. The same password is used to encrypt and decrypt the protected message. If the password is compromised, then the message is no longer secure.

The simplicity of symmetric encryption is also the source of its problem. As Alice tries to secure her communications with more people, she needs to have a different key for each person. This problem also occurs for everyone else in the group who wishes to communicate securely. In a group of N people wishing to communicate securely, N*(N-1)/2 private keys need to exist. As the number of people N increases, the management of the private keys becomes a costly and cumbersome exercise.

3.3 Asymmetric (Public Key) Encryption

Asymmetric, or Public Key, encryption differs from symmetric encryption by using a pair of keys instead of a single key. One of the keys is kept private while the second one is made public so that it can be accessed by anyone. The key-pair works in a complementary manner - information encrypted by one key can only be decrypted using the other. This property of the key-pair can be used to perform two functions, encryption and digital signatures.

3.3.1 Encryption

In using key-pairs for encryption, the public key of the recipient is used to encrypt the message before it is sent to the recipient. The complementary nature of the key-pair means that only the recipient's private key can decrypt the message. As long as the private key remains secure, then the sender can be sure that only the intended recipient can possibly read the message.

Referring to the previous example, where Alice is sending a message to Bob. Alice creates her message then encrypts it using Bob's public key. When Bob receives the encrypted message he uses his secret, private key to decrypt it. As long as Bob's private key has not been compromised, then both Alice and Bob know that the message is secure. Figure 3.2 illustrates this Example.

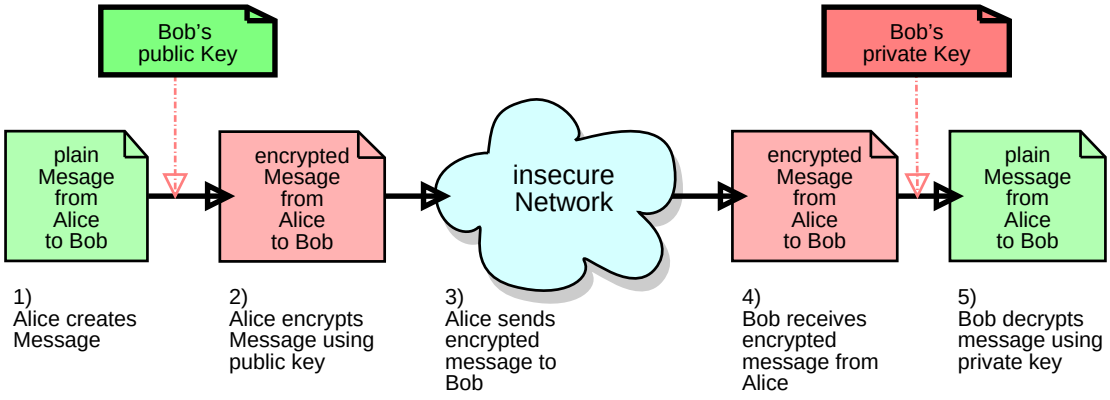

Figure 3.2: Asymmetric encryption example

In practice, the extra complexity of public key encryption over private key encryption results in a performance sacrifice, especially as the message size increases. To overcome this, encryption implementations may combine both encryption classes. The message is first encrypted using a one-use private key that has been randomly generated specifically for the message. The message-specific private key is then encrypted using the recipient's public key and both the encrypted message and encrypted private key are sent to the recipient. On receipt, the recipient uses his private key to decrypt the message-specific private key, thus giving him access to

the message. Since the message-specific private key is typically small in comparison with the message, the combined encryption approach provides the speed benefits of private key encryption along with the manageability of public key encryption.

3.3.2 Digital Signatures

The principle of public/private key-pairs can be used by the sender to digitally sign a message. A digital signature performs the same function as its physical counterpart - the sender 'marks' the message so that recipients can verify that the message really came from the sender. The digital signature also allows recipients to check that the message has not been tampered with since it was sent.

The process of digitally signing a message starts with the creation of a unique identifier for the message. The unique identifier can be created using a mathematical technique called Hashing. A hash function uses a mathematical algorithm to convert the message into a short fixed-length string of bits, often referred to as a 'hash value' or 'message digest', that uniquely represents the message used to create it. The hash value is specific to the contents of the message, thus any change to the message contents will change the hash value that would be generated by the hash function. Next, the hash value is encrypted using the sender's private key. Finally, the message is sent along with the encrypted hash value. On receiving the message and the encrypted hash value, the recipient can only decrypt the hash value using the sender's public key. This confirms that the message came from the sender and no-one else, as long as the sender's private key remains secure. The message can be re-hashed and compared with the decrypted hash value - if the values do not match then the message has been altered since it was sent. Figure 3.3 shows the process for Alice sending a digitally signed message to Bob.

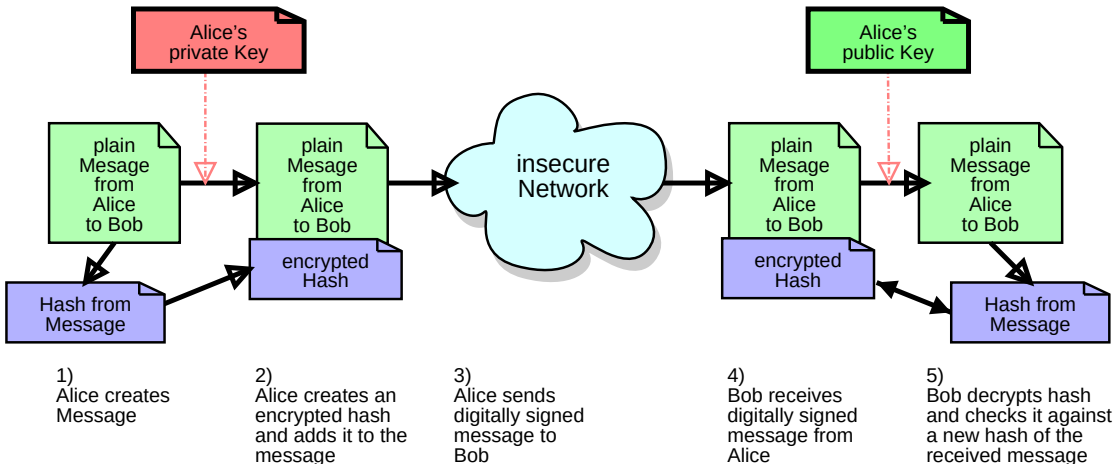

Figure 3.3: Digital signature example

3.3.3 Combining Encryption with Digital Signatures

The two functions of encryption and digital signatures can be combined during the sending of a message. The result is an encrypted message with an attached encrypted hash that only the intended recipient can read and that can be checked for both the sender's identity and for any evidence of tampering. Ostensibly the same key-pair can be used for both encryption and digital signatures. However, in practice users are generally advised to have a separate key-pair for each function. The reason for this stems from the differing natures of the functions and the resultant different key management requirements.

When a key-pair is used for encryption, the key-pair owner wants to ensure that the private key is never lost. If the owner loses the private key, any messages encrypted with the corresponding public key would no longer be retrievable. The key-pair owner therefore has an incentive to back-up the private key and maintain the backup for as long as possible. The backup may be kept long after the active life of the key-pair has passed, to allow the owner to decrypt old messages at a later date.

When a key-pair is used for digital signatures however, the key-pair owner should destroy the private key as soon as its active life has passed. Otherwise, if the private key is ever compromised in future then it can be used to forge the owner's digital signature on documents backdated to the key's active lifetime.

The conflict in requirements, archiving versus destruction, results in the advice regarding separate key-pairs.

3.4 PKI - The Implementation of Public Key Cryptography

Turning the theory of public key cryptography into a useful, real-world system requires more than just the implementation of the core algorithm. A number of supporting operational elements need to be in place before public key cryptography can be used effectively. The supporting infrastructure is collectively known as Public Key Infrastructure, or PKI for short.

A PKI consists of a set of policies, procedures and services to support applications of public key cryptography. The operational issues to running a PKI include how keys should be managed, how users have their identities checked, and how a specific user's public key is made available to other users. A PKI can therefore be split into the following components:

- A Security Policy
- A Certificate Authority (CA)
- A Registration Authority (RA)
- A Directory Service

3.4.1 Security Policy

The security policy contains definitions of the actual operation of the PKI. The operation of the other PKI components should be detailed here, as well as procedures for key generation, issuance, storage, and revocation. The security policy in effect acts as the framework on which the PKI is built.

3.4.2 Certificate Authority (CA)

So far we have discussed keys and their part in public key encryption. However, a key by itself does not contain supporting information such as who it belongs to, who issued the key, and the period over which it is valid. Without this information, then there is nothing linking a public key with its correct owner. The solution takes the form of digital certificates. A certificate contains information linking a specific public key to a specific individual. Taking our example of Alice and Bob - if Alice wanted to get Bob's public key to send him an encrypted message, she would in fact first obtain Bob's digital certificate, which confirms Bob's identity and contains the public key.

One obvious problem is how can Alice be sure that she has retrieved Bob's genuine certificate, and not a certificate for an imposter? The solution comes in the form of the Certificate Authority. The CA manages the process of certificate creation, issuance and revocation - it acts as a trusted third party for the certificates. The CA has its own certificate with which it digitally signs certificates that it issues. Trust in the CA's certificate may in turn be derived from the digital signature of a higher-level CA. Eventually, a CA is reached that the user trusts with no further required proof, e.g. certain trusted government agencies. These root CAs are few in number by definition, and form the basis of the trust hierarchy on which PKI relies.

The current industry standard for digital certificates is the CCITT X.509 international standard.

3.4.3 Registration Authority (RA)

When a user applies for a digital certificate from a CA, the CA has to verify that the applicant is truly who he claims to be. The role of the Registration Authority is to provide this verification. A real-world analogy would be a Notary Public, for example. Certain legal contracts require the signing process to be witnessed by a Notary Public, who acts to verify the signer's identity. In a similar way, the RA verifies the identity of the applicant and passes the application on to the CA. The degree of rigor applied by the RA during the verification will affect the degree of trust in the digital certificate. Some PKIs actually use Notary Publics to act as RAs - an applicant would physically sign a form, witnessed and notarized by a Notary Public, before sending the form to the CA. Other PKIs offer lower levels of trust, possibly only needing a specific e-mail address to which the certificate will be linked.

Finally, some PKIs may offer multiple levels of trusted certificates, depending on the applicant's requirements.

3.4.4 Directory Service

In our example with Alice sending an encrypted message to Bob, we have not yet discussed where and how Alice gets hold of Bob's certificate. The solution forms another component of a PKI - the directory service. In the same way that you might look in a standard phonebook to look up a telephone number, the directory service allows you to look up the digital certificate for someone to whom you wish to send an encrypted message.

Chapter 4

Specifications for 3GPP applications

The complete specification is defined in ETSI TS102.221[10]. Physical and electrical specifications are not part of this document. The initial communication procedures and the transmission protocols are managed by the card reader, the system software and the JavaCard itself and are therefore also unnoted in document.

4.1 Commands

The UICC knows a fistfull commands for selecting a file, reading and writing data and some for security. This commands are mostly identic to the standard ISO-commands (and are also used in today's commonly used GSM cards). There are only minor differences - so for a more precise description read the source code or the specification ETSI TS102.221[10].

4.2 Security

The security functions are mainly for locking files for read or update and offer an multi-level security system. The card's software may permit or deny access to files depending of a security state. States may be changed by PIN-commands or values stored in other files.

For example, after entering a PIN the card lets you read out the phone book, or lets you change another PIN. It's also possible to disable a security level by writing something to a file and thereby disabe a PIN inquiry.

Security functions are optional - even if it would be useless to implement a card for secure access without security functions. But if security functions are implemented, the exact security structure must be declared in the EF_{ARR} 4.7 file.

4.3 Files

The files of a UICC are arranged hierachically. There are four types of files defined. The Master File (MF) defines the root of the filesystem, is mandatory and must exist only once. This file is initially selected (it is the initial working directory). At the Master File level there may reside Elementary Files (EF's - like regular files) and Dedicated Files (DS's - like directories), which are root for other EF's or DF's. The (mandatory) EF_{DIR} at MF level is for grouping application specific files. EF_{DIR} stores references to Application Dedicated Files (ADF's) which are root for any files related to this application.

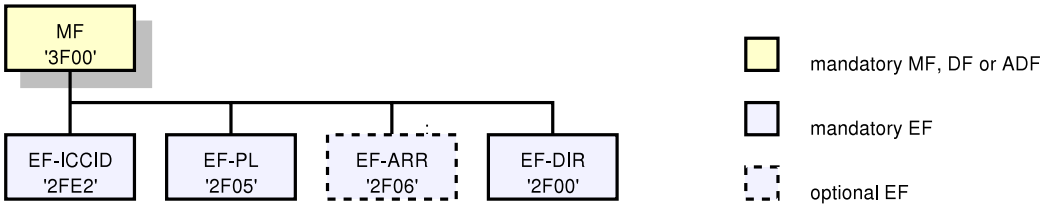

Figure 4.1: Minimal file structure for 3GPP

Files can be mandatory or optional. The file size of an optional file may be zero. All implemented files with a file size greater than zero shall contain all mandatory data items. Optional data items may either be filled with 'F', or, if located at the end of an file, need not exist.

4.3.1 The Master File (MF)

Identifier: '3F00'	Structure: Dedicated File	Mandatory
	Update activity: low	
Access Conditions		
READ	ALW	
UPDATE	NEV	
DEACTIVATE	NEV	
ACTIVATTE	NEV	

Table 4.1: MF

This is the root node of the filesystem in the UICC. It is implicitly selected at power on, and can be selected euther by a regular select to '3F00' or by a select command without given id and 'return-no-data' set. (please see secion Commands for more details)

4.3.2 Elementar Files at the Master-File level

There are four Elementary Files at the Master-File level specified in TS 31.102[6]. These files are required for the ISIM.

4.3.2.1 EF_{DIR}

Identifier: '2F00'	Structure: Linear fixed		Mandatory
SFI: Mandatory			
Record size: X bytes	Update activity: low		
Access Conditions READ ALW UPDATE ADM DEACTIVATE ADM ACTIVATTE ADM			
Bytes	Description	M/O	Length
1 - X	Application template TLV object	M	X Bytes

Table 4.2: EF_{DIR} (at MF-Level)

This EF is specified in ETSI TS102.221[10] and contains the Application Identifier (AID) and the Application Label as mandatory elements. The USIM application can only be selected by means of the AID selection. The EF_{DIR} entry shall not contain a path object for application selection. It is recommended that the application label does not contain more than 32 bytes.

EF_{DIR} is a linear fixed file. This file is under the responsibility of the issuer when at the MF level. It can be under a different responsibility, when situated at lower level. See table 4.2 for structure of this EF.

In table 4.4 the coding of the mandatory DOs and the optional DOs that has special meaning to this document. All other DOs are according to ISO/IEC 7816-5[13].

4.3.2.2 EF_{ICCID} (ICC Identity)

This EF provides a unique identification number for the UICC and is shown in table 4.5.

Identification Number
Contents: According to ITU-T Recomendation E.118[15].
Purpose: Card identification number
Coding: BCD, left justified and padded with 'F'. The order of digits is coded in figure 4.2

Length	Description	Status
1	Application template tag = '61'	M
1	Length of the application template = '03' - '7F'	M
1	Application Identifier tag = '4F'	M
1	AID length = '01 - '10'	M
'01' - '10'	AID value. For 3G applications, see 3GPP TS 31.110[8]	M
1	Application label tag = '50'	O
1	Application label length	O
see note 1	Application label value	O

NOTE 1 The application label is a DO that contains a string of bytes provided by the application provider to be shown to the user for information, e.g. operator name. The value part of the application label shall be coded according annex A. In ISO/IEC 7816-5 [13] the application label is limited to 16 bytes, this does not apply to the present document it is howewer recommended that the number of bytes does not exceed 32.

NOTE 2 Other DOs from ISO/IEC 7816-5 [13] may, at the application issuer's discretion, be present as well.

Table 4.4: Coding of an application template entry in EF_{DIR} (at MF-Level)

4.3.2.3 EF_{PL} (Preferred Languages)

This EF contains the codes for up to n languages. This information determined by the user/operator, defines the preferred languages of the user, for the UICC, in order of priority. EF_{PL} is shown in table 4.6.

Language Code

Coding: Each language code is a pair of alpha-numeric characters, defined in ISO 639[11]. Each alpha-numeric character shall be coded on one byte using the SMS default 7-bit coded alphabet as defined in TS 23.038[3] with bit 8 set to 0.
Unused languages shall be set to 'FF FF'.

4.3.2.4 EF_{ARR} (Access Rule Reference)

This EF contains the access rules for files located under the MS in the UICC. If the security tag '8B' is indicated in the FCP it contains a reference to a record in this

Identifier: '2FE2'	Structure: transparent		Mandatory
File size: 10 bytes		Update activity: low	
Access Conditions READ ALW UPDATE NEV DEACTIVATE ADM ACTIVATTE ADM			
Bytes	Description	M/O	Length
1 - 10	Identification number	M	X Bytes

Table 4.5: EF_{ICCID} (at MF-Level)

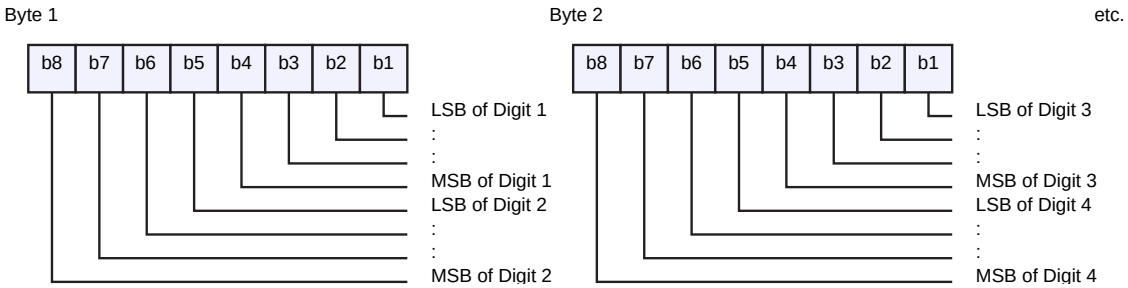

Figure 4.2: EF_{ICCID}, Field "Identification number" (at MF-Level)

file. EF_{ARR} is shown in table 4.7.

This EF contains one more records containing access rule information according to the referenced format as defined in ISO/IEC 7819-9[14]. Each represents an access rule. Unused bytes in the record are set to 'FF'.

Identifier: '2F05'	Structure: transparent	Mandatory
SFI: Mandatory		
File size: 2n bytes	Update activity: low	
Access Conditions READ ALW UPDATE PIN DEACTIVATE ADM ACTIVATTE ADM		

Bytes	Description	M/O	Length
1 - 2	1st language code (highest priority)	M	2 Bytes
3 - 4	2nd language code	O	2 Bytes
...			
2n-1 - 2n	nth language code (lowest priority)	O	2 Bytes

Table 4.6: EF_{PL} (at MF-Level)

Identifier: '2F06'	Structure: Linear fixed	Optional
Record length: X bytes	Update activity: low	
Access Conditions READ ALW UPDATE ADM DEACTIVATE ADM ACTIVATTE ADM		

Bytes	Description	M/O	Length
1 - x	Access Rule TLV data objects	M	X Bytes

Table 4.7: EF_{ARR} (at MF-Level)

Chapter 5

The ISIM specification

The specification defines a set of files, an Application Protocol and a set of commands. The following section speficies the files for the IMS session defining access conditions, data items and coding. A data item is a part of an File which represents a complete logical entity.

5.1 Commands

The ISIM specification defines only one command: AUTHENTICATE. This command is used during the procedure for authenticating the ISIM to its home network and vice versa. In addition, a cipher key and an integrity key are calculated. For the execution of the command, the ISIM uses the subscriber authentication key K, which is stored in the ISIM.

5.2 Files

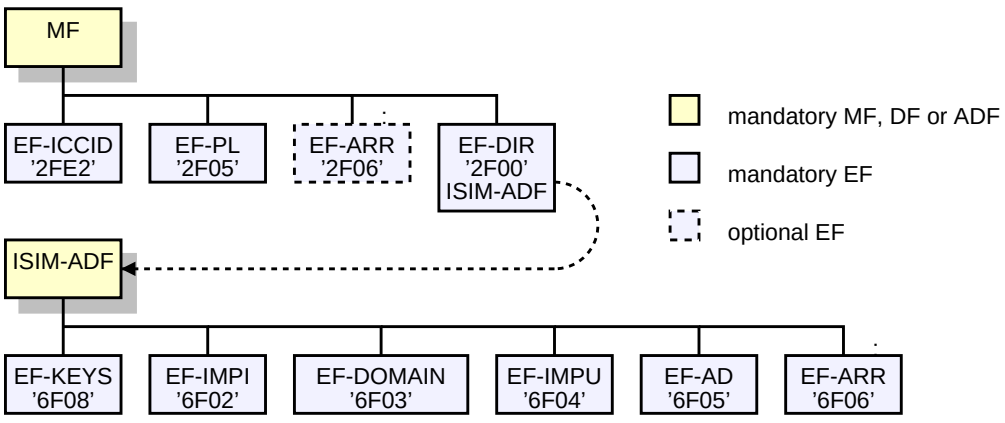

Figure 5.1: File structure for ISIM

5.2.1 Elementary Files at the ISIM ADF (Application DF) level

The Elementary Files in the ISIM ADF contain service and network related information and are required for UE to operate in an IP Multimedia Subsystem.

5.2.1.1 EF_{Keys} (Ciphering and Integrity Keys for IMS)

Identifier: '6F08'	Structure: transparent		Mandatory
SFI: '08'			
File size: 33 bytes		Update activity: high	
Access Conditions READ PIN UPDATE PIN DEACTIVATE ADM ACTIVATTE ADM			
Bytes	Description	M/O	Length
1	Key set identifier KSI	M	1 Bytes
2 - 17	Chipering key CK	M	16 Bytes
18 - 33	Integrity key IK	M	16 Bytes

Table 5.1: EF_{Keys}

This EF contains the ciphering key CK, the integrity key IK and the key set identifier KSI for the IP Multimedia Subsystem. EF_{Keys} is shown in table 5.1, the coding of the KSI-Field shows figure 5.2.

The least significant bit of the CK is the least significant bit of the 17^{th} byte. The most significant bit of the CK is the most significant bit of the 2^{nd} byte.

The least significant bit of the IK is the least significant bit of the 33^{rd} byte. The most significant bit of the CK is the most significant bit of the 18^{th} byte.

5.2.1.2 EF_{IMPI} (IMS private identifier)

This EF contains the private SIP Identity (SIP URI) ot the user and is is shown in table 5.2.

The URI-Field holds the private SIP URI of the User. For contents and coding of the URI-TLV data object values see RFC 3261[17]. The tag value of the URI TLV data object shall be '80'.

5.2.1.3 EF_{DOMAIN} (SIP domain URI)

This EF contains the SIP entry point in the home operator's network, if different from the host part of the private SIP URI of the user from file EF_{IMPI}. EF_{DOMAIN} is shown in table 5.3.

Identifier: '6F02'	Structure: transparent	Mandatory
SFI: '02'		
File size: X bytes	Update activity: low	
Access Conditions READ PIN UPDATE ADM DEACTIVATE ADM ACTIVATTE ADM		

Bytes	Description	M/O	Length
1 - X	URI TLV data object	M	X Bytes

Table 5.2: EF_{IMPI}

Identifier: '6F03'	Structure: transparent	Mandatory
SFI: '05'		
File size: X bytes	Update activity: low	
Access Conditions READ PIN UPDATE ADM DEACTIVATE ADM ACTIVATTE ADM		

Bytes	Description	M/O	Length
1 - X	URI TLV data object	M	X Bytes

Table 5.3: EF_{DOMAIN}

The URI-Field holds the request-URI. For contents and coding of the URI-TLV data object values see RFC 3261[17]. The tag value of the URI TLV data object shall be '80'.

5.2.1.4 EF_{IMPU} (IMS public Identifier of user)

This EF contains one ore more public SIP Identities (SIP URI) of the user and is shown in table 5.4.

The URI-Field holds the request-URI. For contents and coding of the URI-TLV data object values see RFC 3261[17]. The tag value of the URI TLV data object shall be '80'.

5.2.1.5 EF_{AD} (Administrative Data)

This EF contains information concerning the mode of operation according to the type if ISIM, such as normal (to be used by IMS subscribers for IMS operations), type approval (to allow specific use of the Terminal during type approval procedures

Identifier: '6F04'	Structure: transparent		Mandatory
SFI: '04'			
File size: X bytes	Update activity: low		
Access Conditions READ PIN UPDATE ADM DEACTIVATE ADM ACTIVATTE ADM			
Bytes	Description	M/O	Length
1 - X	URI TLV data object	M	X Bytes

Table 5.4: EF_{IMPU}

Identifier: '6FAD'	Structure: transparent		Mandatory
SFI: '03'			
File size: 3+X bytes	Update activity: low		
Access Conditions READ ALW UPDATE ADM DEACTIVATE ADM ACTIVATTE ADM			
Bytes	Description	M/O	Length
1	UE operation mode	M	1 Bytes
2 - 3	Additional information	M	2 Bytes
4 - 3+X	RFU	O	X Bytes

Table 5.5: EF_{IMPU}

of e.g. the network equipment) or manufacturer specific (to allow the Terminal manufacturer to perform specific proprietary auto-test in its Terminal e.g. maintenance phases).

It also provides indication of thether some Terminal features should be activated during normal operation.

Field "UE operation mode" holds the mode of operation for UE and is coded as follows:

UE operation mode

'00' normal operation (this is the initial value)
'80' type approval operations
'01' normal operation with specific facilities enabled
'81' type approval operations with specific facilities enabled
'02' maintenance (off line)

Field "Additional information" is for coding specific enabled facilities and is shown in figure 5.3.

The OFM bit is used to control the Ciphering Indicator as specified in TS 22.101[1] Terminal manufacturer specific information (if b2=1 in byte 1).

5.2.1.6 EF_{ARR} (Access Rule Reference)

This EF (shown in table 5.6) contains the access rules for files located under the ISIM-ADF in the UICC. If the security attribute tag '8B' is indicated in the FCP it contains a reference to a recored in this file.

Identifier: '6F06'	Structure: transparent		Mandatory
SFI: '06'			
File size: X bytes	Update activity: low		
Access Conditions READ ALW UPDATE ADM DEACTIVATE ADM ACTIVATTE ADM			
Bytes	Description	M/O	Length
1 - X	Access Rule TLV data ovject	M	X Bytes

Table 5.6: EF_{ARR}

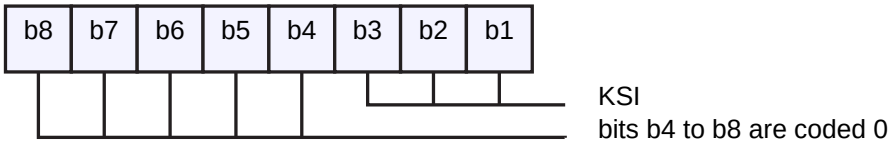

Figure 5.2: EF_{KEYS}, Field "Key Set Identifier Coding"

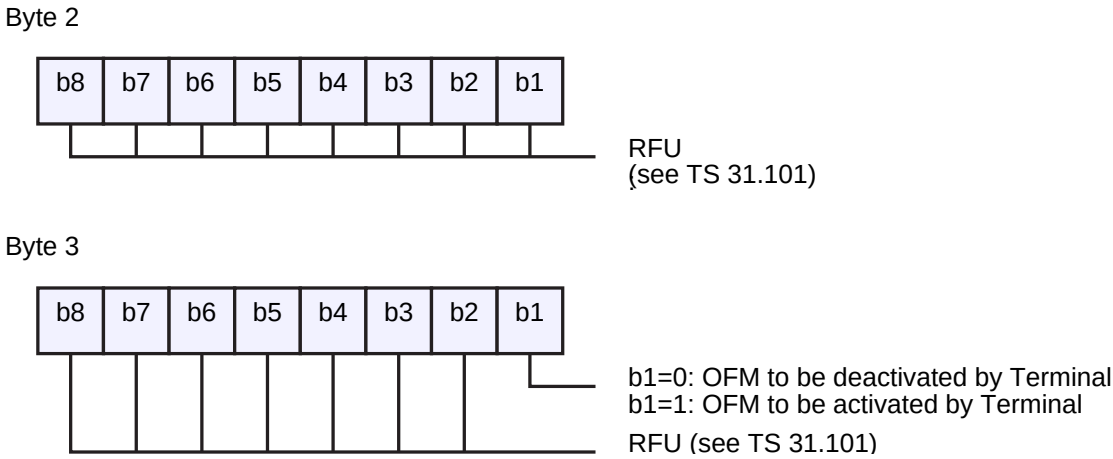

Figure 5.3: EF_{AD}, Field "Additional Information"

Chapter 6

Sample Implemetation

The sample implementation is done on a IBM JavaCard (JC/OP)[22]. This is an OpenCard compilant JavaCard. The card is able to store multiple applicatinos (the applets) which can be selected by commands provided by the OpenCard specification. At last one applet is always installed on the card: The applet-loader. Applets can communicate with other applets over special channels supervised by the JavaVM.

The complete implementation of the ISIM specification is done in one applet. So no inter-applet communication is needed.

6.1 JavaCard Applet

The applet implements most of the commands required by 3GPP specification for applications. Especially commands which are only required for USAT UICC's are not implemented. (This commands are only for USIM applications which requires to interact directly to the user).

6.1.1 Requirements and restrictions for OpenCard compilant cards

The OpenCard specification defines the 'SELECT BY NAME'-command as command for selecting of the installed applets (and this command is handled by the OpenCard internally). But the specification won't allow all options as 3GPP requires for this command. So the implementation allows the usage of an alternate class byte (80 additionally to 00) for SELECT-commands.

The implementation also provides basic (but untested) support for multiple channels. As the current IBM-Card supports only the JavaCard 2.1 specification, multiple channels are currently not supported by the card.

Support for secure communication is also missing for the same restriction.

6.1.2 The ISIM application

The applet must be uploaded to the card with name (=AID) 'A0 00 00 87'. This name is the first part of the officially registered registration application provider identifier (RID) by 3GPP.

The ISIM part of the applet itself registers with AID='A0 00 00 87 10 04'. An USIM application would have AID='A0 00 00 87 10 02'.

This procedure is required because the applet may need to keep control after selecting another 3GPP application. In JavaCard's point of view there is only one applet with name 'A0 00 00 87'. If you select 'A0 00 00 87 10 04' you select the same applet again, but the applet itself remembers to switch to the right 3GPP application.

For uploading the applet to the IBM JavaCard, the IBM JC/OP Tools[23] must be are used.

6.2 Test client

There is a small test client which communicates with the card via PC/SC. The clients requires at least JPCSC-0.6 installed. The test client may be used to send some commands to the card. It also decodes card's return data into a human readable format.

Chapter 7

Summary

The aim of this work was first, to overview the IP Multimedia Subscriber Identity Module (ISIM) and second to show one possible implementation at a JavaCard.

7.1 The ISIM application

The ISIM application is one of the applications that are stored on an UICC. The purpose of ISIM is to offer a secure communication between the user equipment and the connected network. With this aim ISIM supplies some encryption keys to the UE. With this keys the UE may establish a secure communication to the IMS (the network used by the UE).

For safety related purposes the UICC operates at multiple security levels. The levels may be changed by entering PIN's. Only if a minimum security level is activated, the encryption keys are disclosed to the UE.

This procedure permits a operation in many UE's. Maybe a cell phone or a wireless lan card connected to a computer - everyone can establish a secure connection into the IMS. The advantage of the secured communication is that the network provider can unambiguously identify the UICC because the encryption keys and some other identity keys are assigned by the network provider itself and can't be changed later. Furtheremore the keys of the UICC's can't be copied without having the right authorisation (PIN) and so not even stolen UICC's can establish a secure communication.

7.2 Implementation on a JavaCard

The sample implementation shows that it's impossible to completely implement a ISIM onto the IBM JavaCard. Because of the programming interface of the OpenCard compatible IBM JavaCard there are some conflicts to the ISIM specification.

The sample implementation therefore injures the specification by remapping some commands codes. Otherwise an implementation of a ISIM would be possible - and is possible, maybe on another (not OpenCard compatible) JavaCard.

For a meaningful use, there are some other applications that have to be implemented. At least the Universal Subscriber Identity Module (USIM) which manages mainly the usual functions for logging into the network must be implemented. In comparision to ISIM the USIM is a very complex application and the possibility to implement the USIM onto a JavaCard would be an excercise of a further elaboration.

Bibliography

[1] 3GPP TS 22.101: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Service aspects; Service principles (Release 5)".

[2] 3GPP TS 22.228: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Service requirements for the IP Multimedia Core Network Subsystem, Stage 1 (Release 5)".

[3] 3GPP TS 23.038: "3rd Generation Partnership Project; Technical Specification Group Terminals; Digital cellural telecommunications system (Phase 2+) CGSM; Universal Mobile Telecommunications System (UMTS); Alphabets and language-specific information (Release 4)".

[4] 3GPP TS 25.401: "Universal Mobile Telecommunications System (UMTS); UTRAN Overall Description; (Version 3.1.0 Release 1999)".

[5] 3GPP TS 31.101: "3rd Generation Partnership Project; Technical Specification Group Terminals; UICC-Terminal Interface; Physical and Logical Characteristics".

[6] 3GPP TS 31.102: "3rd Generation Partnership Project; Technical Specification Group Terminals; Characteristics of the USIM Application".

[7] 3GPP TS 31.103: "3rd Generation Partnership Project; Technical Specification Group Terminals; Characteristics of the ISIM Application (Release 5)".

[8] 3GPP TS 31.110: "3rd Generation Partnership Project; Technical Specification Group Terminals; Numbering system for telecommunication IC card applications (Release 4)".

[9] 3GPP TS 33.203: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Access security for IP-based services (Release 5)".

[10] ETSI TS 102 221: "Smart Cards; UICC-Terminal interface; Physical and logical characteristics (Release 5)".

[11] ISO639: "Code for the representation of names of languages".

[12] ISO/IEC 7816: "Identification cards".

[13] ISO/IEC 7816-5: "Identification cards - Integrated curcuit(s) cards with contacts - Part 5: Numbering system and registration procedure for application identifiers".

[14] ISO/IEC 7815-9: "Identification cards - Integrated curcuit(s) cards with contacts - Part 9: Additional interindustry commands and security attributes".

[15] ITU-T Recomendation E.118: "The International Telecommunications Charge Card".

[16] RFC 2068: "Hypertext Transfer Protocol – HTTP/1.1" R. Fielding, J. Gettys, J, Mogul, H. Frystyk, T. Berners-Lee. January 1997.

[17] RFC 3261: "SIP: Session Initiation Protocol" J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, June 2002.

[18] RSA Laboratories Crypto FAQ at `http://www.rsasecurity.com/rsalabs/faq/sections.html`.

[19] IEEE Working Groups - IEEE P1363 Standard Specifications For Public-Key Cryptography at `http://grouper.ieee.org/groups/1363/index.html`.

[20] Global Platform at `http://www.globalplatform.org`.

[21] Card Specification 2.0.1: "Global Platform Open Platform Card Specification Version 2.0.1 7. April 2000".

[22] IBM JavaCard system software at `http://www.zurich.ibm.com/jcop/products/cards.html`.

[23] IBM JavaCard tools at `http://www.zurich.ibm.com/jcop/products/tools.html`.

Appendix A

Source code listings

A.1 JavaCard applet source

A.1.1 Apps3GPP/ACL.java

```
1  /*
2   * ACL.java
3   *
4   * Created on 16. Februar 2003, 13:57
5   */
6  package Apps3GPP;
7
8  /**
9   *  This class is a _very_ basic and incomplete implementation for handling
10  *  access lists for files.
11  *
12  *  @see     <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
13  *  @author  Schroettner Robert<rs@ednet.at>
14  *  @version 1.0
15  */
16 public class ACL {
17
18     ////////////////////////////////////////////////////////////////////////
19     //
20     //  Member variables
21     //
22
23     /** Allow always (0) - no authentication required
24      */
25     protected static final byte lvALW  = 0;
26
27     /** Require PIN (1) - authenticate with PIN first
28      */
29     protected static final byte lvPIN  = 1;
30
31     /** Require PIN2 (2) - authenticate with PIN2 first
32      */
33     protected static final byte lvPIN2 = 2;
34
35     /** RFU3 (3) - level reserved for future use
36      */
37     protected static final byte lvRFU3 = 3;
38
39     /** RFU4 (4) - level reserved for future use
40      */
```

```
41      protected static final byte lvRFU4 = 4;
42
43      /** RFU5 (5) - level reserved for administrative authority
44       */
45      protected static final byte lvRFU5 = 5;
46
47      /** RFU6 (6) - level reserved for administrative authority
48       */
49      protected static final byte lvRFU6 = 6;
50
51      /** Never allowed (7) - access is always denied
52       */
53      protected static final byte lvNEV  = 7;
54
55      /** Administrative access (9)
56       */
57      protected static final byte lvADM = 9;
58
59      /** required access level for read access
60       */
61      protected byte read;
62
63      /** required access level for update access
64       */
65      protected byte update;
66
67      /** required access level for file deactivation
68       */
69      protected byte deactivate;
70
71      /** required access level for file activation
72       */
73      protected byte activate;
74
75      /** isAllowed(ACCESS_READ) checks for read access
76       */
77      protected static final byte ACCESS_READ       = 0;
78
79      /** isAllowed(ACCESS_UPDATE) checks for update
80       */
81      protected static final byte ACCESS_UPDATE     = 1;
82
83      /** isAllowed(ACCESS_DEACTIVATE) checks for file deactivation
84       */
85      protected static final byte ACCESS_DEACTIVATE = 2;
86
87      /** isAllowed(ACCESS_ACTIVATE) checks for file activation
88       */
89      protected static final byte ACCESS_ACTIVATE   = 3;
90
91      ////////////////////////////////////////////////////////////////////////
92      //
93      //  Methods for setting/getting status
94      //
95
96      /**
97       *  Check if requested access is allowed for given security level
98       *  @param access requested access (ACCESS_xxx)
99       *  @param level  current security level
100      *  @return true if access is allowed
101      */
102     protected boolean isAllowed(byte access, byte level) {
103         switch (access) {
104             case ACCESS_READ:       return level >= read;
105             case ACCESS_UPDATE:     return level >= update;
106             case ACCESS_DEACTIVATE: return level >= deactivate;
```

```
107             case ACCESS_ACTIVATE:   return level >= activate;
108         }
109         // invalid access mode -> deny access
110         return false;
111     }
112 
113     ////////////////////////////////////////////////////////////////////////
114     //
115     //  Constructors
116     //
117 
118     /** Creates a new instance of ACL
119      *  @param _read       required minimum level for read access (lvXXX)
120      *  @param _update     required minimum level for update access (lvXXX)
121      *  @param _deactivate required minimum level for file deactivation (lvXXX)
122      *  @param _activate   required minimum level for file activation (lvXXX)
123      */
124     protected ACL(byte _read, byte _update, byte _deactivate, byte _activate) {
125         read       = _read;
126         update     = _update;
127         deactivate = _deactivate;
128         activate   = _activate;
129     }
130 }
```

A.1.2 Apps3GPP/Status.java

```
1  /*
2   * Status.java
3   *
4   * Created on 22. Februar 2003, 00:26
5   */
6  package Apps3GPP;
7 
8  import javacard.framework.Util;
9  import javacard.framework.ISOException;
10 //import javacard.framework.Shareable;
11 
12 /**
13  *  Valid values for return codes (StatusWord).
14  *
15  *  @see     <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0, Section 10.2.1</a>
16  *  @author  Schroettner Robert<rs@ednet.at>
17  *  @version 1.0
18  */
19 public class Status { //implements Shareable {
20 
21     ////////////////////////////////////////////////////////////////////////
22     //
23     // Normal ending
24     //
25 
26     /** Normal ending of the command.
27      *  <br>
28      *  Value: 0x9000<br>
29      *  Category: Normal Processing
30      */
31     public static final short OK_NO_ERROR = (short)0x9000;
32 
33     /** Normal ending of the command, with extra information from the procative
34      *  UICC containing a command for the terminal.
35      *  Length len bytes of the response data.<br>
36      *  Value: 0x90xx<br>
37      *  Category: Normal Processing<br>
38      *
```

```
39          *  @param len  len of the response data
40          */
41         public short OK_NO_ERROR_INFO(byte len) {
42             return Util.makeShort((byte)0x90, len);
43         }
44
45         ////////////////////////////////////////////////////////////////////////
46         //
47         // Postponed processing
48         //
49
50         /** SIM Application Toolkit is busy - command cannot be executed at present,
51          *  further normal commands are allowed.
52          *  <br>
53          *  Value: 0x9300<br>
54          *  Category: Postponed processing
55          */
56         public static final short INFO_TOOKIT_BUSY = (short)0x9300;
57
58         ////////////////////////////////////////////////////////////////////////
59         //
60         // Warnings
61         //
62
63         /** No Information given, state of non valatile memory unchanged.
64          *  <br>
65          *  Value: 0x6200<br>
66          *  Category: Warnings
67          */
68         public static final short WARN_NO_INFO                 = (short)0x6200;
69
70         /** Part of the returned data may be corrupted.
71          *  <br>
72          *  Value: 0x6281<br>
73          *  Category: Warnings
74          */
75         public static final short WARN_RETURNED_DATA_CORRUPTED = (short)0x6281;
76
77         /** End of file/record reached fefore reading Le bytes.
78          *  <br>
79          *  Value: 0x6282<br>
80          *  Category: Warnings
81          */
82         public static final short WARN_EOF_REACHED             = (short)0x6282;
83
84         /** Selected file invalidated.
85          *  <br>
86          *  Value: 0x6283<br>
87          *  Category: Warnings
88          */
89         public static final short WARN_SELECTED_FILE_INV       = (short)0x6283;
90
91         /** Command successful but after using an internal update retry routine
92          *  cnt times.
93          *  <br>
94          *  Value: 0x63Cx<br>
95          *  Category: Warnings
96          *
97          *  @param cnt number of retries
98          */
99         public short WARN_SUCCESSFUL_AFTER(byte cnt) {
100            return Util.makeShort((byte)0x63, (byte)(0xC0 | (cnt&0x0F)));
101        }
102
103        /** Verification failed, cnt retries remaining.
104         *  <br>
```

```
105         *  Value: 0x63Cx<br>
106         *  Category: Warnings
107         *
108         *  @param cnt number of remaining retries
109         */
110        public short WARN_VERIFY_FAILED(byte remain) {
111            return Util.makeShort((byte)0x63, (byte)(0xC0 | (remain&0x0F)));
112        }
113
114        ////////////////////////////////////////////////////////////////////////
115        //
116        // Execution errors
117        //
118
119        /** No information given, state of non-valatile memory unchanged.
120         *  <br>
121         *  Value: 0x6400<br>
122         *  Category: Execution errors
123         */
124        public static final short ERR_NO_INFO              = (short)0x6400;
125
126        /** No information given, state of non-valatile memory changed.
127         *  <br>
128         *  Value: 0x6500<br>
129         *  Category: Execution errors
130         */
131        public static final short ERR_NO_INFO_MEM_CHANGED = (short)0x6500;
132
133        /** Memory problem.
134         *  <br>
135         *  Value: 0x6581<br>
136         *  Category: Execution errors
137         */
138        public static final short ERR_MEMOY_PROBLEM        = (short)0x6581;
139
140        ////////////////////////////////////////////////////////////////////////
141        //
142        // Checking errors
143        //
144
145        /** Wrong length.
146         *  <br>
147         *  Value: 0x6700<br>
148         *  Category: Checking errors
149         */
150        public static final short CHK_WRONG_LENGTH          = (short)0x6700;
151
152        /** The interpretation of this status word is command dependent.
153         *  <br>
154         *  Value: 0x67xx<br>
155         *  Category: Checking errors
156         *
157         *  @param reasong any return code except 0x00
158         */
159        public short CHK_FAILED_67(byte reason) {
160            return Util.makeShort((byte)0x67, reason);
161        }
162
163        /** Wrong parameter(s) P1-P2.
164         *  <br>
165         *  Value: 0x6B00<br>
166         *  Category: Checking errors
167         */
168        public static final short CHK_WRONG_PARAMETER_P1P2 = (short)0x6B00;
169
170        /** Instruction code not supported or invalid.
```

```
171         *  <br>
172         *  Value: 0x6D00<br>
173         *  Category: Checking errors
174         */
175        public static final short CHK_INS_NOT_SUPPORTED    = (short)0x6D00;
176
177        /** Class not supported.
178         *  <br>
179         *  Value: 0x6E00<br>
180         *  Category: Checking errors
181         */
182        public static final short CHK_CLA_NOT_SUPPORTED    = (short)0x6E00;
183
184        /** Technical problem, no precise diagnostics.
185         *  <br>
186         *  Value: 0x6F00<br>
187         *  Category: Checking errors
188         */
189        public static final short CHK_TECHNICAL_PROBLEM    = (short)0x6F00;
190
191        /** The interpretation of this status word is command dependent.
192         *  <br>
193         *  Value: 0x6Fxx<br>
194         *  Category: Checking errors
195         *
196         *  @param reasong any return code except 0x00
197         */
198        public short CHK_FAILED_6F(byte reason) {
199            return Util.makeShort((byte)0x6F, reason);
200        }
201
202        ////////////////////////////////////////////////////////////////////////
203        //
204        // Functions CLA not supported
205        //
206
207        /** No information given.
208         *  <br>
209         *  Value: 0x6800<br>
210         *  Category: Functions CLA not supported
211         */
212        public static final short CLA_NO_INFO              = (short)0x6800;
213
214        /** Logical channel not supported.
215         *  <br>
216         *  Value: 0x6881<br>
217         *  Category: Functions CLA not supported
218         */
219        public static final short CLA_NO_LOGICAL_CHANNEL   = (short)0x6881;
220
221        /** Secure messaging not supported.
222         *  <br>
223         *  Value: 0x6882<br>
224         *  Category: Functions CLA not supported
225         */
226        public static final short CLA_NO_SECURE_MESSAGING = (short)0x6882;
227
228        ////////////////////////////////////////////////////////////////////////
229        //
230        // Command not allowed
231        //
232
233        /** No information given.
234         *  <br>
235         *  Value: 0x6900<br>
236         *  Category: Command not allowed
```

```
237         */
238        public static final short CMD_NO_INFO                    = (short)0x6900;
239
240        /** Command incompatible with file structure.
241         *  <br>
242         *  Value: 0x6981<br>
243         *  Category: Command not allowed
244         */
245        public static final short CMD_INCOMPATIBLE               = (short)0x6981;
246
247        /** Security status not satisfied.
248         *  <br>
249         *  Value: 0x6982<br>
250         *  Category: Command not allowed
251         */
252        public static final short CMD_SECURITY_NOT_SATISFIED     = (short)0x6982;
253
254        /** Authentication/PIN method blocked.
255         *  <br>
256         *  Value: 0x6983<br>
257         *  Category: Command not allowed
258         */
259        public static final short CMD_PIN_BLOCKED                = (short)0x6983;
260
261        /** Referenced data invalidated.
262         *  <br>
263         *  Value: 0x6984<br>
264         *  Category: Command not allowed
265         */
266        public static final short CMD_DATA_INVALID               = (short)0x6984;
267
268        /** Conditions of used not satisfied.
269         *  <br>
270         *  Value: 0x6985<br>
271         *  Category: Command not allowed
272         */
273        public static final short CMD_CONDITIONS_NOT_SATISFIED = (short)0x6985;
274
275        /** Command not allowed (no EF selected).
276         *  <br>
277         *  Value: 0x6986<br>
278         *  Category: Command not allowed
279         */
280        public static final short CMD_NOT_ALLOWED                = (short)0x6986;
281
282        ////////////////////////////////////////////////////////////////////////
283        //
284        // Wrong parameters
285        //
286
287        /** Incorrect parameters in the data field.
288         *  <br>
289         *  Value: 0x6A80<br>
290         *  Category: Wrong parameters
291         */
292        public static final short PARA_INCORRECT_DATA          = (short)0x6A80;
293
294        /** Function not supported.
295         *  <br>
296         *  Value: 0x6A81<br>
297         *  Category: Wrong parameters
298         */
299        public static final short PARA_FUNCTION_NOT_SUPPORTED = (short)0x6A81;
300
301        /** File not found.
302         *  <br>
```

```
303         *  Value: 0x6A82<br>
304         *  Category: Wrong parameters
305         */
306        public static final short PARA_FILE_NOT_FOUND          = (short)0x6A82;
307
308        /** Record not found.
309         *  <br>
310         *  Value: 0x6A83<br>
311         *  Category: Wrong parameters
312         */
313        public static final short PARA_RECORD_NOT_FOUND        = (short)0x6A83;
314
315        /** Incorrect parameter(s) P1-P2.
316         *  <br>
317         *  Value: 0x6A86<br>
318         *  Category: Wrong parameters
319         */
320        public static final short PARA_INCORRECT_P1P2          = (short)0x6A86;
321
322        /** Lc inconsistent with P1-P2.
323         *  <br>
324         *  Value: 0x6A87<br>
325         *  Category: Wrong parameters
326         */
327        public static final short PARA_INCORRECT_LC            = (short)0x6A87;
328
329        /** Referenced data not found.
330         *  <br>
331         *  Value: 0x6A88<br>
332         *  Category: Wrong parameters
333         */
334        public static final short PARA_DATA_NOT_FOUND          = (short)0x6A88;
335
336        ////////////////////////////////////////////////////////////////////////
337        //
338        // Application errors
339        //
340
341        /** INCREASE cannot be performed, max value reached.
342         *  <br>
343         *  Value: 0x9850<br>
344         *  Category: Application errors
345         */
346        public static final short APP_NO_INCREASE           = (short)0x9850;
347
348        /** Authentication error, application specific.
349         *  <br>
350         *  Value: 0x9862<br>
351         *  Category: Application errors
352         */
353        public static final short APP_AUTHENTICATION_ERROR = (short)0x9862;
354
355        /** This is only a wrapper for ISOException.throwit().
356         * @param sw Statusword to return
357         * @see ISOException#throwIt
358         */
359        public static void throwIt(short sw) throws ISOException {
360            ISOException.throwIt(sw);
361        }
362    }
```

A.1.3 Apps3GPP/File.java

```
1   /*
2    * File.java
```

```
3   *
4   * Created on 16. Februar 2003, 14:00
5   */
6  package Apps3GPP;
7  import  Apps3GPP.Status;
8  import  Apps3GPP.DF;
9
10 import javacard.framework.APDU;
11 import javacard.framework.ISOException;
12
13 /**
14  *  This is the abstract base class for all file classes in an applet's
15  *  file system.
16  *
17  *  All files have:
18  *  <ul>
19  *   <li>A FID (16-bit file identifier) which may by disabled with NO_FID.</li>
20  *   <li>A parent dedicated file (which is null if the file has no parent).</li>
21  *   <li>Read/update/deactivate/activate access condition via acl object.</li>
22  *   <li>activated/deactivated-flag.</li>
23  *  </ul>
24  *
25  *  @see     <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
26  *  @author  Schroettner Robert<rs@ednet.at>
27  *  @version 1.0
28  */
29 public abstract class File {
30
31     ////////////////////////////////////////////////////////////////////////
32     //
33     //  Member variables
34     //
35
36     /**
37      *  Special FID value for condition "assign no fid to this file".
38      *  If NO_FID is used in Constructor, {@link #eqFid eqFid(...)}
39      *  returns always false.
40      */
41     public static final short NO_FID = (short)0x0000;
42
43     /**
44      *  FID: 16-bit file identifier.
45      */
46     private short fid;
47
48     /**
49      *  Access-Rights for this file.
50      */
51     private ACL acl;
52
53     /**
54      *  Parent DF (or null for none).
55      */
56     private DF parent = null;
57
58     /**
59      *  Flag for activated/deactivated.
60      */
61     private boolean activated = true;
62
63     ////////////////////////////////////////////////////////////////////////
64     //
65     //  Methods for setting/getting status
66     //
67
68     /**
```

```
69          *  Get 16-bit file identifier.
70          *  @return FID (or special value {@link #NO_FID})
71          */
72         protected short getFid() {
73             return fid;
74         }
75
76         /**
77          *  Check if this file has a FID assigned
78          *  @return true if there is a FID assigned
79          */
80         protected boolean hasFid() {
81             return fid != NO_FID;
82         }
83
84         /**
85          *  Check if this files identifier is equal the requested.
86          *  If the objects fid is {@link #NO_FID}, this function always
87          *  returns false.
88          *  @param  cmp  FID to compare with objects FID
89          *  @return true if the FID's are equal
90          */
91         protected boolean eqFid(short cmp) {
92             return hasFid() ? fid == cmp : false;
93         }
94
95         /**
96          *  Get parent DF (if any).
97          *  @return parent DF (or null)
98          */
99         protected DF getParent() {
100            return parent;
101        }
102
103        /**
104         *  Set parent DF.
105         *  Due lack of a garbage collector on JavaCard (memory can't be freed)
106         *  childs can be set only once.
107         *  @param  df  parent DF
108         */
109        protected void setParent(DF df) {
110            if (parent != null)
111                // parent already set -> ERROR!
112                Status.throwIt(Status.CHK_TECHNICAL_PROBLEM);
113
114            parent = df;
115        }
116
117        /**
118         *  Return if this file is activated or deactivated.
119         *  Deactivates files can only be selected or activated but
120         *  all other file functions (read, update, ...) will fail.
121         *  @return true if this file is activated
122         */
123        protected boolean isActivated() {
124            return activated;
125        }
126
127        /**
128         *  Activate or deactivate file.
129         *  @param active  parameter to activate or deactivate this file
130         */
131        protected void setActivated(boolean active) {
132            activated = active;
133        }
134
```

```
135     /**
136      *  Check if requestet access {@linkplain ACL#lvALW ACL.lvXXX} is
137      *  allowed for current security level.
138      *  @param  access  any of the  {@linkplain ACL#lvALW ACL.lvXXX} values
139      *  @return true for access allowed
140      */
141     protected boolean isAllowed(byte access) {
142         byte level = ACL.lvALW; // for now, level=0 (no pin entered)
143 
144         if (acl != null)
145             return acl.isAllowed(access, level);
146 
147         // default if no acl exists: allow access
148         return true;
149     }
150 
151     ////////////////////////////////////////////////////////////////////////
152     //
153     //  Constructors
154     //
155 
156     /**
157      *  Creates a new instance of File without FID.
158      *  @param _acl  access rule object
159      */
160     protected File(ACL _acl) {
161         fid = NO_FID;
162         acl = _acl;
163     }
164 
165     /**
166      *  Creates a new instance of File.
167      *  @param _fid  file identifier for this file
168      *  @param _acl  access rule object
169      */
170     protected File(short _fid, ACL _acl) {
171         fid = _fid;
172         acl = _acl;
173     }
174 }
```

A.1.4 Apps3GPP/EF.java

```
1  /*
2   * EF.java
3   *
4   * Created on 18. Februar 2003, 00:12
5   */
6  package Apps3GPP;
7  import  Apps3GPP.File;
8 
9  import javacard.framework.APDU;
10 
11 /**
12  *  Abstract base class for Elementary-Files.
13  *  EF has the same functionality as File, but can be selected by
14  *  a Short-FID. The Short-FID is usually aquired from the last 5 bits
15  *  of the FID but 3GPP sometimes does not meet this condition. So
16  *  the SFI is stored in a separately member variable.
17  *
18  *  @see     <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
19  *  @author  Schroettner Robert<rs@ednet.at>
20  *  @version 1.0
21  */
22 public abstract class EF extends File {
```

```
23
24      ////////////////////////////////////////////////////////////////////////
25      //
26      //  Member variables
27      //
28
29      /**
30       *  Special FID value for condition "assign no fid to this file".
31       *  If NO_FID is used in Constructor, {@link #eqFid eqFid(...)}
32       *  returns always false.
33       */
34      public static final byte NO_SFI = (short)0x00;
35
36      /**
37       *  Short-FID: 8-bit file identifier
38       */
39      private byte sfi;
40
41      ////////////////////////////////////////////////////////////////////////
42      //
43      //  Methods for setting/getting status
44      //
45
46      /**
47       *  Get the Short-FID
48       *  @return SFI
49       */
50      protected byte getSfi() {
51          return sfi;
52      }
53
54      /**
55       *  Check if this file has a FID assigned
56       *  @return true if there is a FID assigned
57       */
58      protected boolean hasSfi() {
59          return sfi != NO_SFI;
60      }
61
62      /**
63       *  Check if this files identifier is equal the requested.
64       *  If the objects fid is {@link #NO_FID}, this function always
65       *  returns false.
66       *  @param  cmp  FID to compare with objects FID
67       *  @return true if the FID's are equal
68       */
69      protected boolean eqSfi(byte cmp) {
70          return hasSfi() ? sfi == cmp : false;
71      }
72
73
74      ////////////////////////////////////////////////////////////////////////
75      //
76      //  Methods implementing filesystem commands
77      //
78
79      ////////////////////////////////////////////////////////////////////////
80      //
81      //  Constructors
82      //
83
84      /**
85       *  Creates a new instance of EF
86       *  @param _fid  file identifier for this file
87       *  @param _sfi  short fid
88       *  @param _acl  access rule object
```

```
89       */
90      protected EF(short _fid, byte _sfi, ACL _acl) {
91          super(_fid, _acl);
92          sfi = _sfi;
93      }
94  }
```

A.1.5 Apps3GPP/EF_Transparent.java

```
1   /*
2    * Transparent.java
3    *
4    * Created on 16. Februar 2003, 14:01
5    */
6   package Apps3GPP;
7   import  Apps3GPP.EF;
8
9   /**
10   *  Class for elementary files with structure transparent.
11   *
12   *  This files contain unstructured data which can be accessed via
13   *  READ/UPDATE BINARY commands
14   *
15   *  @see     <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
16   *  @author  Schroettner Robert<rs@ednet.at>
17   *  @version 1.0
18   */
19  public class EF_Transparent extends EF {
20
21      ////////////////////////////////////////////////////////////////////////
22      //
23      //  Member variables
24      //
25
26      /** databytes for this file */
27      protected byte data[];
28
29      ////////////////////////////////////////////////////////////////////////
30      //
31      //  Methods for setting/getting status
32      //
33
34      ////////////////////////////////////////////////////////////////////////
35      //
36      //  Constructors
37      //
38
39      /**
40       *  Creates a new instance of Transparent
41       *  @param _fid   file identifier for this file
42       *  @param _sfi   short fid
43       *  @param _acl   access rule object
44       *  @param _data  data bytes for this file
45       */
46      protected EF_Transparent(short _fid, byte _sfi, ACL _acl, byte[] _data) {
47          super(_fid, _sfi, _acl);
48          data = _data;
49      }
50
51  //    public static byte[] setupData(short len, byte value) {
52  //        byte ret[] = new byte[len];
53  //        for (short i=0; i<len; i++) ret[i] = value;
54  //        return ret;
55  //    }
56  }
```

A.1.6 Apps3GPP/EF_Linear.java

```
/*
 * Transparent.java
 *
 * Created on 16. Februar 2003, 14:01
 */
package Apps3GPP;
import  Apps3GPP.EF;
import  Apps3GPP.Status;

/**
 *  Class for elementary files with structure linear fixed.
 *
 *  This files contain data in fixed length records which can be accessed via
 *  READ/UPDATE/SEARCH RECORD commands
 *
 *  @see     <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
 *  @author  Schroettner Robert<rs@ednet.at>
 *  @version 1.0
 */
public class EF_Linear extends EF { // EF_Transparent {

    ////////////////////////////////////////////////////////////////////////
    //
    //  Member variables
    //

    /** databytes for this file */
    protected byte data[];
    /** length for each record */
    protected short recLen;

    ////////////////////////////////////////////////////////////////////////
    //
    //  Methods for setting/getting status
    //

    ////////////////////////////////////////////////////////////////////////
    //
    //  Constructors
    //

    /** Creates a new instance of Transparent<br>
     *  Place reliably that sizeof(_data) a multiple of _reclen<br>
     *  The created file will hold sizeof(_data) / _reclen record sets.
     *  @param _fid    file identifier for this file
     *  @param _sfi    short fid
     *  @param _acl    access rule object
     *  @param _reclen record length
     *  @param _data   data bytes for this file
     */
    protected EF_Linear(short _fid, byte _sfi, ACL _acl,
                        short _recLen, byte[] _data) {
        super(_fid, _sfi, _acl);
        data = _data;
        recLen = _recLen;

        if (_data.length / _recLen > 253)
            Status.throwIt(Status.CHK_TECHNICAL_PROBLEM);
    }
}
```

A.1.7 Apps3GPP/EF_Cyclic.java

```
1  /*
2   * EF_Cyclic.java
3   *
4   * Created on 18. Februar 2003, 00:22
5   */
6  package Apps3GPP;
7  import  Apps3GPP.EF;
8
9  /**
10  *  Class for elementary files with structure cyclic.
11  *
12  *  This files contain data in fixed length records which can be accessed via
13  *  READ/UPDATE/SEARCH RECORD and INCREASE commands
14  *
15  *  @see <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
16  *  @author ï¿½ Schrttner Robert<rs@ednet.at>
17  *  @version 1.0
18  */
19 public class EF_Cyclic extends EF_Linear {
20
21     ////////////////////////////////////////////////////////////////////////
22     //
23     //  Member variables
24     //
25
26     ////////////////////////////////////////////////////////////////////////
27     //
28     //  Methods for setting/getting status
29     //
30
31     ////////////////////////////////////////////////////////////////////////
32     //
33     //  Constructors
34     //
35
36     /**
37      *  Creates a new instance of EF_Cyclic
38      *  @param _fid    file identifier for this file
39      *  @param _sfi    short fid
40      *  @param _acl    access rule object
41      *  @param _reclen record length
42      *  @param _data   data bytes for this file
43      */
44     protected EF_Cyclic(short _fid, byte _sfi, ACL _acl,
45                         byte _recLen, byte[] _data) {
46         super(_fid, _sfi, _acl, _recLen, _data);
47     }
48 }
```

A.1.8 Apps3GPP/DF.java

```
1  /*
2   * DF.java
3   *
4   * Created on 18. Februar 2003, 00:11
5   */
6  package Apps3GPP;
7  import  Apps3GPP.File;
8
9  /**
10  *  Class for Dedicated-Files (=Directories).
11  *  DF has the same functionality as File, but can additionally
12  *  contain some child-Files (=Files and Subdirectories)
13  *  If you need a name (=AID) for your DFm use class ADF!
14  *
```

```
15   *  @see      <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
16   *  @author   Schroettner Robert<rs@ednet.at>
17   *  @version 1.0
18   */
19  public class DF extends File {
20
21      ////////////////////////////////////////////////////////////////////////
22      //
23      //  Member variables
24      //
25
26      /**
27       *  child DF's and EF's for this DF
28       */
29      private File[] childs;
30
31      ////////////////////////////////////////////////////////////////////////
32      //
33      //  Methods for setting/getting status
34      //
35
36      /**
37       *  Get child files for this DF.
38       *  @return array of childs
39       */
40      protected File[] getChilds() {
41          return childs;
42      }
43
44      /**
45       *  Set child files for this DF. Due lack of a garbage collector on JavaCard
46       *  (memory can't be freed) childs can be set only once
47       *  @param files  array of files to be used as child files
48       *  @return true if set was successful
49       */
50       private boolean setChilds(File[] files) {
51          if (childs != null)
52              // childs already defined!
53              return false;
54
55          // set childs
56          childs = files;
57
58          // update parent field in all childs
59          if (childs != null)
60              for (short i=0; i<childs.length; i++)
61                  if (childs[i] != null)
62                      childs[i].setParent(this);
63          return true;
64      }
65
66      /**
67       *  Add child file to DF
68       *  @param file  new file to install under this DF. If file is null,
69       *               only the number of free entries for files is returned.
70       *  @return number of files which still can be added
71       *          if the file can't be added, this function returns -1
72       */
73  /* -- FOR NOW, NOT REQUIRED!!!
74      protected short addChild(File file) {
75          if (childs == null)
76              // no space for childs!
77              return -1;
78
79          for (short i=0; i<childs.length; i++)
80              if (childs[i] == null) {
```

```
81                  if (file != null)
82                      file.setParent(this);
83
84                  childs[i] = file;
85                  return (short)(childs.length - i -1);
86              }
87          return -1;
88      }
89  */
90
91      ////////////////////////////////////////////////////////////////////////
92      //
93      //  Constructors
94      //
95
96      /**
97       *  Creates a new instance of DF
98       *  @param _fid     file identifier for this file
99       *  @param _acl     access rule object
100      *  @param _childs  list of childs under this DF
101      */
102     protected DF(short _fid, ACL _acl, File[] _childs) {
103         super(_fid, _acl);
104         setChilds(_childs);
105     }
106 }
```

A.1.9 Apps3GPP/ADF.java

```
1   /*
2    * ADF.java
3    *
4    * Created on 18. Februar 2003, 21:09
5    */
6
7   package Apps3GPP;
8   import  Apps3GPP.DF;
9
10  import javacard.framework.APDU;
11  import javacard.framework.Util;
12  import javacard.framework.JCSystem;
13
14  /**
15   *  Class for Application-Dedicated-Files (=Directories with AID).
16   *  ADF has the same functionality as DF, but has additionally
17   *  a name (=Application-Identifier)
18   *
19   *  According to 3GPP specification, ADF's must be registered in the
20   *  EFdir-File at MF-Level and _may_ have a parent DF.
21   *
22   *  @see <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
23   *  @author ï¿½ Schrttner Robert<rs@ednet.at>
24   *  @version 1.0
25   */
26  public abstract class ADF extends DF {
27
28      ////////////////////////////////////////////////////////////////////////
29      //
30      //  Member variables
31      //
32      /** Application is terminated.
33       */
34      public static final byte STATE_TERMINATED   = 0;
35      /** Application in currently starting up
36       */
```

```
37      //public static final byte STATE_ACTIVATING  = 1;
38      /** Application is active and ready to use
39       */
40      public static final byte STATE_ACTIVE      = 2;
41      /** Application is currently shutting down
42       */
43      //public static final byte STATE_TERMINATING = 3;
44      /** Current application state (Transient array!)
45       */
46      private byte state[];
47
48      /**
49       *  FID for selecting the ADF of the current Application
50       */
51      public static final short CURRENT_ADF = (short)0x7FFF;
52
53      /**
54       *  Name (AID) for this file
55       */
56      private byte[] name;
57
58      ////////////////////////////////////////////////////////////////////////////
59      //
60      //  Methods for setting/getting status
61      //
62
63      /** Get name (=AID) of this ADF
64       *  @return name
65       */
66      protected byte[] getName() {
67          return name;
68      }
69
70      /**
71       *  Compare if given name is equal to ADF's name if only up to
72       *  name.length characters are compared. So if this ADF has
73       *  '11 22 33...' and you compare with '11 22' this function
74       *  will return true.<br>
75       *  If this ADF's name is null, this function always returns true.
76       *  @param buffer buffer with name to compare
77       *  @param ofs    start offset for name in buffer
78       *  @param len    number of bytes in buffer to compare
79       *  @return true if names are equal
80       */
81      protected boolean eqName(byte[] buffer, short ofs, short len) {
82          if (name == null) return true;
83  //        if (len < buffer.length) return false;
84          len = len < (short)name.length ? len : (short)name.length;
85          return Util.arrayCompare(name, (short)0,
86                                   buffer, ofs, len) == 0;
87      }
88
89      protected byte getApplicationState() {
90          return state[0];
91      }
92
93      ////////////////////////////////////////////////////////////////////////////
94      //
95      //  Methods implementing filesystem commands
96      //
97
98      ////////////////////////////////////////////////////////////////////////////
99      //
100     //  Constructors
101     //
102
```

```
103     /**
104      *  Creates a new instance of ADF
105      *  @param _fid    file identifier for this ADF
106      *  @param _acl    access list
107      *  @param _name   name (=AID) for this ADF
108      *  @param _childs list of child files
109      */
110     protected ADF(short _fid, ACL _acl, byte[] _name, File[] _childs) {
111         super(_fid, _acl, _childs);
112         name = _name;
113 
114         state =
115             JCSystem.makeTransientByteArray((byte)1,
116                                             JCSystem.CLEAR_ON_DESELECT);
117         state[0] = STATE_TERMINATED;
118     }
119 
120     /**
121      *  Process APDU command<br>
122      *  @param  apdu  current processing APDU
123      *  @return true if this command was processed
124      */
125     protected abstract boolean process(APDU apdu);
126 
127     /**
128      *  called if this application has been selected
129      *  @return true if selection was successful
130      */
131     protected boolean select() {
132         state[0] = STATE_ACTIVE;
133         return true;
134     }
135 
136     /**
137      *  called if this application will be deselected
138      */
139     protected void deselect() {
140         state[0] = STATE_TERMINATED;
141     }
142 }
```

A.1.10 Apps3GPP/apduHelper.java

```
1  /*
2   *
3   * Package:  Apps3GPP
4   * Filename: APDU.java
5   * Class:    APDU
6   * Date:     04.03.2003 22:39:38
7   *
8   */
9  package Apps3GPP;
10 import  Apps3GPP.Status;
11 
12 import javacard.framework.APDU;
13 import javacard.framework.ISO7816;
14 import javacard.framework.ISOException;
15 import javacard.framework.Util;
16 
17 /**
18  *  This class is a Helper class for some commonly used functions for
19  *  for apdu processing.
20  *
21  *  @see <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
22  *  @author  Schroettner Robert<rs@ednet.at>
```

```
23   *  @version 1.0
24   */
25  abstract class apduHelper {
26      /**
27       *  Get logical channel number: bits b1,b2 of CLA.
28       *  @return channel number b1,b2 of CLA
29       */
30      protected static byte channel(APDU apdu) {
31          return (byte)(apdu.getBuffer()[ISO7816.OFFSET_CLA] & (byte)0x03);
32      }
33
34      /**
35       *  Get secure message inicator: bits b3,b4 of CLA.
36       *  @return smi b3,b4 of CLA shiftet to b1,b2
37       */
38      protected static byte smi(APDU apdu) {
39          return (byte)((apdu.getBuffer()[ISO7816.OFFSET_CLA]>>2) & (byte)0x03);
40      }
41
42      /**
43       *  Get class byte CLA with channel and smi masked out (b1-b4).
44       *  @return class byte CLA
45       */
46      protected static byte CLA(APDU apdu) {
47          return (byte)(apdu.getBuffer()[ISO7816.OFFSET_CLA] & (byte)~0x0F);
48      }
49
50      /**
51       *  Get instruction byte INS.
52       *  @return instruction byte INS
53       */
54      protected static byte INS(APDU apdu) {
55          return apdu.getBuffer()[ISO7816.OFFSET_INS];
56      }
57
58      /**
59       *  Get parameter byte P1.
60       *  @return instruparameterction byte P1
61       */
62      protected static byte P1(APDU apdu) {
63          return apdu.getBuffer()[ISO7816.OFFSET_P1];
64      }
65
66      /**
67       *  Get parameter byte P2.
68       *  @return parameter byte P2
69       */
70      protected static byte P2(APDU apdu) {
71          return apdu.getBuffer()[ISO7816.OFFSET_P2];
72      }
73
74  //    protected static short setIncomingAndReceive(APDU apdu) {
75  //        return apdu.setIncomingAndReceive();
76  //    }
77
78  //    protected static short receiveBytes(APDU apdu, byte ofs) {
79  //        return apdu.receiveBytes(ofs);
80  //    }
81
82      /**
83       *  Reset internal state. Call this function for every new
84       *  apdu processing (this function currently does nothing)
85       */
86      protected static void reset(APDU apdu) {
87          // prepare some static data for new apcu processing
88      }
```

```
89
90      /**
91       *  Call apdu's setIncomingAndReceive()-Function and return
92       *  number of available data bytes in apdu-buffer.
93       *  @return number of data bytes available in apdu buffer
94       */
95      protected static short receiveStart(APDU apdu) {
96          // receive first data block from apdu and redurn number of bytes read
97          return apdu.setIncomingAndReceive();
98      }
99
100     /**
101      *  Get up to Le data bytes from apdu into buffer.
102      *  NOTE: If there are more bytes of data available than Lc, only
103      *        Lc bytes are read. Remaining data will be ignored and
104      *        remains in the apdu buffer (or resides in the terminal).
105      *  @param apdu incuming APDU
106      *  @param out  buffer, filled with data
107      *  @param ofs  start offset in buffer
108      *  @return number of bytes read
109      *  @throws ISOException Statusword: Status.CHK_WRONG_LENGTH
110      *  @throws IndexOutOfBoundsException (out-buffer is too small for Le bytes)
111      */
112     protected static short receiveBytes(APDU apdu, byte out[], short ofs) {
113         byte buffer[] = apdu.getBuffer();
114
115         // copy data to output buffer
116         short bytesRead = receiveStart(apdu);
117         Util.arrayCopyNonAtomic(buffer, (short)ISO7816.OFFSET_CDATA,
118                                 out, ofs, bytesRead);
119         while (true) {
120             // if there are more bytes available than the apcu-buffer can hold
121             // then receive the next block now
122             short read = apdu.receiveBytes(ISO7816.OFFSET_CDATA);
123             if (read == 0) break;
124
125             // append received bytes to output buffer
126             Util.arrayCopyNonAtomic(buffer, (short)ISO7816.OFFSET_CDATA,
127                                     out, (short)(ofs+bytesRead), read);
128             bytesRead += read;
129         }
130
131         // return number of read bytes
132         return bytesRead;
133     }
134
135     /**
136      *  Check and get exactly 2 byte of data as short.
137      *  NOTE: If the apdu-buffer size is less than 2 bytes, this
138      *        function will fail. so, maybe this function will work
139      *        only for mode T=1
140      *  @param apdu incuming APDU
141      *  @return 2 bytes of data as short
142      *  @throws ISOException Statusword: Status.CHK_WRONG_LENGTH
143      */
144     protected static short receiveShort(APDU apdu) {
145         return receiveShort(apdu, receiveShort(apdu));
146     }
147
148     /**
149      *  Check and get exactly 2 byte of data as short.
150      *  NOTE: If the apdu-buffer size is less than 2 bytes, this
151      *        function will fail. so, maybe this function will work
152      *        only for mode T=1.<br>
153      *  NOTE: This function expects that you have already called
154      *        receiveStart().
```

```
155         *  @param apdu       incoming APDU
156         *  @param startBytes number of data bytes already read into apdu buffer
157         *  @return 2 bytes of data as short
158         *  @throws ISOException Statusword: Status.CHK_WRONG_LENGTH
159         */
160        protected static short receiveShort(APDU apdu, short startBytes) {
161            byte buffer[] = apdu.getBuffer();
162
163            if (startBytes != (short)2)
164                Status.throwIt(Status.CHK_WRONG_LENGTH);
165
166            // convert data bytes to short and return
167            return Util.getShort(buffer, ISO7816.OFFSET_CDATA);
168        }
169
170        /**
171         *  Check for no data. This function calls receiveStart()
172         *  and throws an ISOException if there is any data available.
173         *  @param apdu incuming APDU
174         *  @throws ISOException Statusword: Status.CHK_WRONG_LENGTH
175         */
176        protected static void receiveNothing(APDU apdu) {
177            // there must be exactly 2 bytes of data
178            if (receiveStart(apdu) != (short)0)
179                Status.throwIt(Status.CHK_WRONG_LENGTH);
180
181        }
182
183        protected static short sendStart(APDU apdu) {
184            return apdu.setOutgoing();
185        }
186
187        /**
188         *  Send one byte of response data and statusword 9000.
189         *  @param apdu  outgoing APDU
190         *  @param data  data to send
191         *  @throws ISOException Statusword: Status.CHK_WRONG_LENGTH
192         */
193        protected static void sendData(APDU apdu, byte data) {
194            short le = sendStart(apdu);
195            if (le < (short)1)
196                // le must be at least 1 byte
197                Status.throwIt(Status.CHK_WRONG_LENGTH);
198
199            apdu.setOutgoingLength((short)1); // send 1 byte
200            apdu.getBuffer()[0] = data;       // save data in apdu buffer
201            apdu.sendBytes((short)0, (short)1);
202        }
203
204        /**
205         *  Send two bytes response data and statusword 9000.
206         *  @param apdu  outgoing APDU
207         *  @param data  data to send
208         *  @throws ISOException Statusword: Status.CHK_WRONG_LENGTH
209         */
210        protected static void sendData(APDU apdu, short data) {
211            short le = sendStart(apdu);
212            if (le < (short)2)
213                // le must be at least 2 bytes
214                Status.throwIt(Status.CHK_WRONG_LENGTH);
215
216            apdu.setOutgoingLength((short)2);           // send 2 bytes
217            Util.setShort(apdu.getBuffer(),
218                          (short)0, data);              // save data in apdu buffer
219            apdu.sendBytes((short)0, (short)2);
220        }
```

```
221
222     /**
223      *  Send le bytes response data from offset ofs and statusword 9000.
224      *  @param apdu  outgoing APDU
225      *  @param data  data buffer
226      *  @param ofs   start offset for data to send
227      *  @param le    number of bytes to send
228      */
229     protected static void sendData(APDU apdu, byte[] data,
230                                    short ofs, short le) {
231         if (sendStart(apdu) < le)
232             // the expected le must be at least le bytes
233             Status.throwIt(Status.CHK_WRONG_LENGTH);
234
235         apdu.setOutgoingLength(le);
236         apdu.sendBytesLong(data, ofs, le);
237     }
238 }
```

A.1.11 Apps3GPP/TLV.java

```
1  /*
2   * TLV.java
3   *
4   * Created on 4. Mai 2003, 16:29
5   */
6  package Apps3GPP;
7
8  /**
9   *  Helper for coding/decoding TLV structures
10  *
11  *  @see     <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
12  *  @author  Schroettner Robert<rs@ednet.at>
13  *  @version 1.0
14  */
15 public class TLV {
16
17      /** Write tag and number byte for TLV object
18       *  @param  buf  Destination for writing data
19       *  @param  ofs  Position in buffer for start writing
20       *  @param  tag  Tag value
21       *  @param  len  Length value
22       *  @return new ofs after write
23       */
24      private static short _write(byte buf[], short ofs, byte tag, byte len) {
25          buf[ofs++] = tag; // Tag
26          buf[ofs++] = len; // Number of following data bytes
27          return ofs;
28      }
29
30      /** File size tag 80 (for EF's only).
31       *  <pre>
32       *   Byte 1    M  Tag 80</pre>
33       *   Byte 2    M  Length (2)</pre>
34       *   Byte 3-4  M  Number of allocated data excl. structural information
35       *  </pre>
36       */
37      public class FileSize {
38          /** Max number of bytes required for writing this tag
39           */
40          public static final short maxLength = 4;
41      }
42
43      /** Write file size tag to 80 buffer
44       *  @param  buf  Destination for writing data
```

```
45          *  @param  ofs  Position in buffer for start writing
46          *  @param  fid  File identifier
47          *  @return new ofs after write
48          */
49         public static short FileSize_write(byte buf[], short ofs, short size) {
50             ofs = _write(buf, ofs, (byte)0x80, (byte)0x02);
51             buf[ofs++] = (byte)((size & (short)0xff00) >> 8);
52             buf[ofs++] = (byte)((size & (short)0x00ff) << 0);
53             return ofs;
54         }
55
56
57         /** Total file size tag 81 (optional).
58          *  <pre>
59          *   Byte 1    M  Tag 81
60          *   Byte 2    M  Length (2)
61          *   Byte 3-4  M  Number of allocated data incl. structural information
62          *  </pre>
63          */
64         public class TotalFileSize {
65             /** Max number of bytes required for writing this tag
66              */
67             public static final short maxLength = 4;
68         }
69
70         /** Write total file size tag to 81 buffer
71          *  @param  buf  Destination for writing data
72          *  @param  ofs  Position in buffer for start writing
73          *  @param  fid  File identifier
74          *  @return new ofs after write
75          */
76         public static short TotalFileSize_write(byte buf[], short ofs, short size) {
77             ofs = _write(buf, ofs, (byte)0x81, (byte)0x02);
78             buf[ofs++] = (byte)((size & (short)0xff00) >> 8);
79             buf[ofs++] = (byte)((size & (short)0x00ff) << 0);
80             return ofs;
81         }
82
83
84         /** File descriptor tag 82 (mandatory).
85          *  <pre>
86          *   Byte 1    M  Tag 82
87          *   Byte 2    M  Length (2 or 5)
88          *   Byte 3    M  File descriptor byte
89          *   Byte 4    M  Data coding byte (=21)
90          *   Byte 5-6  C  Record length (001 - 00FF)
91          *   Byte 7    C  Number of records
92          *   C: These bytes are only mandatory for linear fixed and cyclic files
93          *  </pre>
94          */
95         public class FileDescriptor {
96             /** File descriptor bit for a NOT SHAREABLE file
97              */
98             public static final byte FD_NOT_SHAREABLE = (byte)0x00;
99             /** File descriptor bit for a SHAREABLE file       (else: NOT SHAREABLE)
100              */
101             public static final byte FD_SHAREABLE     = (byte)0x40;
102
103             /** File descriptor bits for a WORKING_EF
104              */
105             public static final byte FD_WORKING  = (byte)0x00;
106             /** File descriptor bits for a INTERNAL_EF            (else: WORKING_EF)
107              */
108             public static final byte FD_INTERNAL = (byte)0x08;
109             /** File descriptor bits for a DF                     (else: WORKING_EF)
110              */
```

```
111         public static final byte FD_DF       = (byte)0x38;
112
113         /** File descriptor bits for a NO INFORMATION GIVEN file
114          */
115         public static final byte FD_NO_INFORMATION = (byte)0x00;
116         /** File descriptor bits for a TRANSPARENT file
117          *                                          (else: No information given)
118          */
119         public static final byte FD_TRANSPARENT    = (byte)0x01;
120         /** File descriptor bits for a LINEAR FIXED file
121          *                                          (else: No information given)
122          */
123         public static final byte FD_LINEAR         = (byte)0x02;
124         /** File descriptor bits for a CYCLIC file  (else: No information given)
125          */
126         public static final byte FD_CYCLIC         = (byte)0x06;
127
128
129         /** Max number of bytes required for writing this tag
130          */
131         public static final short maxLength = 7;
132     }
133
134     /** Write file descriptor tag 82 to buffer
135      *  @param  buf  Destination for writing data
136      *  @param  ofs  Position in buffer for start writing
137      *  @param  fd   File descriptor (binary or'ed FileDescriptor.FD_XXX values)
138      *  @return new ofs after write
139      */
140     public static short FileDescriptor_write(byte buf[], short ofs, byte fd) {
141         ofs = _write(buf, ofs, (byte)0x82, (byte)0x02);
142         buf[ofs++] = fd;          // File descriptor
143         buf[ofs++] = (byte)0x21; // Data coding byte 0x21
144         return ofs;
145     }
146
147     /** Write file descriptor tag 82 to buffer
148      *  @param  buf  Destination for writing data
149      *  @param  ofs  Position in buffer for start writing
150      *  @param  fd   File descriptor (binary or'ed FD_XXX values)
151      *  @return new ofs after write
152      */
153     public static short FileDescriptor_write(byte buf[], short ofs, byte fd,
154                                 short recLength, byte numRecords) {
155         ofs = _write(buf, ofs, (byte)0x82, (byte)0x05);
156         buf[ofs++] = fd;          // File descriptor
157         buf[ofs++] = (byte)0x21; // Data coding byte 0x21
158         buf[ofs++] = (byte)((recLength & (short)0xff00) >> 8);
159         buf[ofs++] = (byte)((recLength & (short)0x00ff) << 0);
160         buf[ofs++] = numRecords;
161         return ofs;
162     }
163
164
165     /** File identifier tag 83 (mandatory).
166      *  <pre>
167      *   Byte 1    M  Tag 83
168      *   Byte 2    M  Length (2)
169      *   Byte 3-4  M  File size in bytes (excluding structural information)
170      *  </pre>
171      */
172     public class FileIdentifier {
173         /** Max number of bytes required for writing this tag
174          */
175         public static final short maxLength = 4;
176     }
```

```
177
178     /** Write file identifier tag 83 to buffer
179      *
180      *  @param  buf  Destination for writing data
181      *  @param  ofs  Position in buffer for start writing
182      *  @param  fid  File identifier
183      *  @return new ofs after write
184      */
185     public static short FileIdentifier_write(byte buf[], short ofs, short fid) {
186         ofs = _write(buf, ofs, (byte)0x83, (byte)0x02);
187         buf[ofs++] = (byte)((fid & (short)0xff00) >> 8);
188         buf[ofs++] = (byte)((fid & (short)0x00ff) << 0);
189         return ofs;
190     }
191
192
193     /** File name tag 84 (mandatory for ADF).
194      *  <pre>
195      *   Byte 1     M  Tag 84
196      *   Byte 2     M  Length (1-16)
197      *   Byte 3-18  M  File name
198      *  </pre>
199      */
200     public class FileName {
201         /** Max number of bytes required for writing this tag
202          */
203         public static final short maxLength = 18;
204     }
205
206     /** Write file identifier tag 84 to buffer
207      *  @param  buf  Destination for writing data
208      *  @param  ofs  Position in buffer for start writing
209      *  @param  aid  File name
210      *  @return new ofs after write
211      */
212     public static short FileName_write(byte buf[], short ofs, byte[] aid) {
213         byte len = (byte)((aid == null) ? 0 : aid.length);
214         ofs = _write(buf, ofs, (byte)0x84, len);
215         for (short i=0; i<len; i++)
216             buf[ofs++] = aid[i];
217         return ofs;
218     }
219
220
221     /** Short file identifier 88 (optional for EF).
222      *  <pre>
223      *   Byte 1    M  Tag 88
224      *   Byte 2    M  Length (1)
225      *   Byte 3    M  SFI value
226      *  </pre>
227      */
228     public class ShortIdentifier {
229         /** Max number of bytes required for writing this tag
230          */
231         public static final short maxLength = 3;
232     }
233
234     /** Write ShortIdentifier information tag 88 to buffer
235      *  @param  buf  Destination for writing data
236      *  @param  ofs  Position in buffer for start writing
237      *  @param  sfi  SFI value
238      *  @return new ofs after write
239      */
240     public static short ShortIdentifier_write(byte buf[], short ofs, byte sfi) {
241         ofs = _write(buf, ofs, (byte)0x88, (byte)1);
242         buf[ofs++] = sfi;
```

```
243         return ofs;
244     }
245
246
247     /** Live cycle identifier 8A (optional for mandatory).
248      *  <pre>
249      *   Byte 1     M  Tag 8A
250      *   Byte 2     M  Length (1)
251      *   Byte 3     M  Live cycle information
252      *  </pre>
253      */
254     public class LiveCycle {
255         public static final byte LC_NO_INFORMATION = (byte)0x00;
256         public static final byte LC_CREATION       = (byte)0x01;
257         public static final byte LC_INITIALIZATION = (byte)0x03;
258         public static final byte LC_ACTIVATED      = (byte)0x05;
259         public static final byte LC_DEACTIVATED    = (byte)0x04;
260         public static final byte LC_TERMINATED     = (byte)0x0C;
261
262         /** Max number of bytes required for writing this tag
263          */
264         public static final short maxLength = 3;
265     }
266
267     /** Write LiveCycle information tag 8A to buffer
268      *  @param  buf  Destination for writing data
269      *  @param  ofs  Position in buffer for start writing
270      *  @param  sfi  state (one of the LiveCycle.LC_xxx values)
271      *  @return new ofs after write
272      */
273     public static short LiveCycle_write(byte buf[], short ofs, byte state) {
274         ofs = _write(buf, ofs, (byte)0x8A, (byte)1);
275         buf[ofs++] = state;
276         return ofs;
277     }
278
279
280     /** Security attributes tag 86/8B/8C/AB (mandatory).
281      *  <pre>
282      *   Byte 1     M  Tag
283      *   Byte 2     M  Length
284      *   Byte 3..   M  data
285      *  </pre>
286      */
287     public class SecurityAttr {
288         /** Max number of bytes required for writing this tag
289          */
290         public static final short maxLength = 5;
291     }
292
293     /** Write security attribute tag 8B to buffer
294      *  @param  buf  Destination for writing data
295      *  @param  ofs  Position in buffer for start writing
296      *  @param  rnu  Record number in EFarr
297      *  @return new ofs after write
298      */
299     public static short SecurityAttr_write(byte buf[], short ofs, byte rnu) {
300         ofs = _write(buf, ofs, (byte)0x8B, (byte)3);
301         buf[ofs++] = (byte)0x2F; // FID of EFarr (always 2F06)
302         buf[ofs++] = (byte)0x06;
303         buf[ofs++] = rnu;
304         return ofs;
305     }
306
307
308     /** Proprietary information tag A5 (mandatory for MF).
```

```
309         *  <pre>
310         *   Byte 1      M  Tag A5
311         *   Byte 2      M  Length (0...x)
312         *   Byte 3-2+x  M  Proprietary data
313         *  </pre>
314         */
315        public class ProprietaryInformation {
316            /** Max number of bytes required for writing this tag
317             */
318            public static final short maxLength = 2+3;
319        }
320
321        /** Write proprietary information tag A5 to buffer
322         *  Codes always: UICC characteristigs (80), No Clock Stop allowed,
323         *                Supply Voltage A (5V) --> use this tag only for the MF
324         *  @param  buf  Destination for writing data
325         *  @param  ofs  Position in buffer for start writing
326         *  @return new ofs after write
327         */
328        public static short ProprietaryInformation_write(byte buf[], short ofs) {
329            ofs = _write(buf, ofs, (byte)0xA5, (byte)3);
330            ofs = _write(buf, ofs, (byte)0x80, (byte)1);
331            buf[ofs++] = (byte)0x10; // No Clock Stop allowed, Supply Voltage A
332            return ofs;
333        }
334
335
336        /** Pin status information tag C6 (mandatory for DF).
337         *  <pre>
338         *   Byte 1      M  Tag C6
339         *   Byte 2      M  Length (0...x)
340         *   Byte 3-2+x  M  data...
341         *  </pre>
342         */
343        public class PinStatus {
344            /** Max number of bytes required for writing this tag
345             */
346            public static final short maxLength = 2;
347        }
348
349        /** Write proprietary information tag A5 to buffer
350         *  Codes always: UICC characteristigs (80), No Clock Stop allowed,
351         *                Supply Voltage A (5V) --> use this tag only for the MF
352         *  @param  buf  Destination for writing data
353         *  @param  ofs  Position in buffer for start writing
354         *  @return new ofs after write
355         */
356        public static short PinStatus_write(byte buf[], short ofs) {
357            ofs = _write(buf, ofs, (byte)0xC6, (byte)0);
358            return ofs;
359        }
360
361
362        /** Max number of bytes required for writing tags for EF's
363         */
364        public static short maxLengthEF  = FileDescriptor.maxLength
365                                         + FileIdentifier.maxLength
366                                         + ProprietaryInformation.maxLength
367                                         + LiveCycle.maxLength
368                                         + SecurityAttr.maxLength
369                                         + FileSize.maxLength
370                                         + TotalFileSize.maxLength
371                                         + ShortIdentifier.maxLength;
372
373        /** Max number of bytes required for writing tags for DF's
374         */
```

```
375     public static short maxLengthDF = FileDescriptor.maxLength
376                                     + FileIdentifier.maxLength
377                                     + FileName.maxLength
378                                     + ProprietaryInformation.maxLength
379                                     + LiveCycle.maxLength
380                                     + PinStatus.maxLength
381                                     + TotalFileSize.maxLength;
382
383     /** Max number of bytes required for writing all tags
384      */
385 //    public static short maxLength = maxLengthEF > maxLengthDF ?
386 //                                    maxLengthEF : maxLengthDF ;
387 }
```

A.1.12 Apps3GPP/Current.java

```
1  /*
2   *
3   * Package:  Apps3GPP
4   * Filename: Current.java
5   * Class:    Current
6   * Date:     05.03.2003 02:39:24
7   *
8   */
9  package Apps3GPP;
10 import  Apps3GPP.DF;
11 import  Apps3GPP.EF;
12 import  Apps3GPP.ADF;
13 import  Apps3GPP.TLV;
14
15 import javacard.framework.JCSystem;
16 import javacard.framework.ISOException;
17
18 /**
19  * This class manages the currently seleced files, record numbers and the used
20  * channel. All data is stored in Transient-Memory (with CLEAR_ON_DESELECT)
21  *
22  *  @see     <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
23  *  @author  Schroettner Robert<rs@ednet.at>
24  *  @version 1.0
25  */
26 public class Current {
27
28     ////////////////////////////////////////////////////////////////////////
29     //
30     //  Member variables
31     //
32
33     /** initial value for selected DF */
34     private DF initDF;
35     /** initial value for selected EF */
36     private EF initEF;
37     /** initial value for selected ADF */
38     private ADF initADF;
39     /** initial value for current record number */
40     private byte initRC;
41
42     /** Number of supported channels
43      */
44     protected static final byte numChannels = (byte)4;
45
46     /**
47      *  Transient array for objects.
48      *  <ul>
49      *   <li>obj[EF+ofs]: Currently selected DF (or ADF)</li>
```

```
50        *    <li>obj[DF+ofs]: Currently selected EF - or null</li>
51        *    <li>obj[APP+ofs]: Currently selected Application - or null</li>
52        *  </ul>ofs is currentSize * Number of active channel
53        */
54       private Object obj[];
55
56       /** Offset for EF
57        */
58       private   static final byte objEF       = (byte)0;
59
60       /** Offset for DF
61        */
62       private   static final byte objDF       = (byte)1;
63
64       /** Offset for APP
65        */
66       private   static final byte objAPP      = (byte)2;
67
68       /** Number of required Entries for one channel
69        */
70       private   static final byte objSize     = (byte)3;
71
72       /**
73        *  Transient array for numbers.
74        *  <ul>
75        *  <li>number[numberCH]          stores the active channel number (0-3)</li>
76        *  <li>number[numberOP]          stores status for opened channels</li>
77        *  <li>number[numberRC+channel] stores the current record number</li>
78        *  </ul>
79        */
80       private byte number[];
81
82       /** Offset for Channel number
83        */
84       private   static final byte numberCH    = (byte)0;
85
86       /** Offset for Channel status
87        */
88       private   static final byte numberOP    = (byte)1;
89
90       /** Number of required Entries (channel independent)
91        */
92       private   static final byte numberFixed = (byte)2;
93
94       /** Offset for Record number
95        */
96       private   static final byte numberRC    = (byte)2;
97
98       /** Number of required Entries for one channel
99        */
100      private   static final byte numberSize  = (byte)1;
101
102      /**
103       *  Transient byte buffer for general usage.
104       *  Currently, this buffer is used for SEARCH BY NAME only.
105       *  The name is a AID-name and consists of RID+PIX value.
106       *  For this usage, the bufferSize is set to 16 bytes.
107       */
108      protected byte buffer[];
109
110      /**
111       *  Size of the transient byte buffer
112       */
113 //    private static final short bufferSize = TLV.maxLength > (short)16 ?
114 //                                            TLV.maxLength : (short)16 ;
115
```

```
116     /**
117      *  Create new instance of Current and allocate transient buffers.
118      *  Setup master file.
119      *  applications.
120      *  @param  _initDF   default selected DF - usually the MF.
121      *  @param  _initEF   default selected EF - usually null
122      *  @param  _initADF  default selectet application - usually null
123      *  @param  _initRC   default record number - usually 0
124      */
125     protected Current(DF _initDF, EF _initEF, ADF _initADF, byte _initRC) {
126         initDF  = _initDF;
127         initEF  = _initEF;
128         initADF = _initADF;
129         initRC  = _initRC;
130 
131         short bufferSize = TLV.maxLengthEF > TLV.maxLengthDF ?
132                            TLV.maxLengthEF : TLV.maxLengthDF ;
133 
134               bufferSize = bufferSize > 16 ?
135                            bufferSize : 16;
136 
137         // reserve memory for transient member variables
138         obj =
139             JCSystem.makeTransientObjectArray((byte)(objSize*numChannels),
140                                               JCSystem.CLEAR_ON_DESELECT);
141         number =
142             JCSystem.makeTransientByteArray  ((byte)(numberFixed
143                                                    +numberSize*numChannels),
144                                               JCSystem.CLEAR_ON_DESELECT);
145         buffer =
146             JCSystem.makeTransientByteArray  (bufferSize,
147                                               JCSystem.CLEAR_ON_DESELECT);
148     }
149 
150     /**
151      *  Set the currently used channel
152      *  @param chn  channel number
153      *  @throws ISOException Statusword: CLA_NO_LOGICAL_CHANNEL
154      */
155     protected void setChannel(byte chn) {
156         if (chn < 0 || chn > numChannels)
157             // invalid channel number
158             Status.throwIt(Status.CLA_NO_LOGICAL_CHANNEL);
159 
160         if ((number[numberOP] & (1 << chn)) == 0)
161             // channel not open
162             Status.throwIt(Status.CLA_NO_LOGICAL_CHANNEL);
163         number[numberCH] = chn;
164     }
165 
166     /**
167      *  Get current channel number
168      *  @return currently using channel number
169      */
170     protected byte getChannel() {
171         return number[numberCH];
172     }
173 
174     /**
175      *  Value for newChannel() to search for a unused channel
176      */
177     protected static final byte ANY_CHANNEL = (byte)0xFF;
178 
179     /**
180      *  Open new channel. There can be up to 4 logical channels. Channel 0
181      *  Is always opened after ATR. For opening additional channels, you must
```

```
182         *  send a "MANAGE CHANNEL". The channel number is coded in CLA bits b0, b1.
183         *
184         *  If you open a new channel from channel 0, MF is the current DF the
185         *  current application is undefined for the new channel.
186         *
187         *  If you open a new channel from channel != 0, the current DF and the
188         *  current application is the same for the new channel as for the
189         *  opening channel.
190         *
191         *  @see <a href=http://www.etsi.org>ETSI TS 121 221, Section 8.7</a>
192         *  @return new channel number
193         *  @throws ISOException {@link Status#CLA_NO_LOGICAL_CHANNEL}
194         */
195        protected byte newChannel(byte chn) throws ISOException {
196            if (chn == ANY_CHANNEL)
197            for (chn=0; chn<numChannels; chn++)
198                if ((number[numberOP] & (1 << chn)) == 0) break;
199
200            if (chn < 0 || chn > numChannels)
201                // invalid channel number
202                Status.throwIt(Status.CLA_NO_LOGICAL_CHANNEL);
203
204            if ((number[numberOP] & (1 << chn)) != 0)
205                // channel already open
206                Status.throwIt(Status.CLA_NO_LOGICAL_CHANNEL);
207
208            // open new channel: set bit number 'chn'
209            number[numberOP] |= (1 << chn);
210
211            // initialize vars for new channel depending on active channel
212            byte old = getChannel();
213            if (old != 0) {
214                DF  curDF  = getDF();
215                ADF curADF = getApplication();
216                setChannel(chn);
217                setDF(curDF);             // current DF remains active
218                setApplication(curADF);   // current ADF remains active
219            } else {
220                setChannel(chn);
221                setDF(initDF);            // MF
222                setApplication(initADF); // undefined (null)
223            }
224            setEF(initEF);                // undefined (null)
225            setRecord(initRC);            // 0
226            setChannel(old);
227            return chn;
228        }
229
230        /**
231         *  Close a logical channel
232         *  @param  chn  channel number to be closed
233         *  @throws ISOException {@link Status#CLA_NO_LOGICAL_CHANNEL}
234         */
235        protected void deleteChannel(byte chn) throws ISOException {
236            if (chn < 1 || chn > numChannels)
237                // invalid channel number
238                // NOTE: Channel 0 can't be closed
239                Status.throwIt(Status.CLA_NO_LOGICAL_CHANNEL);
240
241            // close channel: clear bit number 'chn'
242            number[numberOP] &= ~(1 << chn);
243        }
244
245        /**
246         *  Set the current record number
247         *  @param nr  record number
```

```
248         */
249        protected void setRecord(byte nr) {
250            number[numberRC+getChannel()] = nr;
251        }
252
253        /**
254         *  Get current record number
255         *  @return currently user record number
256         */
257        protected byte getRecord() {
258            return number[numberRC+getChannel()];
259        }
260
261        /**
262         *  Get object at given offset for the current channel.
263         *  @param  ofs  offset to object
264         *  @return  object at given offset for current channel number
265         */
266        private byte objIdx(byte ofs) {
267            return (byte)(ofs+getChannel()*objSize);
268        }
269
270        /**
271         *  Set the currently selected EF.
272         *  @param f  EF to make the currently selected
273         */
274        protected void setEF(EF f) {
275            byte idx = objIdx(objEF);
276            obj[idx] = f;
277        }
278
279        /**
280         *  Get the currently selected EF.
281         *  @return currently selected file (EF).<br>
282         *          returns null if no EF currently selected
283         */
284        protected EF getEF() {
285            byte idx = objIdx(objEF);
286            return (EF)obj[idx];
287        }
288
289        /**
290         *  Set the currently selected DF
291         *  @param f  DF to make the currently selected
292         */
293        protected void setDF(DF f) {
294            byte idx = objIdx(objDF);
295            if (f == null) {
296                obj[idx] = this;
297                Status.throwIt(Status.CHK_TECHNICAL_PROBLEM);
298            } else
299                obj[idx] = f;
300        }
301
302        /**
303         *  Get the currently selected DF
304         *  @return currently selected file (DF)
305         */
306        protected DF getDF() {
307            byte idx = objIdx(objDF);
308            if (obj[idx] == null) {
309                obj[idx] = initDF;
310                Status.throwIt(Status.CHK_TECHNICAL_PROBLEM);
311            }
312            return (DF)obj[idx];
313        }
```

```
314
315     /**
316      *  If EF is selected, return currentEF, else return currentDF
317      *  @return currently selected file (EF or DF)
318      */
319     protected File getFile() {
320         if (getEF() != null)
321             return getEF();
322         else
323             return getDF();
324     }
325
326     /**
327      *  Set the currently selected application
328      *  @return currently activated application (or null if none activated)
329      */
330     protected void setApplication(ADF f) {
331         byte idx = objIdx(objAPP);
332         obj[idx] = f;
333     }
334
335     /**
336      *  Get the currently selected application
337      *  @return currently activated application (or null if none activated)
338      */
339     protected ADF getApplication() {
340         byte idx = objIdx(objAPP);
341         return (ADF)obj[idx];
342     }
343 }
```

A.1.13 Apps3GPP/ReadUpdateSearch_RecordOp.java

```
1  package Apps3GPP;
2
3  /**
4   *  Helper class for extracting the File-Selection bits
5   *  and for extracting the Record-Selection mode
6   *  of P2 for READ_RECORD and UPDATE_RECORD commands
7   */
8  public class ReadUpdateSearch_RecordOp {
9
10     /**
11      *  Special SFI-Value for "use current EF"
12      */
13     protected static final byte CURRENT_EF = (byte)0x00;
14
15     /**
16      *  Get short file identifier or CURRENT<br>
17      *  The SFI is coded in b8-b4 of P2
18      *  @param  P2  P2 of current processing APDU
19      *  @return SFI or the special value {@link #CURRENT_EF}
20      */
21     protected final static byte getSFI(byte P2) {
22         return (byte)((byte)(P2 & (byte)0xF8) >> 3);
23     }
24
25     /**
26      *  Get record mode for APDU.P2<br>
27      *  Mode bits are b3-b1 of P2
28      *  @param  P2  P2 of current processing APDU
29      *  @return MODE_xxx or SEARCH_xxx values
30      */
31     protected final static byte getMode(byte P2) {
32         return (byte)(P2 & (byte)0x07);
```

```
33      }
34
35      /**
36       *  Value for APDU.P2 for record selection:
37       *    Read next Record
38       */
39      protected final static byte MODE_NEXT = (byte)2;
40
41      /**
42       *  Value for APDU.P2 for record selection:
43       *    Read previous Record
44       */
45      protected final static byte MODE_PREV = (byte)3;
46
47      /**
48       *  Value for APDU.P2 for record selection:
49       *    Read Record specified in APDU.P1
50       */
51      protected final static byte MODE_ABS  = (byte)4;
52
53      /**
54       *  Value for APDU.P2 for:
55       *  Simple serach, P1 is record number,
56       *  start forward search from record indicated in P1
57       */
58      protected final static byte SEARCH_FORWARD = (byte)3;
59
60      /**
61       *  Value for APDU.P2 for:
62       *  Simple serach, P1 is record number,
63       *  start backward search from record indicated in P1
64       */
65
66      protected final static byte SEARCH_BACKWARD = (byte)4;
67      /**
68       *  Value for APDU.P2 for:
69       *  Enhanced search
70       */
71      protected final static byte SEARCH_ENHANCED = (byte)6;
72  };
```

A.1.14 Apps3GPP/MF.java

```
1   package Apps3GPP;
2   import  Apps3GPP.ADF;
3   import  Apps3GPP.ACL;
4   import  Apps3GPP.Current;
5
6   /**
7    *  Abstract base class for MFimpl.
8    *
9    *  @see     <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
10   *  @author  Schroettner Robert<rs@ednet.at>
11   *  @version 1.0
12   */
13  abstract class MF extends ADF {
14      abstract protected Current getCurrent();
15      abstract protected ADF[] getApplications();
16      abstract protected ADF getMF();
17
18      protected MF(short _fid, ACL _acl, byte[] _name, File[] _childs) {
19          super(_fid, _acl, _name, _childs);
20      }
21  }
```

A.1.15 Apps3GPP/cmdDeActivate.java

```
1  /*
2   * cmdDeActivate.java
3   *
4   * Created on 14. April 2003, 14:00
5   */
6
7  package Apps3GPP;
8  import  Apps3GPP.ACL;
9  import  Apps3GPP.File;
10 import  Apps3GPP.Status;
11
12 import javacard.framework.APDU;
13 import javacard.framework.ISOException;
14
15 /**
16  *  This class is a collector for activating and deactivating
17  *  functions for EF's.
18  *  <br>
19  *  @see <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
20  *  @author ï¿½Schrttner Robert<rs@ednet.at>
21  *  @version 1.0
22  */
23 abstract class cmdDeActivate {
24     /**
25      *  Activate this file.
26      *  <br>
27      *  This command is for EF's only. Using this command on other file
28      *  types may result in throw ISOException(Status.CMD_INCOMPATIBLE).
29      *  @param  apdu  current processing APDU
30      *  @return true  if this command was processed
31      *  @throws ISOException(Status.CMD_INCOMPATIBLE)
32      */
33     protected static boolean activate(APDU apdu, File file)
34         throws ISOException {
35         if (!(file instanceof EF))
36             // this command is only allowed for EF's
37             Status.throwIt(Status.CMD_INCOMPATIBLE);
38
39         if (!file.isAllowed(ACL.ACCESS_ACTIVATE))
40             Status.throwIt(Status.CMD_SECURITY_NOT_SATISFIED);
41
42         file.setActivated(true);
43         return true;
44     }
45
46     /**
47      *  Deactivate file.
48      *  <br>
49      *  This command is for EF's only. Using this command on other file
50      *  types may result in throw ISOException(Status.CMD_INCOMPATIBLE).
51      *  @param  apdu  current processing APDU
52      *  @return true  if this command was processed
53      *  @throws ISOException(Status.CMD_INCOMPATIBLE)
54      */
55     protected static boolean deactivate(APDU apdu, File file)
56         throws ISOException {
57         if (!(file instanceof EF))
58             // this command is only allowed for EF's
59             Status.throwIt(Status.CMD_INCOMPATIBLE);
60
61         // deactivate current file
62         if (!file.isAllowed(ACL.ACCESS_DEACTIVATE))
63             Status.throwIt(Status.CMD_SECURITY_NOT_SATISFIED);
```

```
64
65          file.setActivated(false);
66          return true;
67      }
68  };
```

A.1.16 Apps3GPP/cmdSelect.java

```
1   /*
2    * cmdSelect.java
3    *
4    * Created on 14. April 2003, 14:00
5    */
6
7   package Apps3GPP;
8   import  Apps3GPP.Current;
9   import  Apps3GPP.EF;
10  import  Apps3GPP.Status;
11  import  Apps3GPP.ReadUpdateSearch_RecordOp;
12  import  Apps3GPP.TLV;
13  import  Apps3GPP.MF;
14
15  import javacard.framework.APDU;
16  import javacard.framework.ISOException;
17  import javacard.framework.Util;
18
19  import javacard.framework.JCSystem;
20
21  /**
22   *  This class is a collector for all select functions for files.
23   *  <br>
24   *  @see <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
25   *  @author ï¿½Schrttner Robert<rs@ednet.at>
26   *  @version 1.0
27   */
28  abstract class cmdSelect {
29
30      /**
31       *  Select EF specified by SFI in the current DF and make it the current
32       *  EF. If SFI is {@link ReadUpdateSearch_RecordOp#CURRENT_EF},
33       *  this function only returns the currently selected EF (or throws if
34       *  no EF is selected).<br>
35       *  If the search fails, the function throws the exception and lets
36       *  current selection unchanged.<br>
37       *  Selecting by SFI can be done by some commands (e.g. READ_BINARY). So
38       *  this function may be called by many files member functions.
39       *  @param   sfi  short-file-identifier
40       *  @return  selected file
41       *  @throws  ISOException {@link Status#PARA_FILE_NOT_FOUND}
42       */
43      protected static EF bySFI(byte sfi, Current cur) throws ISOException {
44          // return current EF or throw exception if no EF is selected
45          if (sfi == ReadUpdateSearch_RecordOp.CURRENT_EF) {
46              EF curEF = cur.getEF();
47              if (curEF == null) Status.throwIt(Status.PARA_FILE_NOT_FOUND);
48              return curEF;
49          }
50
51          // get list of childs in current DF
52          File childs[] = cur.getDF().getChilds();
53
54          // search for sfi in child list
55          for (short i=0; i<childs.length; i++)
56              if (childs[i] instanceof EF) { // file is EF
57                  EF ef = (EF)childs[i];
```

```
58                  if (ef.eqSfi(sfi)) {  // SFI found!
59                      cur.setEF(ef);
60                      cur.setRecord((byte)0);
61                      return ef;
62                  }
63              }
64
65          // file not found
66          Status.throwIt(Status.PARA_FILE_NOT_FOUND);
67          return null;
68      }
69
70      /**
71       *  Select the given file and return apdu-data depending on P2.
72       *  This function is used for parsing SELECT-commands. b1-b2 of P2
73       *  may be used for file selection and therefore should be cleared.
74       *  <br>
75       *  Coding of P2:
76       *  <ul><li>b8 = 0</li>
77       *      <li>b7 = X</li>
78       *      <li>b6 = X<br>
79       *      <li>b5 = 0
80       *      <li>b4-b1 = 4 for "Return FCP template"
81       *      <li>b4-b1 = 12 for "No data returned"</li></ul>
82       *  @param apdu  current apdu
83       *  @param P2    P2 of apdu with already processed bits masked out
84       *  @param f     File to be selected
85       *  @return true
86       *  @throws ISOException {@link Status#CHK_WRONG_PARAMETER_P1P2}
87       *  @throws ISOException {@link Status#CHK_TECHNICAL_PROBLEM}
88       */
89      private static boolean fileSelection(APDU apdu, byte P2,
90                                           File f, Current cur)
91              throws ISOException {
92          //
93          // parse/check control bits
94          //
95          boolean returnFCP=false;
96          // return data: b8 and b5 - b1
97          switch (P2 & 0x9F) {
98              case 0x04:
99                  // condition: "Return FCP Template"
100                 returnFCP = true;
101                 break;
102             case 0x0C:
103                 // condition: "No data returned"
104                 break;
105             default:
106                 // unknown bit setting
107                 Status.throwIt(Status.CHK_WRONG_PARAMETER_P1P2);
108         }
109         // mask out processed bits b8 and b5 - b1
110         P2 &= ~(byte)0x9F;
111
112         if (P2 != 0)
113             // there are unknown bits set
114             Status.throwIt(Status.CHK_WRONG_PARAMETER_P1P2);
115
116         //
117         // make file selection
118         //
119         if (f instanceof DF) {
120             // select DF: Set DF as current and clear EF
121             cur.setDF((DF)f   );
122             cur.setEF((EF)null);
123         }
```

```
124          else if (f instanceof EF) {
125              // select EF: Set EF as current
126              cur.setDF(((EF)f).getParent());
127              cur.setEF( (EF)f             );
128          }
129          else {
130              // unknown file type????
131              Status.throwIt(Status.CHK_TECHNICAL_PROBLEM);
132          }
133
134          //
135          // return data
136          //
137          if (returnFCP) {
138              byte buf[] = cur.buffer;
139              short pos = 0;
140
141  //            byte fd = TLV.FileDescriptor.FD_NOT_SHAREABLE;
142              byte fd = TLV.FileDescriptor.FD_SHAREABLE;
143
144              if (f instanceof DF) {
145                  DF df = (DF)f;
146                  fd |= TLV.FileDescriptor.FD_DF;
147
148                  // Write tag 82
149                  pos = TLV.FileDescriptor_write(buf, pos, fd);
150                  // Write tag 83
151                  if (df.hasFid())
152                      pos = TLV.FileIdentifier_write(buf, pos, df.getFid());
153                  // Write tag 84
154                  if (df instanceof ADF)
155                      pos = TLV.FileName_write(buf, pos, ((ADF)df).getName());
156                  // Write tag A5
157                  if (df instanceof MF)
158                      pos = TLV.ProprietaryInformation_write(buf, pos);
159                  // Write tag 8A
160                  if (df.isActivated())
161                      pos = TLV.LiveCycle_write(buf, pos,
162                                                TLV.LiveCycle.LC_ACTIVATED);
163                  else
164                      pos = TLV.LiveCycle_write(buf, pos,
165                                                TLV.LiveCycle.LC_DEACTIVATED);
166                  // Write tag 8B
167                  pos = TLV.SecurityAttr_write(buf, pos, (byte)0xff);
168                  // Write tag C6
169                  pos = TLV.PinStatus_write(buf, pos);
170                  // Write tag 81
171  //                pos = TLV.TotalFileSize_write(buf, pos,
172  //                                              (short)(sizeof(df)+sizeof(df.getChilds())));
173              } else {
174                  EF ef = (EF)f;
175                  fd |= TLV.FileDescriptor.FD_WORKING;
176  //                fd |= TLV.FileDescriptor.FD_INTERNAL;
177
178                  short recLen   = 0;
179                  short fileSize = 0;
180                  if (f instanceof EF_Linear) {
181                      recLen   = (byte)((EF_Linear)f).recLen;
182                      fileSize = (short)((EF_Linear)f).data.length;
183                      if (f instanceof EF_Cyclic) {
184                          fd |= TLV.FileDescriptor.FD_CYCLIC;
185                      } else {
186                          fd |= TLV.FileDescriptor.FD_LINEAR;
187                      }
188                  } else {
189                      fd |= TLV.FileDescriptor.FD_TRANSPARENT;
```

```
190                     fileSize   = (short)((EF_Transparent)f).data.length;
191                 }
192 
193                 // Write tag 82
194                 if (ef instanceof EF_Linear)
195                     pos = TLV.FileDescriptor_write(buf, pos, fd, recLen,
196                                                    (byte)(fileSize/recLen));
197                 else
198                     pos = TLV.FileDescriptor_write(buf, pos, fd);
199                 // Write tag 83
200                 pos = TLV.FileIdentifier_write(buf, pos, ef.getFid());
201                 // Write tag A5
202                   // not written!
203                 // Write tag 8A
204                 if (ef.isActivated())
205                     pos = TLV.LiveCycle_write(buf, pos,
206                                               TLV.LiveCycle.LC_ACTIVATED);
207                 else
208                     pos = TLV.LiveCycle_write(buf, pos,
209                                               TLV.LiveCycle.LC_DEACTIVATED);
210                 // Write tag 8B
211                 pos = TLV.SecurityAttr_write(buf, pos, (byte)0xff);
212                 // Write tag 80
213                 pos = TLV.FileSize_write(buf, pos, fileSize);
214                 // Write tag 81
215 //                pos = TLV.TotalFileSize_write(buf, pos,
216 //                                              (short)(fileSize+sizeof(df)));
217                 // Write tag 88
218                 if (ef.hasSfi())
219                     pos = TLV.ShortIdentifier_write(buf, pos, ef.getSfi());
220             }
221 
222             apduHelper.sendData(apdu, buf, (short)0, pos);
223         }
224         return true;
225     }
226 
227     /**
228      *  Select the given application and return apdu-data depending on P2.
229      *  This function is used for parsing SELECT_BY_NAME-commands. After
230      *  successfully parsing b7-b6 of P2 this function first calls
231      *  {@linkplain ADF#select()} or {@linkplain ADF#deselect()} of the
232      *   selected ADF and then {@linkplain #fileSelection fileSelection(...)}.
233      *  b1-b2 of P2 may be used for file selection and therefore should be
234      *  cleared.<br>
235      *  Coding of P2:
236      *  <ul><li>b7-b6 = 0 for "Application Activation"</li>
237      *      <li>b7-b6 = 4 for "Application Termination/Reset"</li>
238      *      <li>all ather bits are parsed in fileSelection()</li></ul>
239      *  @param apdu  current apdu
240      *  @param P2    P2 of apdu with already processed bits masked out
241      *  @param f     File to be selected
242      *  @return true
243      *  @throws ISOException {@link Status#CHK_WRONG_PARAMETER_P1P2}
244      *  @throws ISOException {@link Status#ERR_NO_INFO}
245      */
246     private static boolean applicationSelection(APDU apdu, byte P2,
247                                                 ADF f, Current cur)
248             throws ISOException {
249         //
250         // parse/check control bits and activate/terminate applications
251         //
252         // session control: b7 and b6
253         switch (P2 & 0x60) {
254             case 0x00: // application activation
255                 if (cur.getApplication() != null)
```

```
256                     // terminate currently selected application
257                     cur.getApplication().deselect();
258                 if (f.select())
259                     cur.setApplication(f);
260                 else
261                     // application selection failed
262                     Status.throwIt(Status.ERR_NO_INFO);
263                 break;
264             case 0x40: // application termination
265                 if (cur.getApplication() == f) {
266                     // if the current applicaation is the application to be
267                     // terminated, no active application remains.
268                     f.deselect();
269                     cur.setApplication((ADF)null);
270                 }
271                 else {
272                     // the given application can't be active (only one
273                     // app can be active at any time - as long as no logical
274                     // channels are supportet) -> reject command
275                     Status.throwIt(Status.ERR_NO_INFO);
276                 }
277                 break;
278             default:
279                 // unknown bit setting for b7 and b8
280                 Status.throwIt(Status.CHK_WRONG_PARAMETER_P1P2);
281         }
282         // mask out processed bits b7 and b6
283         P2 &= (byte)~0x60;
284 
285         //
286         // continue with file selection
287         //
288         return fileSelection(apdu, P2, f, cur);
289     }
290 
291     /**
292      *  Process SELECT by FID command.
293      *  Expected APDU: <pre>00 A4 00 P2 LC &lt;FID&gt; &lt;Le&gt;</pre><br>
294      *  You can select:
295      *  <ol><li>Any file which is an immediate child of the current-DF.</li>
296      *      <li>The parent of the current-DF.</li>
297      *      <li>Any DF which is an immediate child of the current-DF.</li>
298      *      <li>The current DF or ADF</li>
299      *      <li>The MF.</li></ol>
300      *  This function uses {@linkplain #fileSelection fileSelection(...)}.<br>
301      *  For details about coding please
302      *  see ETSI TS 102.221, Section 8.4.1
303      *  @param apdu  current apdu
304      *  @param P2    P2 of apdu with already processed bits masked out
305      *  @return true
306      *  @throws ISOException {@link Status#PARA_FILE_NOT_FOUND}
307      */
308     private static boolean byFID(APDU apdu, byte P2, DF mf, Current cur)
309             throws ISOException {
310         // for P1 != 0x04, b7 and b6 of P2 have no meaning and
311         // shall be set to 0 - so this bits are masked out!
312         // NOTE: Is this correct? should this bits realy be ignored?
313         //
314         P2 &= (byte)~(0x40|0x20);
315 
316         // check how many data bytes are available
317         short dataBytes = apduHelper.receiveStart(apdu);
318 
319         // special case: 'No Data' and P2=0x0C -> select MF 'short form'
320         if (P2 == 0x0C && dataBytes == 0)
321             return fileSelection(apdu, P2, mf, cur);
```

```
322
323         // get fid from apud-data
324         short fid = apduHelper.receiveShort(apdu, dataBytes);
325
326         // check for special fid 0x7FFF -> select current ADF
327         if (fid == ADF.CURRENT_ADF)
328             if (cur.getApplication() != null)
329                 // select ADF of the currently active application
330                 return fileSelection(apdu, P2, cur.getApplication(), cur);
331             else
332                 // no application activated -> select MF
333                 return fileSelection(apdu, P2, mf, cur);
334
335         // get current DF and list of childs of the current DF
336         DF curDF = cur.getDF();
337         File[] childs = curDF.getChilds();
338
339         // ad 1) any file which is an immediate child of the current-DF
340         //  -> search in current DF all immediate childs
341         //
342         if (childs!= null)
343             for (byte i=0; i<childs.length; i++)
344                 if (childs[i].eqFid(fid))
345                     // requested file found: make it the current file
346                     return fileSelection(apdu, P2, childs[i], cur);
347
348         // ad 2) the parent of the current-DF
349         //  -> compare with parent-DF
350         //
351         if (curDF.getParent() != null)
352             if (curDF.getParent().eqFid(fid))
353                 // select current parent-DF
354                 return fileSelection(apdu, P2, curDF.getParent(), cur);
355
356         // ad 3) any DF which is an immediate child of the current-DF
357         //  -> search in parent of the current DF all immediate DF's
358         //
359         if (curDF.getParent() != null) {
360             childs = curDF.getParent().getChilds();
361             if (childs!= null)
362                 for (byte i=0; i<childs.length; i++)
363                     if (childs[i] instanceof DF && childs[i].eqFid(fid))
364                         // requested DF found: make it the current file
365                         return fileSelection(apdu, P2, childs[i], cur);
366         }
367
368         // ad 4) the current DF or ADF
369         //
370         if (curDF.eqFid(fid))
371             // select current DF or ADF
372             return fileSelection(apdu, P2, curDF, cur);
373
374         // ad 5) the MF
375         //
376         if (mf.eqFid(fid))
377             // select MF (=this)
378             return fileSelection(apdu, P2, mf, cur);
379
380         // File not found!
381         Status.throwIt(Status.PARA_FILE_NOT_FOUND);
382         return true;
383     }
384
385     /**
386      *  Process SELECT by NAME command. This command is also
387      *  for activating and terminating applications.<br>
```

```
388         *  Expected APDU:
389         *    <pre>00 A4 04 P2 &lt;LC&gt; &lt;AID&gt; &lt;Le&gt;</pre><br>
390         *  You can select applications by:
391         *  <ol><li>Specifying the exact name.</li>
392         *      <li>Specifying the right truncated name and:</li>
393         *    <ol><li>First of only match.</li>
394         *        <li>Last match.</li>
395         *        <li>Next match (according to the current selected file).</li>
396         *        <li>Previous match (according to the current selected file).</li>
397         *    </ol></li>
398         *  </ol>
399         *  This function uses
400         *    {@linkplain #applicationSelection applicationSelection(...)}.<br>
401         *  For details about coding please
402         *  see: ETSI TS 102.221, Section 8.5.1<br>
403         *  @param apdu  current apdu
404         *  @param P2    P2 of apdu with already processed bits masked out
405         *  @return true
406         *  @throws ISOException {@link Status#PARA_FILE_NOT_FOUND}
407         */
408        private static boolean byAID(APDU apdu, byte P2,
409                                     ADF[] apps, Current cur)
410                throws ISOException {
411            if (apps == null)
412                // no ADF's found
413                Status.throwIt(Status.PARA_FILE_NOT_FOUND);
414
415            short len;
416            try {
417                len = apduHelper.receiveBytes(apdu, cur.buffer, (short)0);
418            } catch (IndexOutOfBoundsException e) {
419                // got more than 16 bytes for AID -> ignore this bytes
420                len = (short)cur.buffer.length;
421            }
422
423            // AID selection control: b2 and b1
424            byte selectMode = (byte)(P2 & (byte)0x03);
425            P2 &= (byte)~0x03;
426
427            short i;
428            switch (selectMode) {
429                case (byte)0:
430                    // Select first or only occurence
431                    for (i=0; i<apps.length; i++)
432                        // compare AID sent with AID of apps[i] but compare only
433                        // up to apps[i].AID.length bytes. so if you have a
434                        // configuration like:
435                        //   AID1: 11 22 33 44
436                        //   AID2: 11 22 33 55
437                        //   AID3: 11 22 33
438                        // Store longer AID's at lower index in apps!
439                        if (apps[i].eqName(cur.buffer, (short)0, len))
440                            // AID found -> select ADF!
441                            return applicationSelection(apdu, P2, apps[i], cur);
442                    break;
443
444                case (byte)1:
445                    // Select last occurence
446                    for (i=(short)(apps.length-1); i>=0; i--)
447                        if (apps[i].eqName(cur.buffer, (short)0, len))
448                            // AID found -> select ADF!
449                            return applicationSelection(apdu, P2, apps[i], cur);
450                    break;
451
452                case (byte)2:
453                    // Select next occurence (after the currently selected)
```

```
454                 i=0;
455                 for (; i<apps.length; i++)
456                     if (apps[i] == cur.getApplication())
457                                 // AID of currently selected found
458                                 break;
459 
460                 for (++i; i<apps.length; i++)
461                     if (apps[i].eqName(cur.buffer, (short)0, len))
462                         // AID found -> select ADF!
463                         return applicationSelection(apdu, P2, apps[i], cur);
464                 break;
465 
466             case (byte)3:
467                 // Select previous occurence (before the currently selected)
468                 i=(short)(apps.length-1);
469                 for (; i>=0; i--)
470                     if (apps[i] == cur.getApplication())
471                         // AID of currently selected found
472                         break;
473 
474                 for (--i; i>=0; i--)
475                     if (apps[i].eqName(cur.buffer, (short)0, len))
476                         // AID found -> select ADF!
477                         return applicationSelection(apdu, P2, apps[i], cur);
478                 break;
479         }
480 
481         Status.throwIt(Status.PARA_FILE_NOT_FOUND);
482         return true;
483     }
484 
485     /**
486      *  Select by path from MF or current DF.
487      *  This function uses {@linkplain #fileSelection fileSelection(...)}.<br>
488      *  For details about coding please
489      *  see: ETSI TS 102.221, Section 8.4.2
490      *  @param apdu    current apdu
491      *  @param P2      P2 of apdu with already processed bits masked out
492      *  @param curFile current selected file (EF or DF)
493      *  @return true
494      *  @throws ISOException {@link Status#PARA_FILE_NOT_FOUND}
495      *  @throws ISOException {@link Status#CHK_WRONG_LENGTH}
496      */
497     private static boolean byPath(APDU apdu, byte P2,
498                                   File curFile, Current cur) {
499         // for P1 != 0x04, b7 and b6 of P2 have no meaning and
500         // shall be set to 0 - so this bits are masked out!
501         // NOTE: Is this correct? should this bits realy be ignored?
502         //
503         P2 &= (byte)~(0x40|0x20);
504 
505         // receive bytes in nameBuffer
506         // NOTE: This will limit path to 8 entires (nameBuffer.length/2)
507         //       but wont restrict functionality since path depth
508         //       dont exceed 8.
509         short len  = apduHelper.receiveBytes(apdu, cur.buffer, (short)0);
510         short curIdx=0;
511         while (curIdx < len && curFile != null) {
512             if (len - curIdx < (short)2)
513                 // odd number of data bytes???
514                 Status.throwIt(Status.CHK_WRONG_LENGTH);
515 
516             // convert next 2 bytes to short
517             short fid = Util.getShort(cur.buffer, curIdx);
518             curIdx += 2;
519 
```

```
520             // check for special fid 0x7FFF -> select current ADF
521             if (fid == ADF.CURRENT_ADF) {
522                   curFile = cur.getApplication();
523                   break;
524             }
525
526             // search for fid in childs of current DF
527             if (curFile instanceof DF) {
528                 File[] childs = ((DF)curFile).getChilds();
529                 if (childs!= null)
530                     for (byte i=0; i<childs.length; i++)
531                         if (childs[i].eqFid(fid)) {
532                             // requested file found
533                             curFile = childs[i];
534                             break;
535                         }
536             }
537
538             // path entry not found!
539             curFile = null;
540         }
541
542         if (curFile != null)
543             return fileSelection(apdu, P2, curFile, cur);
544
545         Status.throwIt(Status.PARA_FILE_NOT_FOUND);
546         return true;
547     }
548
549     /**
550      *  Process SELECT command.
551      *  This function is the main function for all SELECT commands.<br>
552      *  Be sure that CLA and INS are correct - no further tests are achived!
553      *  This function uses {@linkplain #fileSelection fileSelection(...)}
554      *  and {@linkplain #applicationSelection applicationSelection(...)}.<br>
555      *  For details about coding please
556      *  see: ETSI TS 102.221, Section 8
557      *  @param  apdu  current processing APDU
558      *  @return true if this command was processed
559      *  @throws ISOException {@link Status#CHK_WRONG_LENGTH}
560      *  @throws ISOException {@link Status#PARA_FUNCTION_NOT_SUPPORTED}
561      */
562     protected static boolean doSelect(APDU apdu, MF mf) {
563         byte P1 = apduHelper.P1(apdu);
564         byte P2 = apduHelper.P2(apdu);
565         Current cur = mf.getCurrent();
566
567         switch (P1) {
568             case (byte)0x00:
569                 return byFID(apdu, P2, mf.getMF(), cur);
570
571             case (byte)0x01:
572                 // Select child DF from current DF
573                 //  ----> which one? the first? the last?
574                 // CURRENTLY NOT IMPLEMENTED!
575
576                 // for P1 != 0x04, b7 and b6 of P2 have no meading and
577                 // shall be set to 0 - so this bits are masked out!
578                 // NOTE: Is this correct? should this bits realy be ignored?
579                 //
580                 P2 &= (byte)~(0x40|0x20);
581
582                 Status.throwIt(Status.PARA_FUNCTION_NOT_SUPPORTED);
583                 return true;
584
585             case (byte)0x03:
```

```
586                 // Select parent DF current DF
587
588                 // for P1 != 0x04, b7 and b6 of P2 have no meading and
589                 // shall be set to 0 - so this bits are masked out!
590                 // NOTE: Is this correct? should this bits realy be ignored?
591                 //
592                 P2 &= (byte)~(0x40|0x20);
593
594                 DF curDF = cur.getDF();
595                 if (curDF != null)
596                     if (curDF.getParent() != null)
597                         return fileSelection(apdu, P2,
598                                              curDF.getParent(), cur);
599
600                 // no parent file found (already at MF or ADF without FID)
601                 Status.throwIt(Status.PARA_FILE_NOT_FOUND);
602                 return true;
603
604             case (byte)0x04:
605                 // default coding according 3GPP
606                 // NOTE: This coding is problematic OpenCard compliant cards
607                 //       because this cards uses 00 A0 04 xx for selecting
608                //        Applets.
609             case (byte)0x0A:
610                 // aternate coding - may be required for OpenCard compliant
611                 // cards
612                 return byAID(apdu, P2, mf.getApplications(), cur);
613
614             case (byte)0x08:
615                 // Select by path from MF
616                 // See: ETSI TS 102.221, Section 8.4.2
617                 return byPath(apdu, P2, mf.getMF(), cur);
618
619             case (byte)0x09:
620                 // Select by path from current DF
621                 // See: ETSI TS 102.221, Section 8.4.2
622                 return byPath(apdu, P2, cur.getDF(), cur);
623         }
624
625         // function not implemented
626         // (overloading of this command ist not allowed!)
627         Status.throwIt(Status.PARA_FUNCTION_NOT_SUPPORTED);
628         return true;
629     }
630 };
```

A.1.17 Apps3GPP/cmdTransparent.java

```
1  /*
2   * cmdTransparent.java
3   *
4   * Created on 14. April 2003, 14:00
5   */
6  package Apps3GPP;
7  import  Apps3GPP.apduHelper;
8  import  Apps3GPP.File;
9  import  Apps3GPP.EF_Transparent;
10 import  Apps3GPP.Status;
11
12 import javacard.framework.APDU;
13 import javacard.framework.ISOException;
14 import javacard.framework.APDU;
15 import javacard.framework.ISOException;
16
17 /**
```

```
18   *  This class is a collector for functions which access file data in
19   *  unstructured files (readBinary, updateBinary, ...).
20   *  <br>
21   *  @see     <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
22   *  @author  Schroettner Robert<rs@ednet.at>
23   *  @version 1.0
24   */
25  abstract class cmdTransparent {
26
27      ////////////////////////////////////////////////////////////////////////
28      //
29      //  Methods implementing filesystem commands for update file data for
30      //    transparent files
31      //
32
33      /**
34       *  Read Binary.
35       *  <br>
36       *  This command is for EF_Transparent-Files only.
37       *  Using this command on other file types may result in
38       *  throw ISOException(Status.CMD_INCOMPATIBLE).
39       *  @param  apdu  current processing APDU
40       *  @param  ofs   start offset to read bytes
41       *  @return true  if this command was processed
42       *  @throws ISOException(Status.CMD_INCOMPATIBLE)
43       */
44      protected static boolean read(APDU apdu, Current cur)
45              throws ISOException {
46          byte P1 = apduHelper.P1(apdu);
47          byte P2 = apduHelper.P2(apdu);
48          short ofs;
49          File file = cur.getFile();
50
51          if ((P1 & 0x80) == 0) {
52              // b8 of P1 = 0 -> P2/P1 is offset
53              ofs  = (short) (P2 & 0xff);
54              ofs |= (short)((P1 & 0xff) << 8);
55          } else {
56              // b8 of P1 = 1 -> b5-b1 of P1 = SFI, P2 = ofs
57              ofs = P2;
58  //            cur = findSFI((byte)(P1 & 0x1F), current.getDF());
59              file = cmdSelect.bySFI((byte)(P1 & 0x1F), cur);
60          }
61
62          // check if selecte file is of right type
63          if (!(file instanceof EF_Transparent))
64              Status.throwIt(Status.CMD_INCOMPATIBLE);
65
66          EF_Transparent ef = (EF_Transparent)file;
67
68          // check access conditions
69          if (!ef.isAllowed(ACL.ACCESS_READ))
70              Status.throwIt(Status.CMD_SECURITY_NOT_SATISFIED);
71
72          short le = apduHelper.sendStart(apdu);
73
74          if (le + ofs > ef.data.length) {
75              le = (short)(ef.data.length - ofs);
76              if (le > 0)
77                  apduHelper.sendData(apdu, ef.data, ofs, le);
78              // less data available than in le requested -> return warning
79              Status.throwIt(Status.WARN_EOF_REACHED);
80          } else {
81              apduHelper.sendData(apdu, ef.data, ofs, le);
82          }
83
```

```
84          // read and return data
85          return true;
86      }
87
88      /**
89       *  Update Binary.
90       *  <br>
91       *  This command is for EF_Transparent-Files only.
92       *  Using this command on other file types may result in
93       *  throw ISOException(Status.CMD_INCOMPATIBLE).
94       *  @param  apdu  current processing APDU
95       *  @param  ofs   start offset to write bytes
96       *  @return true  if this command was processed
97       *  @throws ISOException(Status.CMD_INCOMPATIBLE)
98       */
99  //    protected static boolean update(APDU apdu, short ofs, File file)
100     protected static boolean update(APDU apdu, Current cur)
101             throws ISOException {
102         byte P1 = apduHelper.P1(apdu);
103         byte P2 = apduHelper.P2(apdu);
104         short ofs;
105         File file = cur.getFile();
106
107         if ((P1 & 0x80) == 0) {
108             // b8 of P1 = 0 -> P2/P1 is offset
109             ofs  = (short) (P2 & 0xff);
110             ofs |= (short)((P1 & 0xff) << 8);
111         } else {
112             // b8 of P1 = 1 -> b5-b1 of P1 = SFI, P2 = ofs
113             ofs = P2;
114 //            cur = findSFI((byte)(P1 & 0x1F), current.getDF());
115             file = cmdSelect.bySFI((byte)(P1 & 0x1F), cur);
116         }
117
118         // check if selecte file is of right type
119         if (!(file instanceof EF_Transparent))
120             Status.throwIt(Status.CMD_INCOMPATIBLE);
121
122         EF_Transparent ef = (EF_Transparent)file;
123
124         // check access conditions
125         if (!ef.isAllowed(ACL.ACCESS_UPDATE))
126             Status.throwIt(Status.CMD_SECURITY_NOT_SATISFIED);
127
128         short lc = apduHelper.receiveStart(apdu);
129
130         // update data
131         return true;
132     }
133
134 };
```

A.1.18 Apps3GPP/cmdRecord.java

```
1  /*
2   * cmdRecord.java
3   *
4   * Created on 14. April 2003, 14:00
5   */
6  package Apps3GPP;
7  import  Apps3GPP.apduHelper;
8  import  Apps3GPP.Current;
9  import  Apps3GPP.EF;
10 import  Apps3GPP.EF_Linear;
11 import  Apps3GPP.EF_Cyclic;
```

```
12 import   Apps3GPP.EF_Transparent;
13 import   Apps3GPP.Status;
14
15 import javacard.framework.APDU;
16 import javacard.framework.ISOException;
17
18 /**
19  *  This class is a collector for functions which access file data in
20  *  record based files (readRecord, writeRecord, ...).
21  *  <br>
22  *  @see <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
23  *  @author ï¿½Schrttner Robert<rs@ednet.at>
24  *  @version 1.0
25  */
26 abstract class cmdRecord {
27
28     /**
29      *  Get select record specified in P2.
30      *  @param  P1    P1 of current APDU
31      *  @param  P2    P2 of current APDU
32      *  @param  cur   Current state
33      *  @return selected record
34      */
35     private static byte ReadUpdateSearch_selectRecord(byte P1, byte P2,
36                                                       Current cur)
37             throws ISOException {
38         byte rec = 0;
39
40         // get selected file
41         EF_Linear file = null;
42         try {
43             file = (EF_Linear)cur.getEF();
44         } catch (ClassCastException e) {
45             // only EF_Linear or EF_Cyclic are record based
46             Status.throwIt(Status.CMD_INCOMPATIBLE);
47         }
48
49         // get select record specified in P2
50         switch (ReadUpdateSearch_RecordOp.getMode(P2)) {
51             case ReadUpdateSearch_RecordOp.MODE_ABS:
52                 if (P1 == 0)
53                     // only return current record
54                     return cur.getRecord();
55                 rec = P1; // use record number specified in P1
56                 break;
57
58             case ReadUpdateSearch_RecordOp.MODE_NEXT:
59                 rec = cur.getRecord();
60                 rec++;
61 //                if (rec == (byte)0xFF)
62 //                    // max record number is 0xff
63 //                    Status.throwIt(Status.PARA_RECORD_NOT_FOUND);
64                 break;
65
66             case ReadUpdateSearch_RecordOp.MODE_PREV:
67                 rec = cur.getRecord();
68                 if (rec == 0)
69                     // calculate last+1 record
70                     rec = (byte)(file.data.length / file.recLen +2);
71                 rec--;
72 //                if (rec == (byte)0x00)
73 //                    // max record number is 0xff
74 //                    Status.throwIt(Status.PARA_RECORD_NOT_FOUND);
75 //                if (rec * recLen > data.length)
76 //                    // out of data
77 //                    Status.throwIt(Status.PARA_RECORD_NOT_FOUND);
```

```
78                  break;
79
80              default:
81                  Status.throwIt(Status.PARA_INCORRECT_P1P2);
82          }
83
84          if (file instanceof EF_Cyclic) {
85              // handle wrap around of record pointer for EF_Cyilc
86              if (rec == 0)
87                  rec = (byte)(file.data.length / file.recLen);
88
89              if (rec * file.recLen >= file.data.length)
90                  rec = 1;
91          } else {
92              // check for EOF
93              if (rec == 0 || rec * file.recLen > file.data.length)
94                  // out of data
95                  Status.throwIt(Status.PARA_RECORD_NOT_FOUND);
96          }
97
98          // set new current record and return
99          cur.setRecord(rec);
100         return rec;
101     }
102
103     /**
104      *  Get SFI from P2 and select EF.
105      *  The SFI is coded in b8-b4 of P2
106      *  - this coding is used in READ_RECORD, UPDATE_RECORD and SEARCH_RECORD
107      *  @param  P2    P2 of current APDU
108      *  @param  cur   Current state
109      *  @return selected EF
110      */
111     private static EF_Linear ReadUpdateSearch_selectFile(byte P2, Current cur)
112             throws ISOException {
113         byte sfi = ReadUpdateSearch_RecordOp.getSFI(P2);
114         try {
115             return (EF_Linear)cmdSelect.bySFI(sfi, cur);
116         } catch (ClassCastException e) {
117             // only EF_Linear or EF_Cyclic may be selected by this function
118             Status.throwIt(Status.CMD_INCOMPATIBLE);
119             return null;
120         }
121     }
122
123     /**
124      *  Get SFI from P2 and select EF.
125      *  The SFI is coded in b5-b1 of P2 if b8 of P2 =0
126      *  - this coding is used in INCREASE only
127      *  @param  P2    P2 of current APDU
128      *  @param  cur   Current's class
129      *  @return selected EF
130      */
131     private static EF_Cyclic Increase_selectFile(byte P2, Current cur) {
132
133         byte sfi;
134         if ((P2 & 0x80) == 0)
135             sfi=ReadUpdateSearch_RecordOp.CURRENT_EF;
136         else
137             sfi=(byte)(P2 & 0x1F);
138
139         File file = null;
140
141         file = cmdSelect.bySFI(sfi, cur);
142
143         try {
```

```
144             return (EF_Cyclic)file;
145         } catch (ClassCastException e) {
146             // only EF_Cyclic may be selected by this function
147             Status.throwIt(Status.CMD_INCOMPATIBLE);
148             return null;
149         }
150     }
151 
152 
153     ////////////////////////////////////////////////////////////////////////
154     //
155     //  Methods implementing filesystem commands for update file data for
156     //    record based files
157     //
158 
159 
160     /**
161      *  Read Record.
162      *  <br>
163      *  This command is for EF_Linear and EF_Cyclic-Files only.
164      *  Using this command on other file types may result in
165      *  throw ISOException(Status.CMD_INCOMPATIBLE).
166      *  @param  apdu    current processing APDU
167      *  @param  cur     Current-Values-Class
168      *  @return true    if this command was processed
169      *  @throws ISOException(Status.CMD_INCOMPATIBLE)
170      */
171 //    protected static boolean read(APDU apdu, MF root)
172     protected static boolean read(APDU apdu, Current cur)
173             throws ISOException {
174 //        if (!(file instanceof EF_Linar))
175 //            // this command is only allowed for EF's
176 //            Status.throwIt(Status.CMD_INCOMPATIBLE);
177         byte P1 = apduHelper.P1(apdu);
178         byte P2 = apduHelper.P2(apdu);
179 
180         // get sfi from P2 and select file
181         //  if function selects a EF_Transparent file this function
182         //  throw ISOException(Status.CMD_INCOMPATIBLE)
183         EF_Linear file = ReadUpdateSearch_selectFile(P2, cur);
184 
185         if (!file.isAllowed(ACL.ACCESS_READ))
186             Status.throwIt(Status.CMD_SECURITY_NOT_SATISFIED);
187 
188         // get select record specified in P2
189         //  this functions throws for EF_Transparend files!
190         byte rec = ReadUpdateSearch_selectRecord(P1, P2, cur);
191 
192         // get number of expected bytes to return
193         short le = apduHelper.sendStart(apdu);
194 
195         // calculate file offset in bytes
196         rec--;
197         short ofs = (short)(rec * file.recLen);
198 //        if (ofs > data.length) {
199 //            cur.setRecord((byte)(data.length/recLen -1));
200 //            Status.throwIt(Status.PARA_RECORD_NOT_FOUND);
201 //        }
202 
203         if (le > file.recLen) {
204             // truncate due EOF
205             le = file.recLen;
206             apduHelper.sendData(apdu, file.data, ofs, le);
207             Status.throwIt(Status.WARN_EOF_REACHED);
208         } else {
209             // return expected number of bytes of data
```

```
210             apduHelper.sendData(apdu, file.data, ofs, le);
211         }
212 
213         // read and return record
214         return true;
215     }
216 
217     /**
218      *  Update Record.
219      *  <br>
220      *  This command is for EF_Linear and EF_Cyclic-Files only.
221      *  Using this command on other file types may result in
222      *  throw ISOException(Status.CMD_INCOMPATIBLE).
223      *  @param  apdu    current processing APDU
224      *  @param  cur     Current-Values-Class
225      *  @return true    if this command was processed
226      *  @throws ISOException(Status.CMD_INCOMPATIBLE)
227      */
228 //    protected static boolean update(APDU apdu, MF root)
229     protected static boolean update(APDU apdu, Current cur)
230             throws ISOException {
231 //        if (!(file instanceof EF_Linar))
232 //            // this command is only allowed for EF's
233 //            Status.throwIt(Status.CMD_INCOMPATIBLE);
234 
235         byte P1 = apduHelper.P1(apdu);
236         byte P2 = apduHelper.P2(apdu);
237 
238         // get sfi from P2 and select file
239         //  if function selects a EF_Transparent file this function
240         //  throw ISOException(Status.CMD_INCOMPATIBLE)
241         EF_Linear file = ReadUpdateSearch_selectFile(P2, cur);
242 
243         if (!file.isAllowed(ACL.ACCESS_UPDATE))
244             Status.throwIt(Status.CMD_SECURITY_NOT_SATISFIED);
245 
246         // get select record specified in P2
247         //  this functions throws for EF_Transparend files!
248         byte rec = ReadUpdateSearch_selectRecord(P1, P2, cur);
249 
250         // get number of expected bytes to return
251         short le = apduHelper.sendStart(apdu);
252 
253         // calculate file offset in bytes
254         rec--;
255         short ofs = (short)(rec * file.recLen);
256 
257         short lc = apduHelper.receiveStart(apdu);
258 
259         // TBD...
260 
261         // update record
262         return true;
263     }
264 
265     /**
266      *  Search Record.
267      *  <br>
268      *  This command is for EF_Linear and EF_Cyclic-Files only.
269      *  Using this command on other file types may result in
270      *  throw ISOException(Status.CMD_INCOMPATIBLE).
271      *  @param  apdu  current processing APDU
272      *  @param  MF    Master file
273      *  @return true  if this command was processed
274      *  @throws ISOException(Status.CMD_INCOMPATIBLE)
275      */
```

```
276     protected static boolean search(APDU apdu, Current cur)
277 //    protected boolean search(APDU apdu, MF root)
278             throws ISOException {
279 //        if (!(file instanceof EF_Linar))
280 //            // this command is only allowed for EF's
281 //            Status.throwIt(Status.CMD_INCOMPATIBLE);
282 
283         byte P1 = apduHelper.P1(apdu);
284         byte P2 = apduHelper.P2(apdu);
285 
286         // get sfi from P2 and select file
287         //  if function selects a EF_Transparent file this function
288         //  throw ISOException(Status.CMD_INCOMPATIBLE)
289         EF_Linear file = ReadUpdateSearch_selectFile(P2, cur);
290 
291         if (!file.isAllowed(ACL.ACCESS_READ))
292             Status.throwIt(Status.CMD_SECURITY_NOT_SATISFIED);
293 
294         switch (ReadUpdateSearch_RecordOp.getMode(P2)) {
295             case ReadUpdateSearch_RecordOp.SEARCH_FORWARD:
296                 // record number to start serach is in P1 (00 = current record)
297                 Status.throwIt(Status.PARA_FUNCTION_NOT_SUPPORTED);
298                 break;
299 
300             case ReadUpdateSearch_RecordOp.SEARCH_BACKWARD:
301                 // record number to start serach is in P1 (00 = current record)
302                 Status.throwIt(Status.PARA_FUNCTION_NOT_SUPPORTED);
303                 break;
304 
305             case ReadUpdateSearch_RecordOp.SEARCH_ENHANCED:
306                 Status.throwIt(Status.PARA_FUNCTION_NOT_SUPPORTED);
307                 break;
308 
309             default:
310                 Status.throwIt(Status.PARA_INCORRECT_P1P2);
311         }
312 
313         // TBD...
314 
315         // search record
316         return true;
317     }
318 
319     /**
320      *  Increase.
321      *  <br>
322      *  This command is for EF_Cyclic-Files only.
323      *  Using this command on other file types may result in
324      *  throw ISOException(Status.CMD_INCOMPATIBLE).
325      *  @param  apdu  current processing APDU
326      *  @param  cur   Current data
327      *  @return true  if this command was processed
328      *  @throws ISOException(Status.CMD_INCOMPATIBLE)
329      */
330     protected static boolean increase(APDU apdu, Current cur)
331             throws ISOException {
332 //        byte P1 = apduHelper.P1(apdu);
333         byte P2 = apduHelper.P2(apdu);
334 
335         // get sfi from P2 and select file
336         //  if function selects a EF_Transparent file this function
337         //  throw ISOException(Status.CMD_INCOMPATIBLE)
338         EF_Cyclic file = Increase_selectFile(P2, cur);
339 
340         if (!file.isAllowed(ACL.ACCESS_READ))
341             Status.throwIt(Status.CMD_SECURITY_NOT_SATISFIED);
```

```
342
343         // TBD...
344
345         // increase current record
346         return true;
347     }
348 };
```

A.1.19 Apps3GPP/Commands.java

```
1  /*
2   * Commands.java
3   *
4   * Created on 22. Februar 2003, 01:39
5   */
6  package Apps3GPP;
7
8  /**
9   *  List of CLA and INS bytes for Commands
10  *
11  *  @see      <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0, Section 10.1.2</a>
12  *  @author  Schroettner Robert<rs@ednet.at>
13  *  @version 1.0
14  */
15 public class Commands {
16     /**
17      *  APDU-Command-List of instructions (INS)
18      *  for class 0x00 (CLA)
19      */
20     public class CLA00 {
21         /** Class Byte 0x00
22          */
23         public static final byte CLA                = (byte)0x00;
24
25         /** SELECT FILE (CLA=0x00, INS=0xA4)
26          */
27         public static final byte SELECT             = (byte)0xA4;
28
29         /** READ BINARY (CLA=0x00, INS=0xB0)
30          */
31         public static final byte READ_BINARY        = (byte)0xB0;
32
33         /** UPDATE BINARY (CLA=0x00, INS=0xD6)
34          */
35         public static final byte UPDATE_BINARY      = (byte)0xD6;
36
37         /** READ RECORD (CLA=0x00, INS=0xB2)
38          */
39         public static final byte READ_RECORD        = (byte)0xB2;
40
41         /** UPDATE RECORD (CLA=0x00, INS=0xDC)
42          */
43         public static final byte UPDATE_RECORD      = (byte)0xDC;
44
45         /** SEARCH RECORD (CLA=0x00, INS=0xA2)
46          */
47         public static final byte SEARCH_RECORD      = (byte)0xA2;
48
49         /** VERIFY (CLA=0x00, INS=0x20)
50          */
51         public static final byte VERIFY             = (byte)0x20;
52
53         /** CHANGE PIN (CLA=0x00, INS=0x24)
54          */
55         public static final byte CHANGE_PIN         = (byte)0x24;
```

```
56
57          /** DISABLE PIN (CLA=0x00, INS=0x26)
58           */
59          public static final byte DISABLE_PIN        = (byte)0x26;
60
61          /** ENABLE PIN (CLA=0x00, INS=0x28)
62           */
63          public static final byte ENABLE_PIN         = (byte)0x28;
64
65          /** UNBLOCK PIN (CLA=0x00, INS=0x2C)
66           */
67          public static final byte UNBLOCK_PIN        = (byte)0x2C;
68
69          /** DEACTIVATE FILE (CLA=0x00, INS=0x04)
70           */
71          public static final byte DEACTIVATE_FILE    = (byte)0x04;
72
73          /** ACTIVATE FILE (CLA=0x00, INS=0x44)
74           */
75          public static final byte ACTIVATE_FILE      = (byte)0x44;
76
77          /** AUTHENTICATE (CLA=0x00, INS=0x88)
78           */
79          public static final byte AUTHENTICATE       = (byte)0x88;
80
81          /** GET CHALLENGE (CLA=0x00, INS=0x84)
82           */
83          public static final byte GET_CHALLENGE      = (byte)0x84;
84
85          /** MANAGE CHANNEL (CLA=0x00, INS=0x70)
86           */
87          public static final byte MANAGE_CHANNEL     = (byte)0x70;
88
89          /** GET RESPONSE (CLA=0x00, INS=0xC0)
90           */
91          public static final byte GET_RESPONSE       = (byte)0xC0;
92      }
93
94      /**
95       *  APDU-Command-List of instructions (INS)
96       *  for class 0x80 (CLA)
97       */
98      public class CLA80 {
99          /** Class Byte 0x80
100          */
101         public static final byte CLA                = (byte)0x80;
102
103         /** INCREASE (CLA=0x80, INS=0x32)
104          */
105         public static final byte INCREASE           = (byte)0x32;
106
107         /** TERMINAL PROFILE (CLA=0x80, INS=0x10) - only for USAT
108          */
109         //public static final byte TERMINAL_PROFILE  = (byte)0x10;
110
111         /** ENVELOPE (CLA=0x80, INS=0xC2) - only for USAT
112          */
113         //public static final byte ENVELOPE          = (byte)0xC2;
114
115         /** FETCH (CLA=0x80, INS=0x12) - only for USAT
116          */
117         //public static final byte FETCH             = (byte)0x12;
118
119         /** TERMINAL RESPONSE (CLA=0x80, INS=0x14) - only for USAT
120          */
121         //public static final byte TERMINAL_RESPONSE = (byte)0x14;
```

```
122     }
123 }
```

A.1.20 Apps3GPP/MFimpl.java

```
1  /*
2   * MFimpl.java
3   *
4   * Created on 18. Februar 2003, 00:24
5   */
6  package Apps3GPP;
7  import  Apps3GPP.MF;
8  import  Apps3GPP.Current;
9  import  Apps3GPP.ADF;
10 import  Apps3GPP.DF;
11 import  Apps3GPP.EF;
12 import  Apps3GPP.apduHelper;
13 import  Apps3GPP.Commands;
14
15 import javacard.framework.APDU;
16
17 /**
18  *  This class implements the filesystems root node (The master file MF).
19  *  It parses APDU's distributes the calls to the file classes.
20  *
21  *  This class provides a MF (0x3F00) and holds the root of the file tree.
22  *  The tree can be built by DF's and by ADF's (with FID)
23  *  All ADF's should be passed to the constructor in the apps-array
24  *  (for working 'selection by name').
25  *
26  *  <pre>
27  *   File[] childs = ... // List of files and dirs at MF level
28  *   AFD[] adfs    = ... // List of adfs at MF level
29  *
30  *   MF root = new MF(childs, adfs);
31  * </pre>
32  *
33  *  and for every incoming APDU call:
34  *  <pre>
35  *    javacard.framework.APDU apdu = ... // incoming apdu
36  *    root.process(apdu);
37  * </pre>
38  *
39  *  @see     <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
40  *  @author  Schrioettner Robert<rs@ednet.at>
41  *  @version 1.0
42  */
43 //public class MFimpl extends ADF implements Current.MF {
44 public class MFimpl extends MF {
45
46     ////////////////////////////////////////////////////////////////////////
47     //
48     //  Member variables
49     //
50
51     /**
52      *  Current values (selected File, selected Record Number, ...).
53      */
54     protected Current current;
55
56     /**
57      *  Return current data.
58      *  (Implementation for abstract class MF)
59      *  @return current data
60      */
```

```
61      protected Current getCurrent() {
62          return current;
63      }
64
65      /**
66       *  Default FID on MF for TS 102 221 compilant filesystem root (3F00).
67       */
68      public static final short defaultMF = (short)0x3F00;
69
70      /**
71       *  List of installed applications (=ADF's).
72       */
73      private ADF apps[] = null;
74
75      /**
76       *  Return list of applications.
77       *  (Implementation for abstract class MF)
78       *  @return list of all ADF's
79       */
80      protected ADF[] getApplications() {
81          return apps;
82      }
83
84      /**
85       *  Return MF (= this).
86       *  (Implementation for abstract class MF)
87       *  @return MF
88       */
89      protected ADF getMF() {
90          return this;
91      }
92
93      ////////////////////////////////////////////////////////////////////////
94      //
95      //  Methods for setting/getting status
96      //
97
98      /**
99       *  Add applications (=ADF)
100      *  @param app  new application to install under this DF. If app is null,
101      *              only the number of free entries for application is returned.
102      *  @return number of files which still can be added
103      *          if the application can't be added, this function returns -1
104      */
105 /* -- FOR NOW, NOT REQUIRED!!!
106     public short addApp(ADF app) {
107         if (apps == null)
108             // no space for applications!
109             return -1;
110
111         for (short i=0; i<apps.length; i++)
112             if (apps[i] == null) {
113                 if (app != null)
114                     app.setParent(this);
115
116                 apps[i] = app;
117                 return (short)(apps.length - i -1);
118             }
119         return -1;
120     }
121 */
122     ////////////////////////////////////////////////////////////////////////
123     //
124     //  Methods implementing filesystem commands
125     //
126
```

```
127     /**
128      *  Process MANAGE CHANNEL command.
129      *  Channel management for logical channels is done by this function and
130      *  a APDU with MANAGE_CHANNEL command.
131      *  Be sure that CLA and INS are correct - no further tests are achived!<br>
132      *  @param  apdu  current processing APDU
133      *  @return true if this command was processed
134      *  @throws ISOException Statusword: Status.CHK_WRONG_LENGTH
135      *  @throws ISOException Statusword: Status.CHK_WRONG_PARAMETER_P1P2
136      */
137     private boolean cmdManage(APDU apdu) {
138         byte P1 = apduHelper.P1(apdu);
139         byte P2 = apduHelper.P2(apdu);
140 
141         switch (P1) {
142             case (byte)0x00: // open new channel
143                 if (P2 != 0)
144                     Status.throwIt(Status.CHK_WRONG_PARAMETER_P1P2);
145 
146                 apduHelper.receiveNothing(apdu);
147                 byte chn = current.newChannel(current.ANY_CHANNEL);
148                 // return number of new opened channel
149                 apduHelper.sendData(apdu, chn);
150                 return true;
151 
152             case (byte)0x80: // close channel
153                 apduHelper.receiveNothing(apdu);
154                 current.deleteChannel(P2);
155                 return true;
156         }
157 
158         // function not implemented
159         // (overloading of this command ist not allowed!)
160         Status.throwIt(Status.CHK_WRONG_PARAMETER_P1P2);
161         return true;
162     }
163 
164     /**
165      *  Process APDU command.
166      *  This is the main function for all incoming APDU's. Its job is mainly
167      *  to parse CLA/INS and delegate the work to other function.
168      *  @param  apdu  current processing APDU
169      *  @return true if this command was processed
170      */
171     protected boolean process(APDU apdu) {
172         current.setChannel(apduHelper.channel(apdu));
173 
174         if (apduHelper.smi(apdu) != 0) {
175             // Secure Cnalles not yes supported!
176             Status.throwIt(Status.CLA_NO_SECURE_MESSAGING);
177             return true;
178         }
179 
180         // run application specific apdu processing
181         if (current.getApplication() != null)
182             if (current.getApplication().process(apdu))
183                 return true;
184 
185         //
186         // fallback to default command processing
187         //
188 
189         // get Class-Byte and mask out logical channels
190         byte CLA = apduHelper.CLA(apdu);
191 
192         // get Instruction-Byte
```

```
193         byte INS = apduHelper.INS(apdu);
194
195         File cur = current.getFile();
196         short ofs;
197         byte P1, P2;
198
199         if (CLA == Commands.CLA00.CLA) switch (INS) {
200             case Commands.CLA00.SELECT:
201                 return cmdSelect.doSelect(apdu, this);
202
203             // TODO: move selectSFI to files methods
204             case Commands.CLA00.READ_BINARY:
205                 return cmdTransparent.read(apdu, current);
206             case Commands.CLA00.UPDATE_BINARY:
207                 return cmdTransparent.update(apdu, current);
208
209             case Commands.CLA00.READ_RECORD:
210                 return cmdRecord.read(apdu, current);
211             case Commands.CLA00.UPDATE_RECORD:
212                 return cmdRecord.update(apdu, current);
213             case Commands.CLA00.SEARCH_RECORD:
214                 return cmdRecord.search(apdu, current);
215
216             case Commands.CLA00.VERIFY:
217             case Commands.CLA00.CHANGE_PIN:
218             case Commands.CLA00.DISABLE_PIN:
219             case Commands.CLA00.ENABLE_PIN:
220             case Commands.CLA00.UNBLOCK_PIN:
221                 // CURRENTLY NOT IMPLEMENTED!
222                 Status.throwIt(Status.CHK_INS_NOT_SUPPORTED);
223                 return true;
224
225             case Commands.CLA00.DEACTIVATE_FILE:
226                 return cmdDeActivate.deactivate(apdu, current.getFile());
227             case Commands.CLA00.ACTIVATE_FILE:
228                 return cmdDeActivate.activate(apdu, current.getFile());
229
230             case Commands.CLA00.AUTHENTICATE:
231             case Commands.CLA00.GET_CHALLENGE:
232                 // CURRENTLY NOT IMPLEMENTED!
233                 Status.throwIt(Status.CHK_INS_NOT_SUPPORTED);
234                 return true;
235
236             case Commands.CLA00.MANAGE_CHANNEL:
237                 return cmdManage(apdu);
238
239             case Commands.CLA00.GET_RESPONSE:
240                 // CURRENTLY NOT IMPLEMENTED!
241                 Status.throwIt(Status.CHK_INS_NOT_SUPPORTED);
242                 return true;
243
244             default:
245                 // run application specific apdu processing
246                 if (current.getApplication() != null)
247                     if (current.getApplication().process(apdu))
248                         return true;
249
250                 Status.throwIt(Status.CHK_INS_NOT_SUPPORTED);
251                 return false;
252         }
253
254         if (CLA == Commands.CLA80.CLA) switch (INS) {
255             // alternate class byte for 'SELECT BY NAME' to
256             // avoid conflicts with OpenCard's 'SELECT BY NAME'
257             case Commands.CLA00.SELECT:
258                 return cmdSelect.doSelect(apdu, this);
```

```
259
260             case Commands.CLA80.INCREASE:
261                 if (current.getEF() != null)
262 //                    return current.getEF().cmdIncrease(apdu, this);
263                     return cmdRecord.increase(apdu, current);
264
265                 Status.throwIt(Status.CHK_INS_NOT_SUPPORTED);
266                 return false;
267
268             // THE FOLLOWING COMMANDS AR FOR USAT UICC's ONLY
269             //case Commands.CLA80.TERMINAL_PROFILE:
270             //case Commands.CLA80.ENVELOPE:
271             //case Commands.CLA80.FETCH:
272             //case Commands.CLA80.TERMINAL_RESPONSE:
273             //    // CURRENTLY NOT IMPLEMENTED!
274             //    ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
275             //    return true;
276
277             default:
278                 // run application specific apdu processing
279                 if (current.getApplication() != null)
280                     if (current.getApplication().process(apdu))
281                         return true;
282
283                 Status.throwIt(Status.CHK_INS_NOT_SUPPORTED);
284                 return false;
285         }
286
287         // command not processed (unknown class)
288         Status.throwIt(Status.CHK_CLA_NOT_SUPPORTED);
289         return true; // processed or not processed?
290 //        return false;
291     }
292
293     ////////////////////////////////////////////////////////////////////////////
294     //
295     //  Constructors
296     //
297
298     /**
299      *  Creates a new instance of Filesystem.
300      *  For a TS 102 221 compilant filesystem FID of the MF is set to defaultMF.
301      *  @param _childs    list of child files for this root
302      *  @param _apps      list of applications under this root
303      */
304     protected MFimpl(File[] _childs, ADF[] _apps) {
305         super(defaultMF, (ACL)null, (byte[])null, _childs);
306         apps = _apps;
307
308         // allocate transient buffer for AID search
309 //        short maxApp = 0;
310 //        if (apps!= null)
311 //            for (short i=0; i<apps.length; i++)
312 //                if (apps[i] != null)
313 //                if (maxApp > apps[i].getName().length)
314 //                    maxApp = (short)apps[i].getName().length;
315
316         // for 'SELECT BY NAME' reserve a data buffer of max 16 bytes
317         // to store the AID to compare with AID's stored in apps[]
318         // max AID length: (RID + PIX) = 16 bytes
319
320         current = new Current((DF)this, (EF)null, (ADF)null, (byte)0);
321     }
322
323     /**
324      *  Called if this applet has been selected.
```

```
325         *  @return true if selection was successful
326         */
327        protected boolean select() {
328            current.newChannel((byte)0); // open channel 0
329            return true;
330        }
331
332        /**
333         *  Called if this applet will be deselected.
334         */
335        protected void deselect() {
336        }
337    }
```

A.1.21 Apps3GPP/USIM.java

```
1   /*
2    * adf.java
3    *
4    * Created on 22. Februar 2003, 20:39
5    */
6
7   package Apps3GPP;
8   import  Apps3GPP.ACL;
9   import  Apps3GPP.ADF;
10  import  Apps3GPP.File;
11  import  Apps3GPP.EF_Transparent;
12
13  import javacard.framework.APDU;
14
15  /**
16   *  Incomplete implementation for an USIM-Application.
17   *
18   *  @see <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0</a>
19   *  @see <a href=http://www.etsi.org>ETSI TS 131 102 V4.2.0</a>
20   *  @author  Schroettner Robert<rs@ednet.at>
21   *  @version 1.0
22   */
23  public class USIM extends ADF {
24
25      /**
26       *  Create Child Files for ISIM Application
27       *  according 3GPP TS 31.103, V5.2.0, Section 4.2
28       */
29      private static File[] createChilds() {
30          return new File[] {
31
32              // EFli (Language Indication)
33              //  This EF contains the codes for one or more languages.
34              //
35              // Update activity: low
36              //
37              new EF_Transparent(             // Transparent EF, optional!
38                  (short)0x6F05,              // FID
39                  (byte)0x02,                 // SFI
40                  new ACL(ACL.lvALW, ACL.lvPIN, // Access-Rules (read/update/
41                          ACL.lvADM, ACL.lvADM),//  deactivate/activate)
42                  new byte[] {                // Data: 2-byte language codes
43                      (byte)0xFF, (byte)0xFF, // 1st language code
44                      (byte)0x07, (byte)0xFF, // 2nd language code ...
45              }),
46
47              // many other files TBD...
48
49          }; // END new File[]
```

```
50      }
51
52      /**
53       *  Create AID for ISIM Application
54       *  according 3GPP TS ETSI TS 101 220 V5.1.0
55       */
56      private static byte[] createAID() {
57          return new byte[] { // A0 00 00 00 87 10 02
58              (byte)0xA0, (byte)0x00, // RID for 3GPP
59              (byte)0x00, (byte)0x00,
60              (byte)0x87,
61              (byte)0x10, (byte)0x02, // App-Code for USIM
62              (byte)0xff, (byte)0x43, // Country Code: Austria
63              (byte)0xff, (byte)0xff, // App-Provider Code (Card issuser)
64              (byte)0x89,             // App-Provider Code (=Telecom)
65              (byte)0x04, (byte)0x03, // Specification Version 4.3.0
66              (byte)0x00,
67              (byte)0x00,             // App-Provider specific data
68          };
69      }
70
71      /**
72       *  get minimum AID-Length to select ISIM
73       */
74      public static byte getMinAIDlength() {
75          return 7; // just RID + App-Code
76      }
77
78      /**
79       *  Creates a new instance of adf
80       */
81      public USIM() {
82          super(
83              ADF.NO_FID, // no FID for ISIM-Application
84              (ACL)null,      // no ACL's required
85              createAID(),
86              createChilds());
87      }
88
89      /**
90       *  Process APDU command<br>
91       *  @param  apdu  current processing APDU
92       *  @return true if this command was processed
93       */
94      protected boolean process(APDU apdu) {
95          return false; // no additional commands
96      }
97
98      /**
99       *  called if this application will be deselected
100      *  @return true if selection was successful
101      */
102     public void deselect() {
103         super.deselect();
104     }
105
106     /**
107      *  called if this application has been selected
108      */
109     public boolean select() {
110         if (!super.select()) return false;
111
112         // do some important things on activation...
113         return true;
114     }
115 }
```

A.1.22 Apps3GPP/ISIM.java

```
1  /*
2   * adf.java
3   *
4   * Created on 22. Februar 2003, 20:39
5   */
6
7  package Apps3GPP;
8  import  Apps3GPP.ACL;
9  import  Apps3GPP.ADF;
10 import  Apps3GPP.File;
11 import  Apps3GPP.EF_Transparent;
12 import  Apps3GPP.EF_Linear;
13
14 import javacard.framework.APDU;
15
16 /**
17  *  Implementation for a ISIM-Application.
18  *
19  *  @see <a href=http://www.etsi.org>ETSI TS 121 221 V5.2.0</a>
20  *  @see <a href=http://www.etsi.org>ETSI TS 131 103 V5.2.0</a>
21  *  @author ï¿½ Schrttner Robert<rs@ednet.at>
22  *  @version 1.0
23  */
24 public class ISIM extends ADF {
25
26     /**
27      *  Create Child Files for ISIM Application
28      *  according 3GPP TS 31.103, V5.2.0, Section 4.2
29      */
30     private static File[] createChilds() {
31         return new File[] {
32
33             // EFkeys (Ciphering and Integtity Keys for IMS)
34             //  This EF contains the ciphering key CK, the integrity key IK
35             //  and the key set identifier KSI for the IP Multimedia Subsystem.
36             //
37             // Update activity: high
38             // Size: 33 bytes
39             //
40             new EF_Transparent(                 // Transparent EF, mandatory!
41                 (short)0x6F08,                  // FID
42                 (byte)0x08,                     // SFI
43                 new ACL(ACL.lvPIN, ACL.lvPIN,   // Access-Rules (read/update/
44                         ACL.lvADM, ACL.lvADM),//  deactivate/activate)
45                 new byte[] {                    // Data:
46                     (byte)0x00,                 // Key Set Identifier KSI (bits 0-3)
47                     (byte)0x07, (byte)0xFF,     // default Ciphering Key CK
48                     (byte)0x00, (byte)0xFF,
49                     (byte)0xFF, (byte)0xFF,
50                     (byte)0x00, (byte)0xFF,
51                     (byte)0x07, (byte)0xFF,     // default Integrity Key IK
52                     (byte)0xFF, (byte)0xFF,
53                     (byte)0xFF, (byte)0xFF,
54                     (byte)0xFF, (byte)0xFF
55             }),
56
57             // EFimpi (IMS private identifier)
58             //  This EF contains the private SIP Identity (SIP URI) of the user.
59             //
60             // Update activity: low
61             //
62             new EF_Transparent(                 // Transparent EF, mandatory!
63                 (short)0x6F02,                  // FID
```

```
64                  (byte)0x02,                     // SFI
65                  new ACL(ACL.1vPIN, ACL.1vADM, // Access-Rules (read/update/
66                          ACL.1vADM, ACL.1vADM),//  deactivate/activate)
67                  new byte[] {                    // Data: URI TLV data object
68                      (byte)0x80,                 // TAG-Value = 80
69                      (byte)0x00, (byte)0xff      // Default: No Data
70              }),
71
72              // EFdomain (SIP domain URI)
73              //  This EF contains the SIP entry point in the home operators
74              //  network, if different from the host part of the private SIP URI
75              //  of the user from file EFIMPI.
76              //
77              // Update activity: low
78              //
79              new EF_Transparent(             // Transparent EF, mandatory!
80                  (short)0x6F03,              // FID
81                  (byte)0x05,                 // SFI
82                  new ACL(ACL.1vPIN, ACL.1vADM, // Access-Rules (read/update/
83                          ACL.1vADM, ACL.1vADM),//  deactivate/activate)
84                  new byte[] {                // Data: URI TLV data object
85                      (byte)0x80,             // TAG-Value = 80
86                      (byte)0x00, (byte)0xff  // Default: No Data
87              }),
88
89              // EFimpu (IMS public Identifier of user)
90              //   This EF contains one or more public SIP Identities (SIP URI)
91              //   of the user.
92              //
93              // Update activity: low
94              //
95              new EF_Transparent(             // Transparent EF, mandatory!
96                  (short)0x6F04,              // FID
97                  (byte)0x04,                 // SFI
98                  new ACL(ACL.1vPIN, ACL.1vADM, // Access-Rules (read/update/
99                          ACL.1vADM, ACL.1vADM),//  deactivate/activate)
100                 new byte[] {                // Data: URI TLV data object
101                     (byte)0x80,             // TAG-Value = 80
102                     (byte)0x00, (byte)0xff  // Default: No Data
103             }),
104
105             // EFad (Administrative Data)
106             //  This EF contains information concerning the mode of operation
107             //  according to the type of ISIM, such as normal (to be used by IMS
108             //  subscribers for IMS operations), type approval (to allow
109             //  specific use of the Terminal during type approval procedures of
110             //  e.g. the network equipment), manufacturer specific (to allow the
111             //  Terminal manufacturer to perform specific proprietary auto-test
112             //  in its Terminal during e.g. maintenance phases).
113             //
114             //  It also provides an indication of whether some Terminal features
115             //  should be activated during normal operation.
116             //
117             // Update activity: low
118             //
119             new EF_Transparent(             // Transparent EF, mandatory!
120                 (short)0x6FAD,              // FID
121                 (byte)0x03,                 // SFI
122                 new ACL(ACL.1vALW, ACL.1vADM, // Access-Rules (read/update/
123                         ACL.1vADM, ACL.1vADM),//  deactivate/activate)
124                 new byte[] {                // Data
125                     (byte)0x00,             // Mode of operation: Normal-Op
126                     (byte)0xff,             // Additional Information: RFU
127                     (byte)0xff              //                         RFU
128             }),
129
```

```
130             // EFarr (Access Rule Reference)
131             //  This EF contains the access rules for files located under the
132             //  ISIM ADF in the UICC. If the security attribute tag '8B' is
133             //  indicated in the FCP it contains a reference to a record in this
134             //  file.
135             //
136             // Update activity: low
137             //
138             new EF_Linear(                      // Linear-Fixes EF, mandatory!
139                 (short)0x6F06,                  // FID
140                 (byte)0x06,                     // SFI
141                 new ACL(ACL.lvALW, ACL.lvADM, // Access-Rules (read/update/
142                         ACL.lvADM, ACL.lvADM),//  deactivate/activate)
143                 (short)1,                       // Record length
144                 new byte[]{                     // Data: Access Rule TLV data objects
145                     (byte)0xff                  // TBD.
146             }),
147         }; // END new File[]
148     }
149
150     /**
151      *  Create AID for ISIM Application
152      *  according 3GPP TS ETSI TS 101 220 V5.1.0
153      */
154     private static byte[] createAID() {
155         return new byte[] { // A0 00 00 00 87 10 04
156             (byte)0xA0, (byte)0x00, // RID for 3GPP
157             (byte)0x00, (byte)0x00,
158             (byte)0x87,
159             (byte)0x10, (byte)0x04, // App-Code for ISIM
160             (byte)0xff, (byte)0x43, // Country Code: Austria
161             (byte)0xff, (byte)0xff, // App-Provider Code (Card issuser)
162             (byte)0x89,             // App-Provider Code (=Telecom)
163             (byte)0x05, (byte)0x05, // Specification Version 5.5.0
164             (byte)0x00,
165             (byte)0x00,             // App-Provider specific data
166         };
167     }
168
169     /**
170      *  get minimum AID-Length to select ISIM
171      */
172     public static byte getMinAIDlength() {
173         return 7; // just RID + App-Code
174     }
175
176     /**
177      *  Creates a new instance of adf
178      */
179     public ISIM() {
180         super(
181             ADF.NO_FID, // no FID for ISIM-Application
182             (ACL)null,      // no ACL's required
183             createAID(),
184             createChilds());
185     }
186
187     /**
188      *  Process APDU command<br>
189      *  @param  apdu  current processing APDU
190      *  @return true if this command was processed
191      */
192     protected boolean process(APDU apdu) {
193         return false; // no additional commands
194     }
195
```

```
196     /**
197      *  called if this application will be deselected
198      *  @return true if selection was successful
199      */
200     public void deselect() {
201         super.deselect();
202     }
203 
204     /**
205      *  called if this application has been selected
206      */
207     public boolean select() {
208         if (!super.select()) return false;
209 
210         // do some important things on activation...
211         return true;
212     }
213 }
```

A.1.23 Apps3GPP/main.java

```
1  /*
2   * main.java
3   *
4   * Created on 23. Februar 2003, 01:25
5   */
6  package Apps3GPP;
7  import  Apps3GPP.MF;
8  import  Apps3GPP.MFimpl;
9  import  Apps3GPP.File;
10 import  Apps3GPP.EF_Linear;
11 import  Apps3GPP.EF_Transparent;
12 import  Apps3GPP.USIM;
13 import  Apps3GPP.ISIM;
14 import  Apps3GPP.ADF;
15 import  Apps3GPP.ACL;
16 
17 import javacard.framework.APDU;
18 import javacard.framework.Applet;
19 import javacard.framework.ISOException;
20 
21 /**
22  *  Main class for Applet
23  *
24  *  @see     <a href=http://www.etsi.org>ETSI TS 121 221 V5.2.0</a>
25  *  @author  Schroettner Robert<rs@ednet.at>
26  *  @version 1.0
27  */
28 public class main extends Applet {
29 
30     /** maximum number of files under root (MR)
31      */
32 //    public static final byte MAX_ROOT_CHILDS = 5;
33 
34     /** maximum number of applications (ADF's)
35      */
36 //    public static final byte MAX_ROOT_APPS   = 3;
37 
38     /**
39      *  Root filesystem (=MF)
40      */
41     protected MF root;
42 
43     /**
44      *  Create Child Files for ISIM Application at MF-Level
```

```
45       *  according ETSI TS 102 221, Section 13
46       */
47      private static File[] createChilds() {
48          return new File[] {
49              // EFdir
50              //  This EF consists of one ore more records, which each record able
51              //  to hold one entry. Each entry in the EFdir is an application
52              //  template Data Object (DO) as defined in ISO/IEC 7815-5.
53              //  An application template DO is a constructed BER-TLV object with
54              //  a maximum length of 127 bytes and has a mandatory AID DO.
55              //
56              // Update activity: low
57              //
58              new EF_Linear(                      // Linear fixed EF, mandatory!
59                  (short)0x2F00,                  // FID
60                  (byte)0x1E,                     // SFI
61                  new ACL(ACL.lvALW, ACL.lvADM, // Access-Rules (read/update/
62                          ACL.lvADM, ACL.lvADM),//  deactivate/activate)
63                  (short)1,                       // Record length
64                  new byte[] {                    // Data: BER-TLV objects
65              }),
66
67              // EFiccid (ICC Identication)
68              //  This EF provides a unique identifcation number for the UICC.
69              //
70              // Update activity: low
71              //
72              new EF_Transparent(                 // Transparent EF, mandatory!
73                  (short)0x2FE2,                  // FID
74                  (byte)0x02,                     // SFI
75                  new ACL(ACL.lvALW, ACL.lvNEV, // Access-Rules (read/update/
76                          ACL.lvADM, ACL.lvADM),//  deactivate/activate)
77                  new byte[] {                    // Data: Identification number
78                      (byte)0xFF, (byte)0xFF,
79                      (byte)0xFF, (byte)0xFF,
80                      (byte)0xFF, (byte)0xFF,
81                      (byte)0xFF, (byte)0xFF,
82                      (byte)0xFF, (byte)0xFF,
83              }),
84
85              // EFpl (Preferred Languages)
86              //  This EF contains the codes for up to n languages. This
87              //  information, determined by the user/operator, defines
88              //  the preferred languages of the user, for the UICC, in order
89              //  of priority.
90              //
91              // Update activity: low
92              //
93              new EF_Transparent(                 // Transparent EF, mandatory!
94                  (short)0x2F05,                  // FID
95                  (byte)0x05,                     // SFI
96                  new ACL(ACL.lvALW, ACL.lvPIN, // Access-Rules (read/update/
97                          ACL.lvADM, ACL.lvADM),//  deactivate/activate)
98                  new byte[] {                    // Data: Identification number
99                      (byte)0xFF, (byte)0x43,   // 1st language: AT
100                     (byte)0xFF, (byte)0xFF,   // 2nd language: unused ...
101             }),
102
103             // EFarr (Access Rule References)
104             //  This EF contains access rules for the file located under the
105             //  MF in the UICC. If the security tag '8B' is indicated in the
106             //  FCP it contains a reference to a record in this file.
107             //
108             // Update activity: low
109             //
110             new EF_Linear(                      // Linear fixed EF, mandatory!
```

```
111                 (short)0x2F06,                  // FID
112                 (byte)0x06,                     // SFI
113                 new ACL(ACL.lvALW, ACL.lvADM,   // Access-Rules (read/update/
114                         ACL.lvADM, ACL.lvADM),//  deactivate/activate)
115                 (byte)1,                        // Record length
116                 new byte[]{                     // Data: Access Rule TLV data objects
117                     (byte)0x00, (byte)0x01
118             }),
119         };
120     }
121
122     /**
123      * Only this class's install method should create the applet object.
124      */
125     protected main(byte[] bArray, short bOffset, byte bLength) {
126 //        if (bArray != null && bArray[bOffset] != 0) {
127             // no install data allowed (in this version)!
128
129             // don't throw error -> just ignore install data
130             //Status.throwIt(Status.PARA_FUNCTION_NOT_SUPPORTED);
131 //        }
132
133         byte[] aid=new byte[] {     // AID is 3GPP registered RID
134             (byte)0xA0, (byte)0x00, (byte)0x00, (byte)0x00, (byte)0x87
135         };
136
137         // create File structures for Applications
138         USIM usim = new USIM();
139         ISIM isim = new ISIM();
140
141         // setup test-ADF's
142         ADF test[] = new ADF[4];
143         for (short i=0; i<test.length; i++)
144            test[i] =
145                new ADF((short)(0x7F40 + i), (ACL)null,
146                        new byte[]{(byte)0xA0, (byte)0x00, (byte)0x00,
147                                   (byte)0x00, (byte)0x87,
148                                   (byte)0x7F, (byte)(0x40+i)},
149                        (File[])null) {
150                    protected boolean process(APDU apdu) {
151                        // no special apdu processing for this application!
152                        return false;
153                    }
154                };
155
156         // File files[] = new File[] { };
157         // ADF  apps[]  = new ADF[]  { usim, isim };
158
159         // setup filesystem root and reserve space for child-files and apps
160         //new Base( (byte[])null, (File[])null , (ADF[])null );
161 //        root = new MFimpl(aid, // files, apps);
162         root = new MFimpl(
163 //          root = new MF(// files, apps);
164 //                        new File[MAX_ROOT_CHILDS],
165 //                        new ADF[MAX_ROOT_APPS]);
166                         createChilds(),
167                         new ADF[] { test[0], test[1],
168                                     test[2], test[3],
169                                     usim   , isim   });
170
171         // add ISIM Application to Filesystem
172 //        root.addApp(usim);
173 //        root.addApp(isim);
174
175         // be sure, the applet is registered by the right AID (=3GPP's RID)
176         //
```

```
177             register(aid, (short)0, (byte)aid.length);
178
179             // register with given AID
180 //            register(bArray, (short)(bOffset + 1), bArray[bOffset]);
181
182             // register applet with min AID from ISIM-ADF
183 //            register();
184 //            register(usim.getName(), (short)0, usim.getMinAIDlength());
185 //            register(isim.getName(), (short)0, isim.getMinAIDlength());
186         }
187
188         /**
189          *  Installs this applet.
190          *  @param bArray   the array containing installation parameters
191          *  @param bOffset  the starting offset in bArray
192          *  @param bLength  the length in bytes of the parameter data in bArray
193          */
194         public static void install(byte[] bArray, short bOffset, byte bLength) {
195             new main(bArray, bOffset, bLength);
196         }
197
198 //      public static void main(String args[]) {
199 //          Object oa[] = new Object[3];
200 //          File fa[] = (File[])oa;
201 //
202 //          //install((byte[])null, (short)0, (byte)0);
203 //      }
204
205         /**
206          *  Process apdu messages (if no other application is selected)
207          *  @param apdu  current processing apdu
208          */
209         public void process(APDU apdu) throws ISOException {
210             if (selectingApplet()) return;
211             // let the filesystem distribute APDU's
212             apduHelper.reset(apdu);
213             root.process(apdu);
214         }
215
216         /**
217          *  called if this application will be deselected
218          *  @return true if selection was successful
219          */
220         public void deselect() {
221             root.deselect();
222         }
223
224         /**
225          *  called if this application has been selected
226          */
227         public boolean select() {
228             return root.select();
229 //          return true;
230         }
231 }
```

A.2 Test Client

A.2.1 Tester/Connect.java

```
1  /*
2   * Connect.java
3   *
4   * Created on 1. Mai 2003, 16:34
```

```
 */
package Tester;

import com.linuxnet.jpcsc.Card;
import com.linuxnet.jpcsc.Context;
import com.linuxnet.jpcsc.PCSC;
import com.linuxnet.jpcsc.State;

/**
 *  Class for connection establishment via PC/SC
 *
 *  @author  Schroettner Robert<rs@ednet.at>
 *  @version 1.0
 */
public class Connect {

    private Context ctx;          // PCSC-Context
    private String  readerList[]; // list of available readers
    private String  reader;       // selected Reader
    private Card    card;         // selected Card

    /** Creates a new instance of connect
     */
    public Connect() {
        // create a new PCSC context
        ctx = new Context();
    }

    /** get the current Context
     *  @returns current Context
     */
    Context getContext() {
        return ctx;
    }

    /** Get list of available readers
     *  @return list of readers
     */
    public String[] listReaders() {
        if (readerList != null)
            return readerList; // reader list already detected

        try {
            ctx.EstablishContext(PCSC.SCOPE_SYSTEM, null, null);
            readerList = ctx.ListReaders();
        } catch (Exception e) {
            System.err.println("Cannot connect to PCSC service!");
            e.printStackTrace();
            System.exit(1);
        }
        return readerList;
    }

    /** Deselect the currently selected reader
     */
    public void deselectReader() {
        if (card != null)
            removeCard(); // remove Card if available

        reader = null;    // maybe a reader is already selected -> de-select
    }

    /** Search for reader by reader-name - first search for exact match,
     *  then try a substring search. If reader was not found return null.
     *  @param  readerName  name of the reader to search for
     *  @return found reader name
```

```
71          */
72         public String selectReaderByName(String readerName) {
73             String names[] = listReaders();
74 
75             deselectReader(); // maybe a reader is already selected -> de-select
76 
77             // search for exact name
78             for (int i=0; i<names.length; i++)
79                 if (names[i].compareTo(readerName) == 0) {
80                     reader = names[i];
81                     return reader;
82                 }
83 
84             // search for substring name
85             for (int i=0; i<names.length; i++)
86                 if (names[i].indexOf(readerName) != -1) {
87                     reader = names[i];
88                     return reader;
89                 }
90 
91             // not found -> return null
92             return reader;
93         }
94 
95         /** Get reader specified by index number,
96          *  If index is invalid, return null.
97          *  @param  readerNumber index of reader to return
98          *  @return reader name
99          */
100        public String selectReaderByNumber(int readerNumber) {
101            String names[] = listReaders();
102 
103            deselectReader(); // maybe a reader is already selected -> de-select
104 
105            try {
106                reader = listReaders()[readerNumber];
107                return reader;
108            } catch (IndexOutOfBoundsException e) {
109                // invalid index: name not found
110                return null;
111            }
112        }
113 
114        /** remove the Card
115         */
116        public void removeCard() {
117            card = null;
118        }
119 
120        /** Get card of currenly selected reader
121         *  @return card
122         */
123        public Card getCard() {
124            if (card != null) return card;
125 
126            // get state of reader and wait for card
127            State[] rsa = new State[1];
128            rsa[0] = new State(reader);
129            do {
130                try {
131                    // get card status
132                    ctx.GetStatusChange(1000, rsa);
133                } catch(Exception e) {
134                    System.err.println("Context.GetStatusChange() failed!\n"
135                                       +"Error class: " + e.getClass() + "\n"
136                                       +"Error message: " + e.getMessage() + "\n");
```

```
137                     System.exit(1);
138                     return null;
139                 }
140             } while ( (rsa[0].dwEventState & PCSC.STATE_PRESENT)
141                       != PCSC.STATE_PRESENT                        );
142 
143 //            System.out.println("ReaderState of " + reader + ":\n"
144 //                               +rsa[0].toString());
145             try{
146                 // connect to card
147                 card = ctx.Connect(reader, PCSC.SHARE_EXCLUSIVE,
148                                    PCSC.PROTOCOL_T1 | PCSC.PROTOCOL_T0);
149             } catch(Exception e) {
150                 System.err.println("Card.Connect() failed!\n"
151                                    +"Error class: " + e.getClass() + "\n"
152                                    +"Error message: " + e.getMessage() + "\n");
153                 System.exit(1);
154                 return null;
155             }
156             return card;
157         }
158 }
```

A.2.2 Tester/ParseCmd.java

```
1  /*
2   * parseCmd.java
3   *
4   * Created on 1. Mai 2003, 18:45
5   */
6 
7  package Tester;
8 
9  import Tester.cmd.Select;
10 
11 /**
12  *  Parse and execute command
13  *
14  *  @author  Schroettner Robert<rs@ednet.at>
15  *  @version 1.0
16  */
17 public class ParseCmd {
18 
19     private Connect con;
20 
21     /** Creates a new instance of parseCmd
22      */
23     public ParseCmd(Connect _con) {
24         con = _con;
25     }
26 
27     /** Print help test
28      *  @param la remaining arguments
29      */
30     protected boolean cmdHelp(String[] la) {
31         if (la.length > 1) {
32             if (la[1].equalsIgnoreCase("quit"   ))
33                 System.out.println("quit: quit program");
34 
35             if (la[1].equalsIgnoreCase("help"   ))
36                 System.out.println("help: print this help");
37 
38             if (la[1].equalsIgnoreCase("atr"    ))
39                 return helpAtr();
40 
```

```
41              if (la[1].equalsIgnoreCase("SELECT" ))
42                  return helpSelect(la);
43          }
44          System.out.println("Commands: atr help SELECT quit");
45          return true;
46      }
47
48      /** Parse and execute given line
49       *  @param line command line
50       */
51      public boolean parse(String line) {
52          line = line.replaceAll("^\\s*",""); // remove multiple spaces
53          line = line.replaceAll("\\s*$","");
54          line = line.replaceAll("\\s+"," ");
55
56          if (line.compareTo("") == 0) return true; // empty line?
57
58          String[] la = line.split(" "); // split line by ' '
59
60          // parse command and distribute execution
61          if (la[0].equalsIgnoreCase("quit"   )) return false;
62          if (la[0].equalsIgnoreCase("help"   )) return cmdHelp  (la);
63          if (la[0].equalsIgnoreCase("atr"    )) return cmdAtr   (la);
64          if (la[0].equalsIgnoreCase("SELECT" )) return cmdSelect(la);
65
66          System.out.println("unknown command: "+line+" try help");
67          return true;
68      }
69
70      /** Convert a nibble (=4 bit value) to hex character
71       *  @param  nibble
72       *  @return hex value of given nibble
73       */
74      protected char nibble2hex(int nibble) {
75          char list[] = {
76              '0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'
77          };
78          return list[nibble&0xf];
79      }
80
81      /** Convert ATR-String to hex
82       *  @param  atr  ATR value of card
83       *  @return string representation of ATR value
84       */
85      protected String atr2str(byte[] atr) {
86          StringBuffer sb = new StringBuffer();
87
88          for (int i=0; i<atr.length; i++) {
89              sb.append(nibble2hex(atr[i]>>4));
90              sb.append(nibble2hex(atr[i]    ));
91              sb.append(' ');
92          }
93
94          sb.append('\'');
95          for (int i=0; i<atr.length; i++)
96              if (atr[i] >= 0x20 && atr[i] <= 0x7F)
97                  sb.append(atr[i]);
98              else
99                  sb.append(".");
100         sb.append('\'');
101
102         return sb.toString();
103     }
104
105     /** print Help text for ATR-Command
106      */
```

```
107     protected boolean helpAtr() {
108         System.out.println("atr: print card's atr data");
109         return true;
110     }
111 
112     /** execute ATR command
113      *  @param la remaining arguments
114      */
115     protected boolean cmdAtr(String[] la) {
116         System.out.println("Card ATR is "+atr2str(con.getCard().Status().rgbAtr));
117         return true;
118     }
119 
120     /** remove element with highest index from list
121      *  @param  la list of arguments
122      *  @return same list as ls with last entry removed
123      */
124     protected String[] removeLast(String[] la) {
125         if (la == null) return null;
126         String la2[] = new String[la.length-1];
127         for (int i=0; i<la2.length; i++)
128             la2[i] = la[i];
129         return la2;
130     }
131 
132     /** print Help text for SELECT-Command
133      *  @param  la list of arguments
134      */
135     protected boolean helpSelect(String[] la) {
136         if (la.length>2) {
137             if (la[2].equalsIgnoreCase("MF")) {
138                 System.out.print(
139                     "SELECT MF\n"
140                    +" SELECT MF   - selects the MF\n"
141                 );
142                 return true;
143             }
144             if (la[2].equalsIgnoreCase("FID")) {
145                 System.out.print(
146                     "SELECT FID <fid>\n"
147                    +"  fid is a 16-bit hex value\n"
148                    +" SELECT FID 6f00   - selects the MF\n"
149                 );
150                 return true;
151             }
152             if (la[2].equalsIgnoreCase("path")) {
153                 System.out.print(
154                     "SELECT path <fid-path>\n"
155                    +"  fid-path is a list of 16-bit hex values"
156                    +  "separated by ':'\n"
157                    +" SELECT path 4567/3c04   - selects <current df>/4567/3c04\n"
158                 );
159                 return true;
160             }
161             if (la[2].equalsIgnoreCase("MFpath")) {
162                 System.out.print(
163                     "SELECT MFpath <fid-path>\n"
164                    +"  fid-path is a list of 16-bit hex values separated "
165                    +   "by ':'\n"
166                    +" SELECT MFpath 3c04:4567   - selects MF/3c04/4567\n"
167                 );
168                 return true;
169             }
170             if (la[2].equalsIgnoreCase("AID")) {
171                 System.out.print(
172                     "SELECT AID <name>\n"
```

```
173                     +"  name if the name of a application (hex digits)\n"
174                     +" SELECT AID A0305060708   - selects application A0305060708\n"
175                 );
176                 return true;
177             }
178         }
179
180         System.out.print(
181             "SELECT: send SELECT command to card\n"
182            +"  SELECT MF                        select the MF\n"
183            +"  SELECT FID <fid> <rm>          select by fid\n"
184            +"  SELECT path <fid-path> [rm]    select relative path\n"
185            +"  SELECT MFpath <fid-path> [rm]  select mf path\n"
186            +"  SELECT AID <name> [rm]         select application\n"
187            +" rm is an optional return modifer:\n"
188            +"  FCP       - return FCP template\n"
189         );
190         return true;
191     }
192
193     /** execute SELECT command
194      *  @param la remaining arguments
195      */
196     protected boolean cmdSelect(String[] la) {
197         boolean returnNoData = true;
198         // last argument may be 'FCP' for 'return FCP data'
199         // parse and remove this argument from list
200         if (la.length > 1) {
201             if (la[la.length-1].equalsIgnoreCase("FCP")) {
202                 returnNoData = false;
203                 la = removeLast(la);
204             }
205         }
206
207         if (la.length < 2) // no more arguments -> print help
208             return helpSelect(new String[]{"SELECT"});
209
210         if (la[1].equalsIgnoreCase("MF")) {
211             if (la.length!=2) // not enough arguments -> print help
212                 return helpSelect(new String[]{"SELECT", "MF"});
213
214             Select.mf(con, returnNoData);
215             return true;
216         }
217         if (la[1].equalsIgnoreCase("FID")) {
218             if (la.length!=3) // not enough arguments -> print help
219                 return helpSelect(new String[]{"SELECT", "FID"});
220
221             Select.fid(con, la[2], returnNoData);
222             return true;
223         }
224         if (la[1].equalsIgnoreCase("path")) {
225             if (la.length!=3) // not enough arguments -> print help
226                 return helpSelect(new String[]{"SELECT", "path"});
227
228             Select.path(con, la[2], returnNoData);
229             return true;
230         }
231         if (la[1].equalsIgnoreCase("MFpath")) {
232             if (la.length!=3) // not enough arguments -> print help
233                 return helpSelect(new String[]{"SELECT", "MFpath"});
234
235             Select.mfPath(con, la[2], returnNoData);
236             return true;
237         }
238         if (la[1].equalsIgnoreCase("aid")) {
```

```
239             if (la.length!=3) // not enough arguments -> print help
240                 return helpSelect(new String[]{"SELECT", "AID"});
241
242             Select.aid(con, la[2], returnNoData);
243             return true;
244         }
245
246         return helpSelect(new String[]{"SELECT"});
247     }
248 }
```

A.2.3 Tester/main.java

```
1  package Tester;
2
3  import com.linuxnet.jpcsc.*;
4  import java.io.BufferedReader;
5  import java.io.InputStreamReader;
6  import java.io.IOException;
7  import java.util.Hashtable;
8  import java.util.Enumeration;
9
10 /**
11  *  Main class for 3GPP-Tester
12  *
13  *  @author  Schroettner Robert<rs@ednet.at>
14  *  @version 1.0
15  */
16
17 public class main {
18
19     /** Print message, usage text on System.err and exit with code 1
20      *  @param msg Message to print
21      */
22     public static void usage(String msg) {
23         System.err.print(
24             msg+"\n"
25            +"Options:␣--readerName␣<name>␣␣␣select␣reader␣by␣name\n"
26            +"␣␣␣␣␣␣␣␣␣--readerNumber␣<idx>␣␣select␣reader␣by␣number\n"
27            +"␣␣␣␣␣␣␣␣␣--listReaders␣␣␣␣␣␣␣␣␣list␣available␣readers\n"
28            +"␣␣␣␣␣␣␣␣␣--help␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣this␣output\n"
29         );
30         System.exit(1);
31     }
32
33     /** Parses command line options and puts all options beginning
34      *  with '-' as key, all other as key data in hashtable
35      *  @param  args  Command line argument list
36      *  @return Hash filled with command line options
37      */
38     protected static Hashtable checkOpts(String[] args) {
39
40         // allowed command line options
41         final String optList[] = {
42             "--readerName", "--readerNumber", "--listReaders", "--help"
43         };
44
45         Hashtable opts = new Hashtable();
46
47         String key = "";
48         opts.put(key, "");
49
50         for (int i=0; i<args.length; i++) {
51 //            System.out.print("checking param "+i+" \""+args[i]+"\"");
52             if (args[i].matches("-.*")) {
```

```
53                  // argument starts with '-' --> key-Value
54  //                System.out.println(" -> is option");
55                  key = args[i].replaceFirst("^\\s*", "");
56
57                  String val = (String)opts.get(key);
58                  if (val == null) val = "";
59
60                  int j;
61                  for (j=0; j<optList.length; j++)
62                      if (optList[j].compareTo(key) == 0) break;
63                  if (j == optList.length)
64                      usage("invalid option "+key);
65
66                  opts.put(key, val);
67              } else {
68                    // --> data value
69  //                System.out.println(" -> is data");
70
71                  String val = (String)opts.get(key);
72                  if (val.length() == 0) val = args[i]; else val+= " " + args[i];
73                  opts.put(key, val);
74              }
75          }
76
77          if (opts.get("--help") != null) // --help given?
78              usage("");
79
80          // DEBUG: list parsed options
81  //        Enumeration elements = opts.keys();
82  //        while (elements.hasMoreElements()) {
83  //            key = (String)elements.nextElement();
84  //            System.out.println("key '"+key+"' = '"+opts.get(key)+"'");
85  //        }
86          return opts;
87      }
88
89
90      /** Parse opts '--readerName' and '--readerNumber' and return selected
91       *  reader  or exit(1)
92       *  @param  con  PC/SC connection
93       *  @param  opts Hash with commant line options
94       *  @return selected reader
95       */
96      public static String selectReader(Connect con, Hashtable opts) {
97          String readerName    = (String)opts.get("--readerName");
98          opts.remove("--readerName");
99  //        System.out.println("--readerName = '"+readerName+"'");
100
101         String readerNumber  = (String)opts.get("--readerNumber");
102         opts.remove("--readerNumber");
103 //        System.out.println("--readerNumber = '"+readerNumber+"'");
104
105         if (readerName != null) {
106             if (readerNumber != null)
107                 usage("please specify --readerName or --readerNumber"
108                       +" (both given)");
109
110             // select by readerName
111             String reader = con.selectReaderByName(readerName);
112             if (reader == null)
113                 usage("reader named '"+readerName+"' not found");
114
115             return reader;
116
117         } else if (readerNumber != null) {
118             // select by reader number
```

```
119             try {
120                 int number = Integer.parseInt(readerNumber);
121                 String reader = con.selectReaderByNumber(number-1);
122                 if (reader == null)
123                     usage("reader number "+readerNumber+" not found");
124 
125                 return reader;
126             } catch (NumberFormatException e) {
127                 usage("invalid reader number '"+readerNumber+"'");
128             }
129         }
130 
131 //         usage("please specify --readerName or --readerNumber (none given)");
132         String reader = con.selectReaderByNumber(0);
133         if (reader == null)
134             usage("no readers found");
135 
136         return reader;
137     }
138 
139     /** Main
140      */
141     public static void main(String[] args) {
142         // parse and load command line option into hash
143         Hashtable opts = checkOpts(args);
144 
145         Connect con = new Connect();
146 
147         // check for command line option --listReaders
148         //   --> print list of readers
149         if (opts.get("--listReaders") != null) {
150             opts.remove("--listReaders");
151 
152             String[] readerNames = con.listReaders();
153 
154             System.out.println(readerNames.length+" Reader(s) found:");
155             for (int i=0; i<readerNames.length; i++)
156                 System.out.println("  "+(i+1)+": "+readerNames[i]);
157             System.exit(0);
158         }
159 
160         String selectedReader = selectReader(con, opts);
161         // if selected reader is null -> use first reader
162 
163         System.out.println("Please insert a card in reader '"+selectedReader+"'");
164 
165         Card card = con.getCard();
166 
167         System.out.println("Thank you!");
168         System.out.println("Blease select Applet by SELECT AID A000000087");
169 
170         BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
171         ParseCmd pc = new ParseCmd(con);
172 
173         // command loop
174         try {
175             do {
176                 System.out.print("> ");
177             } while (pc.parse(in.readLine()));
178         } catch (IOException e) {
179             System.err.println(e);
180             e.printStackTrace();
181             System.exit(1);
182         }
183         System.out.println("Bye.");
184 //         card.Disconnect();
```

```
185     }
186 }
```

A.2.4 Tester/cmd/Select.java

```
1  /*
2   * select.java
3   *
4   * Created on 1. Mai 2003, 19:59
5   */
6
7  package Tester.cmd;
8
9  import Tester.Connect;
10 import Tester.cmd.Commands.CLA00;
11 import Tester.cmd.Commands.CLA80;
12
13 /**
14  *  execute SELECT command and parse responses
15  *
16  *  @author  Schroettner Robert<rs@ednet.at>
17  *  @version 1.0
18  */
19 public class Select {
20
21     /** get byte for 'RETURN NO DATA' or 'RETURN FCP TEMPLATE' from flag
22      */
23     protected static int ndb(boolean returnNoData) {
24         return (returnNoData ? 0x0C : 0x04);
25     }
26
27     /** get string for 'RETURN NO DATA' or 'RETURN FCP TEMPLATE' from flag
28      */
29     protected static String nds(boolean returnNoData) {
30         return (returnNoData ? "" : "␣return␣FCP");
31     }
32
33     /** convert file descriptor to human readable string
34      */
35     protected static String fd2str(int fd) {
36         fd = fd & 0xff;
37         if ((fd & 0x80) != 0) return "RFU␣(1xxxxxxx)";
38
39         StringBuffer sb = new StringBuffer();
40         if ((fd & 0x40) != 0)
41             sb.append("Shareable␣");
42         switch (fd & 0x38) {
43             case 0x00:
44                 sb.append("Working_EF␣"); break;
45             case 0x08:
46                 sb.append("Internal_EF␣"); break;
47             case 0x038:
48                 sb.append("DF␣"          ); break;
49             default:
50                 sb.append("File_type_RFU("+Integer.toHexString(fd & 0x38)+")␣");
51         }
52         switch (fd & 0x07) {
53             case 0x00:
54                 sb.append("No_structure_information"); break;
55             case 0x01:
56                 sb.append("Transparent"             ); break;
57             case 0x02:
58                 sb.append("Linear_fixed"            ); break;
59             case 0x06:
60                 sb.append("Cyclic"                  ); break;
```

```
61              default:
62                  sb.append("Structure␣RFU"           ); break;
63          }
64          return sb.toString();
65      }
66
67      /** convert flle name to human readable string (=hex string)
68       */
69      protected static String name2str(byte ret[], int pos, int len) {
70          StringBuffer sb = new StringBuffer();
71          for (int i=0; i<len; i++)
72              sb.append(Integer.toHexString(ret[i])+"␣");
73          return sb.toString();
74      }
75
76      /** convert propietary info to human readable string
77       */
78      protected static String prop2str(byte ret[], int pos, int len) {
79          StringBuffer sb = new StringBuffer();
80          int le;
81          for (int i=pos; i<pos+len; i++) {
82              switch (ret[i] & 0xff) {
83                  case 0x80:
84                      le = ret[++i]&0xff;
85                      sb.append("UICC␣characteristics"
86                                +"␣len␣"+le+"␣");
87                      if ((ret[i] & 0x82) != 0)
88                          sb.append("RFU(X-----X-)");
89                      else {
90                          if ((ret[i]&0x10)!= 0) sb.append("Supply␣class␣A,␣");
91                          if ((ret[i]&0x20)!= 0) sb.append("Supply␣class␣B,␣");
92                          if ((ret[i]&0x40)!= 0) sb.append("Supply␣class␣C,␣");
93                          if ((ret[i]&0x01)!= 0) {
94                              sb.append("Clock␣stop␣not␣allowed␣");
95                              switch (ret[i]&0x0C) {
96                                  case 0x00: sb.append("(no␣preferred␣level)");
97                                             break;
98                                  case 0x04: sb.append("(high␣level␣preferred)");
99                                             break;
100                                 case 0x08: sb.append("(lov␣level␣preferred)");
101                                            break;
102                                 case 0xC0: sb.append("(RFU)");
103                                            break;
104                             }
105                         } else {
106                             sb.append("Clock␣stop␣allowed␣");
107                             switch (ret[i]&0x0C) {
108                                 case 0x00: sb.append("(never)");
109                                            break;
110                                 case 0x04: sb.append("(unless␣at␣high␣level)");
111                                            break;
112                                 case 0x08: sb.append("(unless␣at␣low␣level)");
113                                            break;
114                                 case 0xC0: sb.append("(RFU)");
115                                            break;
116                             }
117                         }
118                     }
119                     i+= le;
120                     break;
121
122                 case 0x81:
123                     le = ret[++i]&0xff;
124                     sb.append("Power␣consumption"
125                               +"␣len␣"+le+"␣supply␣class␣"
126                               +Integer.toHexString(ret[i]&0xff)
```

```
127                             +" consumption "+Integer.toHexString(ret[i]&0xff)
128                             +" ref frequency "+Integer.toHexString(ret[i]&0xff));
129                     i+= le;
130                     break;
131
132                 case 0x82:
133                     le = ret[++i]&0xff;
134                     sb.append("Minimum clock frequency"
135                             +" len "+le+" "+Integer.toHexString(ret[i]&0xff));
136                     i+= le;
137                     break;
138                 case 0x83:
139                     le = ret[++i]&0xff;
140                     sb.append("Available memory "
141                             +" len "+le+" "+Integer.toHexString(ret[i]&0xff));
142                     i+= le;
143                     break;
144
145                 default:
146                     sb.append(Integer.toHexString(ret[i]&0xff)+" ");
147             }
148         }
149         return sb.toString();
150     }
151
152     /** convert live cycle to human readable string
153      */
154     protected static String live2str(int live) {
155         switch (live) {
156             case 0x00:
157                 return "No information given";
158             case 0x01:
159                 return "Creatin state";
160             case 0x03:
161                 return "Initialisation state";
162             case 0x05:
163             case 0x07:
164                 return "Operational state - activated";
165             case 0x04:
166             case 0x06:
167                 return "Operational state - deactivated";
168         }
169         if ((live & 0x0C) == 0x0C)
170             return "Termination state";
171         return "RFU ("+Integer.toHexString(live)+")";
172     }
173
174     /** parse and print respose of a select command
175      */
176     protected static void parseResponse(byte ret[]) {
177         int pos = 0;
178         int len, size, fd, cod, recLen, recNum, fid, sfi, live;
179         byte name[];
180
181         while (pos < ret.length-2) switch (ret[pos++]) {
182             case (byte)0x80:
183                 len  = ret[pos++]&0xff;
184                 size = (ret[pos+1]&0xff) + 255*(ret[pos]&0xff);
185                 System.out.println("  TLV Tag 80 len "+len
186                                    +" File size: "+size);
187                 pos += len;
188                 break;
189
190             case (byte)0x81:
191                 len  = ret[pos++]&0xff;
192                 size = (ret[pos]&0xff)<<8 | (ret[pos+1]&0xff);
```

```
193                 System.out.println("␣␣TLV␣Tag␣81␣len␣"+len
194                                    +"␣Total␣file␣size:␣"+size);
195                 pos += len;
196                 break;
197
198             case (byte)0x82:
199                 len  = ret[pos++]&0xff;
200                 fd   = ret[pos  ]&0xff;
201                 cod  = ret[pos+1]&0xff;
202                 if (len>2) {
203                     recLen = (ret[pos+3]&0xff) + 255*(ret[pos+2]&0xff);
204                     recNum =  ret[pos+4]&0xff;
205                     System.out.println("␣␣TLV␣Tag␣82␣len␣"+len
206                                        +"␣File␣descriptor:␣'"+fd2str(fd)
207                                        +"',␣Coding␣"+Integer.toHexString(cod)
208                                        +",␣"+recNum+"␣Records␣of␣"
209                                        +recLen+"␣bytes");
210                 } else
211                     System.out.println("␣␣TLV␣Tag␣82␣len␣"+len
212                                        +"␣File␣descriptor␣'"+fd2str(fd)
213                                        +"',␣Coding␣"+Integer.toHexString(cod));
214                 pos += len;
215                 break;
216
217             case (byte)0x83:
218                 len  = ret[pos++]&0xff;
219                 fid  = (ret[pos]&0xff)<<8 | (ret[pos+1]&0xff);
220                 System.out.println("␣␣TLV␣Tag␣83␣len␣"+len
221                                    +"␣File␣identifier:␣"+Integer.toHexString(fid));
222                 pos += len;
223                 break;
224
225             case (byte)0x84:
226                 len  = ret[pos++]&0xff;
227                 System.out.println("␣␣TLV␣Tag␣83␣len␣"+len
228                                    +"␣File␣name:␣"+name2str(ret, pos, len));
229                 pos += len;
230                 break;
231
232             case (byte)0x88:
233                 len  = ret[pos++]&0xff;
234                 sfi  = ret[pos  ]&0xff;
235                 System.out.println("␣␣TLV␣Tag␣88␣len␣"+len
236                                    +"␣Short␣file␣identifier:␣"
237                                    +Integer.toHexString(sfi));
238                 pos += len;
239                 break;
240
241             case (byte)0x8A:
242                 len  = ret[pos++]&0xff;
243                 live = ret[pos  ]&0xff;
244                 System.out.println("␣␣TLV␣Tag␣8A␣len␣"+len
245                                    +"␣Live␣cycle:␣"+live2str(live));
246                 pos += len;
247                 break;
248
249             case (byte)0xA5:
250                 len  = ret[pos++]&0xff;
251                 System.out.println("␣␣TLV␣Tag␣A5␣len␣"+len
252                                    +"␣Propietary␣info:␣"+prop2str(ret, pos, len));
253                 pos += len;
254                 break;
255
256             default:
257                 int tag = ret[pos-1]&0xff;
258                     len = ret[pos  ]&0xff;
```

```
259                 System.out.println("␣␣TLV␣Tag␣"+Integer.toHexString(tag)
260                                    +"␣len␣"+len);
261                 pos += len+1;
262                 break;
263             }
264         }
265
266         /** execute SELECT MF
267          */
268         public static void mf(Tester.Connect con, boolean returnNoData) {
269             if (!returnNoData)
270                 System.out.println("WARNING:␣ignoring␣'FCP'");
271             System.out.print("SELECT␣MF:␣");
272             byte[] r=Send.apdu(con, CLA00.CLA, CLA00.SELECT, 0, 0x0c);
273             parseResponse(r);
274         }
275
276         /** execute SELECT BY FID
277          */
278         public static void fid(Tester.Connect con, String fid,
279                                boolean returnNoData) {
280             System.out.print("SELECT␣FID␣"+fid+nds(returnNoData)+":␣");
281             try {
282                 int ifid = Integer.parseInt(fid, 16);
283
284                 byte[] r=Send.apdu(con, CLA00.CLA, CLA00.SELECT, 0,
285                                    ndb(returnNoData),
286                                    new byte[]{ (byte)(ifid>>8), (byte)ifid });
287                 parseResponse(r);
288             } catch (NumberFormatException e) {
289                 System.out.println("'"+fid+"'␣is␣not␣a␣valid␣fid␣"
290                                    +"-␣hex␣value␣expected");
291             }
292         }
293
294         /** execute SELECT BY RELATIVE PATH
295          */
296         public static void path(Tester.Connect con, String path,
297                                 boolean returnNoData) {
298             System.out.println("SELECT␣path␣"+path+nds(returnNoData)+":␣");
299             byte[] r=Send.apdu(con, CLA00.CLA, CLA00.SELECT, 8, ndb(returnNoData));
300             parseResponse(r);
301         }
302
303         /** execute SELECT BY MF PATH
304          */
305         public static void mfPath(Tester.Connect con, String path,
306                                   boolean returnNoData) {
307             System.out.println("SELECT␣MFpath␣"+path+nds(returnNoData)+":␣");
308             byte[] r=Send.apdu(con, CLA00.CLA, CLA00.SELECT, 9, ndb(returnNoData));
309             parseResponse(r);
310         }
311
312         /** execute SELECT BY AID
313          */
314         public static void aid(Tester.Connect con, String aid,
315                                boolean returnNoData) {
316             System.out.print("SELECT␣AID␣"+aid+nds(returnNoData)+":␣");
317             byte[] baid= new byte[(int)(aid.length()/2+0.5)];
318             try {
319                 for (int i=0; i<baid.length; i++) {
320                     String sub = aid.substring((int)(i*2), (int)(i*2+2));
321 //                    System.out.print(" conv "+sub);
322                     baid[i] = (byte)Integer.parseInt(sub, 16);
323                 }
324
```

```
325             byte[] r=Send.apdu(con, CLA00.CLA, CLA00.SELECT, 4,
326                                ndb(returnNoData), baid);
327             parseResponse(r);
328         } catch (NumberFormatException e) {
329             System.out.println("'"+aid+"' is not a valid name "
330                                +"- only hex digits allowed");
331         }
332     }
333 }
```

A.2.5 Tester/cmd/Send.java

```
1  /*
2   * Send.java
3   *
4   * Created on 1. Mai 2003, 20:31
5   */
6
7  package Tester.cmd;
8
9  import com.linuxnet.jpcsc.Apdu;
10 import com.linuxnet.jpcsc.Apdu.Format;
11 import com.linuxnet.jpcsc.PCSCException;
12
13 import Tester.Connect;
14
15 /**
16  *  Collection of some static functions for sending data to card
17  *
18  *  @author  Schroettner Robert<rs@ednet.at>
19  *  @version 1.0
20  */
21 public class Send {
22
23     //static Apdu.Format af = new Apdu.Format(Apdu.HEX_SPACE_HEX_FORMAT,
24     //                                        true, true);
25     static Apdu.Format af = new Apdu.Format(Apdu.HEX_SPACE_HEX_FORMAT,
26                                             true, false);
27     /** send apdu to card
28      */
29     public static byte[] apdu(Tester.Connect con, Apdu apdu) {
30         System.out.println("APDU len " +apdu.toString(af));
31         try {
32             byte[] response = con.getCard().Transmit(apdu);
33             System.out.println(" RECEIVED "+Apdu.ba2s(response, af));
34             System.out.println(" Status: "+Status.toString(response));
35             return response;
36         } catch(PCSCException pe) {
37             System.err.println("Card.Transmit(): failed!\n"
38                                +"PCSC Error Message: " + pe.getMessage());
39             System.exit(1);
40         } catch(Exception e) {
41             System.err.println("Card.Transmit() failed!\n"
42                                +"Error class: " + e.getClass() + "\n"
43                                +"Error message: " + e.getMessage());
44             System.exit(1);
45         }
46         return new byte[0];
47     }
48
49     /** build and send apdu to card
50      */
51     public static byte[] apdu(Tester.Connect con,
52                               byte cla, byte ins, int p1, int p2) {
53         return apdu(con, new Apdu(cla, ins, p1, p2, 0, new byte[0], 0, 0));
```

```
54      }
55
56      /** build and send apdu to card
57       */
58      public static byte[] apdu(Tester.Connect con,
59                                byte cla, byte ins, int p1, int p2, byte data[]) {
60          return apdu(con, new Apdu(cla, ins, p1, p2, data.length, data, 0, 0));
61      }
62  }
```

A.2.6 Tester/cmd/Status.java

```
1   /*
2    * Status.java
3    *
4    * Created on 22. Februar 2003, 00:26
5    */
6
7   package Tester.cmd;
8
9   //import javacard.framework.Util;
10  //import javacard.framework.ISOException;
11  //import javacard.framework.Shareable;
12
13  /**
14   *  Valid values for return codes (StatusWord).
15   *
16   *  @see <a href=http://www.etsi.org>ETSI TS 121 221 V5.1.0, Section 10.2.1</a>
17   *  @author  Schroettner Robert<rs@ednet.at>
18   *  @version 1.0
19   */
20  public class Status { //implements Shareable {
21
22      private static class ent {
23          public String name;
24          public short val;
25          public short mask;
26
27          public ent(String _name, int _val) {
28              name = _name;
29              val  = (short)_val;
30              mask = (short)0xffff;
31          }
32          public ent(String _name, int _val, int _mask) {
33              name = _name;
34              val  = (short)_val;
35              mask = (short)_mask;
36          }
37      };
38
39      private static ent codes[] = {
40      ////////////////////////////////////////////////////////////////////////
41      //
42      // Normal ending
43      //
44
45      /** Normal ending of the command.
46       *  <br>
47       *  Value: 0x9000<br>
48       *  Category: Normal Processing
49       */
50          new ent("OK_NO_ERROR", 0x9000),
51
52      /** Normal ending of the command, with extra information from the procative
53       *  UICC containing a command for the terminal.
```

```
54          *  Length len bytes of the response data.<br>
55          *  Value: 0x90xx<br>
56          *  Category: Normal Processing<br>
57          *
58          *  @param len  len of the response data
59          */
60             new ent("OK_NO_ERROR_INFO", 0x9000, 0xFF00),
61
62         ////////////////////////////////////////////////////////////////////////
63         //
64         // Postponed processing
65         //
66
67         /** SIM Application Toolkit is busy - command cannot be executed at present,
68          *  further normal commands are allowed.
69          *  <br>
70          *  Value: 0x9300<br>
71          *  Category: Postponed processing
72          */
73             new ent("INFO_TOOKIT_BUSY", 0x9300),
74
75         ////////////////////////////////////////////////////////////////////////
76         //
77         // Warnings
78         //
79
80         /** No Information given, state of non valatile memory unchanged.
81          *  <br>
82          *  Value: 0x6200<br>
83          *  Category: Warnings
84          */
85             new ent("WARN_NO_INFO", 0x6200),
86
87         /** Part of the returned data may be corrupted.
88          *  <br>
89          *  Value: 0x6281<br>
90          *  Category: Warnings
91          */
92             new ent("WARN_RETURNED_DATA_CORRUPTED", (short)0x6281),
93
94         /** End of file/record reached fefore reading Le bytes.
95          *  <br>
96          *  Value: 0x6282<br>
97          *  Category: Warnings
98          */
99             new ent("WARN_EOF_REACHED", (short)0x6282),
100
101        /** Selected file invalidated.
102         *  <br>
103         *  Value: 0x6283<br>
104         *  Category: Warnings
105         */
106            new ent("WARN_SELECTED_FILE_INV", (short)0x6283),
107
108        /** Command successful but after using an internal update retry routine
109         *  cnt times.
110         *  <br>
111         *  Value: 0x63Cx<br>
112         *  Category: Warnings
113         */
114        /** Verification failed, cnt retries remaining.
115         *  <br>
116         *  Value: 0x63Cx<br>
117         *  Category: Warnings
118         */
119            new ent("WARN_SUCCESSFUL_AFTER/WARN_VERIFY_FAILED", (short)0x63C0, 0xFFF0),
```

```
120
121
122     ////////////////////////////////////////////////////////////////////////
123     //
124     // Execution errors
125     //
126
127     /** No information given, state of non-valatile memory unchanged.
128      *  <br>
129      *  Value: 0x6400<br>
130      *  Category: Execution errors
131      */
132         new ent("ERR_NO_INFO", (short)0x6400),
133
134     /** No information given, state of non-valatile memory changed.
135      *  <br>
136      *  Value: 0x6500<br>
137      *  Category: Execution errors
138      */
139         new ent("ERR_NO_INFO_MEM_CHANGED", (short)0x6500),
140
141     /** Memory problem.
142      *  <br>
143      *  Value: 0x6581<br>
144      *  Category: Execution errors
145      */
146         new ent("ERR_MEMOY_PROBLEM", (short)0x6581),
147
148     ////////////////////////////////////////////////////////////////////////
149     //
150     // Checking errors
151     //
152
153     /** Wrong length.
154      *  <br>
155      *  Value: 0x6700<br>
156      *  Category: Checking errors
157      */
158         new ent("CHK_WRONG_LENGTH", (short)0x6700),
159
160     /** The interpretation of this status word is command dependent.
161      *  <br>
162      *  Value: 0x67xx<br>
163      *  Category: Checking errors
164      */
165         new ent("CHK_FAILED_67", (short)0x6700, 0xff00),
166
167     /** Wrong parameter(s) P1-P2.
168      *  <br>
169      *  Value: 0x6B00<br>
170      *  Category: Checking errors
171      */
172         new ent("CHK_WRONG_PARAMETER_P1P2", (short)0x6B00),
173
174     /** Instruction code not supported or invalid.
175      *  <br>
176      *  Value: 0x6D00<br>
177      *  Category: Checking errors
178      */
179         new ent("CHK_INS_NOT_SUPPORTED", (short)0x6D00),
180
181     /** Class not supported.
182      *  <br>
183      *  Value: 0x6E00<br>
184      *  Category: Checking errors
185      */
```

```
186             new ent("CHK_CLA_NOT_SUPPORTED", (short)0x6E00),
187
188         /** Technical problem, no precise diagnostics.
189          *  <br>
190          *  Value: 0x6F00<br>
191          *  Category: Checking errors
192          */
193             new ent("CHK_TECHNICAL_PROBLEM", (short)0x6F00),
194
195         /** The interpretation of this status word is command dependent.
196          *  <br>
197          *  Value: 0x6Fxx<br>
198          *  Category: Checking errors
199          */
200             new ent("CHK_FAILED_6F", (short)0x6F00, 0xff00),
201
202         ////////////////////////////////////////////////////////////////////////
203         //
204         // Functions CLA not supported
205         //
206
207         /** No information given.
208          *  <br>
209          *  Value: 0x6800<br>
210          *  Category: Functions CLA not supported
211          */
212             new ent("CLA_NO_INFO", (short)0x6800),
213
214         /** Logical channel not supported.
215          *  <br>
216          *  Value: 0x6881<br>
217          *  Category: Functions CLA not supported
218          */
219             new ent("CLA_NO_LOGICAL_CHANNEL", (short)0x6881),
220
221         /** Secure messaging not supported.
222          *  <br>
223          *  Value: 0x6882<br>
224          *  Category: Functions CLA not supported
225          */
226             new ent("CLA_NO_SECURE_MESSAGING", (short)0x6882),
227
228         ////////////////////////////////////////////////////////////////////////
229         //
230         // Command not allowed
231         //
232
233         /** No information given.
234          *  <br>
235          *  Value: 0x6900<br>
236          *  Category: Command not allowed
237          */
238             new ent("CMD_NO_INFO", (short)0x6900),
239
240         /** Command incompatible with file structure.
241          *  <br>
242          *  Value: 0x6981<br>
243          *  Category: Command not allowed
244          */
245             new ent("CMD_INCOMPATIBLE", (short)0x6981),
246
247         /** Security status not satisfied.
248          *  <br>
249          *  Value: 0x6982<br>
250          *  Category: Command not allowed
251          */
```

```
252             new ent("CMD_SECURITY_NOT_SATISFIED", (short)0x6982),
253
254         /** Authentication/PIN method blocked.
255          *  <br>
256          *  Value: 0x6983<br>
257          *  Category: Command not allowed
258          */
259             new ent("CMD_PIN_BLOCKED", (short)0x6983),
260
261         /** Referenced data invalidated.
262          *  <br>
263          *  Value: 0x6984<br>
264          *  Category: Command not allowed
265          */
266             new ent("CMD_DATA_INVALID", (short)0x6984),
267
268         /** Conditions of used not satisfied.
269          *  <br>
270          *  Value: 0x6985<br>
271          *  Category: Command not allowed
272          */
273             new ent("CMD_CONDITIONS_NOT_SATISFIED", (short)0x6985),
274
275         /** Command not allowed (no EF selected).
276          *  <br>
277          *  Value: 0x6986<br>
278          *  Category: Command not allowed
279          */
280             new ent("CMD_NOT_ALLOWED", (short)0x6986),
281
282         ////////////////////////////////////////////////////////////////////////
283         //
284         // Wrong parameters
285         //
286
287         /** Incorrect parameters in the data field.
288          *  <br>
289          *  Value: 0x6A80<br>
290          *  Category: Wrong parameters
291          */
292             new ent("PARA_INCORRECT_DATA", (short)0x6A80),
293
294         /** Function not supported.
295          *  <br>
296          *  Value: 0x6A81<br>
297          *  Category: Wrong parameters
298          */
299             new ent("PARA_FUNCTION_NOT_SUPPORTED", (short)0x6A81),
300
301         /** File not found.
302          *  <br>
303          *  Value: 0x6A82<br>
304          *  Category: Wrong parameters
305          */
306             new ent("PARA_FILE_NOT_FOUND", (short)0x6A82),
307
308         /** Record not found.
309          *  <br>
310          *  Value: 0x6A83<br>
311          *  Category: Wrong parameters
312          */
313             new ent("PARA_RECORD_NOT_FOUND", (short)0x6A83),
314
315         /** Incorrect parameter(s) P1-P2.
316          *  <br>
317          *  Value: 0x6A86<br>
```

```
318         *  Category: Wrong parameters
319         */
320            new ent("PARA_INCORRECT_P1P2", (short)0x6A86),
321
322        /** Lc inconsistent with P1-P2.
323         *  <br>
324         *  Value: 0x6A87<br>
325         *  Category: Wrong parameters
326         */
327            new ent("PARA_INCORRECT_LC", (short)0x6A87),
328
329        /** Referenced data not found.
330         *  <br>
331         *  Value: 0x6A88<br>
332         *  Category: Wrong parameters
333         */
334            new ent("PARA_DATA_NOT_FOUND", (short)0x6A88),
335
336        ////////////////////////////////////////////////////////////////////////
337        //
338        // Application errors
339        //
340
341        /** INCREASE cannot be performed, max value reached.
342         *  <br>
343         *  Value: 0x9850<br>
344         *  Category: Application errors
345         */
346            new ent("APP_NO_INCREASE", (short)0x9850),
347
348        /** Authentication error, application specific.
349         *  <br>
350         *  Value: 0x9862<br>
351         *  Category: Application errors
352         */
353            new ent("APP_AUTHENTICATION_ERROR", (short)0x9862)
354        };
355
356        private static String short2hex(short val) {
357            String s = Integer.toHexString(val&0xffff);
358            StringBuffer sb = new StringBuffer();
359            for (int i=4; i>s.length(); i--) sb.append('0');
360            sb.append(s);
361            return sb.toString();
362        }
363
364        /** get last 2 bytes of array and parse status */
365        public static String toString(byte[] d) {
366            if (d == null) return "No status";
367            if (d.length < 2) return "Too less data for status";
368            short status  = d[d.length-2];
369                  status  = (short)(status << 8);
370                  status |= (short)(0xff & d[d.length-1]);
371
372    //        System.out.println("status="+status);
373
374            for (int i=0; i<codes.length; i++)
375                if (codes[i].val == (status & codes[i].mask))
376                    if (codes[i].mask == (short)0xffff)
377                        return short2hex(status)+" "+codes[i].name;
378                    else
379                        return short2hex(status)+" "+codes[i].name+
380                               " (value="+(status & codes[i].mask)+")";
381
382            return short2hex(status)+" Unknown code";
383        }
```

```
384 }
```

A.2.7 Tester/APDUTool.java

```
1  package Tester;
2
3  import java.awt.*;
4  import java.awt.event.*;
5  import javax.swing.*;
6  import javax.swing.event.*;
7  import javax.swing.border.*;
8  import javax.swing.text.*;
9
10 import com.linuxnet.jpcsc.*;
11
12 public class APDUTool extends JFrame{
13     private static Apdu.Format format = new Apdu.Format(Apdu.HEX_COLON_HEX_FORMAT,
14                                                         true, false);
15     private static String defaultAID = "A0000000030000";
16     private static String defaultAPDU = "80CA9F7F00";
17
18     public static void main(String[] args){
19         final APDUTool t = new APDUTool();
20         t.addWindowListener(new WindowAdapter() {
21             public void windowClosing(WindowEvent e) {
22                 System.exit(0);
23             }
24         });
25         t.pack();
26         t.show();
27     }
28
29     private static String[] labelsText = {
30         "Reader:␣␣␣␣␣␣␣␣",
31         "Applet␣AID:␣␣␣",
32         "APDU:␣␣␣␣␣␣␣␣␣␣",
33     };
34
35     private static String[] buttonsText = {
36         "Connect␣...",
37         "Select␣...␣",
38         "Send␣...␣␣␣",
39     };
40
41     private Context ctx;
42     private Card card;
43     private String[] readerNames;
44     private JComboBox readerBox;
45     private JButton readerConnector;
46     private NumberField appletField;
47     private JButton appletConnector;
48     private NumberField apduField;
49     private JButton apduSender;
50     private JTextArea textArea;
51
52     private APDUTool(){
53         JPanel rootp = new JPanel();
54         rootp.setLayout(new BorderLayout());
55         super.getContentPane().add(rootp);
56
57         ctx = new Context();
58         try{
59             ctx.EstablishContext(PCSC.SCOPE_SYSTEM, null, null);
60             readerNames = ctx.ListReaders();
61         }catch(Exception e){
```

```
62              JOptionPane.showMessageDialog(this, "Cannot␣connect␣to␣PCSC␣service!",
                    "Error",
63                                              JOptionPane.ERROR_MESSAGE);
64              System.exit(0);
65          }
66
67          if (readerNames.length == 0){
68              JOptionPane.showMessageDialog(this, "No␣readers␣available!", "Error",
69                                            JOptionPane.ERROR_MESSAGE);
70              System.exit(0);
71          }
72
73          {
74              JPanel inputp = new JPanel();
75              inputp.setLayout(new BoxLayout(inputp, BoxLayout.Y_AXIS));
76              rootp.add(inputp, BorderLayout.NORTH);
77              {
78                  // the reader panel
79                  JPanel p = new JPanel();
80                  p.setLayout(new BoxLayout(p, BoxLayout.X_AXIS));
81                  p.setBorder(BorderFactory.createTitledBorder("␣Reader␣Selector␣"));
82                  inputp.add(p);
83                  JLabel l = new JLabel(labelsText[0]);
84                  p.add(l);
85                  readerBox = new JComboBox(readerNames);
86                  p.add(readerBox);
87                  readerConnector = new JButton(buttonsText[0]);
88                  p.add(readerConnector);
89              }
90              {
91                  // the aid panel
92                  JPanel p = new JPanel();
93                  p.setLayout(new BoxLayout(p, BoxLayout.X_AXIS));
94                  p.setBorder(BorderFactory.createTitledBorder("␣Applet␣Selector␣"));
95                  inputp.add(p);
96                  JLabel l = new JLabel(labelsText[1]);
97                  p.add(l);
98                  appletField = new NumberField(defaultAID, 20, 32, 16);
99                  p.add(appletField);
100                 appletConnector = new JButton(buttonsText[1]);
101                 p.add(appletConnector);
102             }
103             {
104                 // the apdu panel
105                 JPanel p = new JPanel();
106                 p.setLayout(new BoxLayout(p, BoxLayout.X_AXIS));
107                 p.setBorder(BorderFactory.createTitledBorder("␣Apdu␣Sender␣"));
108                 inputp.add(p);
109                 JLabel l = new JLabel(labelsText[2]);
110                 p.add(l);
111                 apduField = new NumberField(defaultAPDU, 20, 256, 16);
112                 p.add(apduField);
113                 apduSender = new JButton(buttonsText[2]);
114                 p.add(apduSender);
115             }
116         }
117
118         {
119             // the messages panel
120             JPanel p = new JPanel();
121             p.setLayout(new BoxLayout(p, BoxLayout.X_AXIS));
122             p.setBorder(BorderFactory.createTitledBorder("␣Output␣"));
123             rootp.add(p, BorderLayout.CENTER);
124             textArea = new JTextArea(20, 80);
125             JScrollPane sp = new JScrollPane(textArea);
126             p.add(sp);
```

```
127         }
128
129         {
130             // the exit panel
131             JPanel p = new JPanel();
132             p.setLayout(new BoxLayout(p, BoxLayout.X_AXIS));
133             p.setBorder(BorderFactory.createEmptyBorder(5, 0, 5, 0));
134             rootp.add(p, BorderLayout.SOUTH);
135             JButton b = new JButton("Quit");
136             p.add(b);
137             b.addActionListener(new ActionListener() {
138                 public void actionPerformed(ActionEvent ev) {
139                     System.exit(0);
140                 }
141             });
142         }
143
144         // reader connector: connect to reader and wait for card insertion
145         readerConnector.addActionListener(new ActionListener() {
146             public void actionPerformed(ActionEvent ev) {
147                 if (card != null){
148                     // try to disconnect any previous card connection
149                     try{
150                         card.Disconnect(PCSC.LEAVE_CARD);
151                     }catch(Exception e){}
152                 }
153                 // check reader until card is inserted
154                 String reader = (String) readerBox.getSelectedItem();
155                 JOptionPane.showMessageDialog(APDUTool.this,
156                                               "Trying␣to␣connect␣to␣reader␣"
157                                             + reader + ".\n"
158                                             +"Do␣not␣forget␣to␣insert␣a␣card!",
159                                              "Connecting␣...",
160                                              JOptionPane.INFORMATION_MESSAGE);
161                 State[] rsa = new State[1];
162                 rsa[0] = new State(reader);
163                 do{
164                     try{
165                         ctx.GetStatusChange(1000, rsa);
166                     }catch(Exception e){
167                         textArea.append("Context.GetStatusChange()␣failed!\n");
168                         textArea.append("Error␣class:␣" + e.getClass() + "\n");
169                         textArea.append("Error␣message:␣" + e.getMessage() + "\n");
170                         return;
171                     }
172                 }while((rsa[0].dwEventState & PCSC.STATE_PRESENT) != PCSC.
                        STATE_PRESENT);
173                 textArea.append("ReaderState␣of␣" + reader + ":\n");
174                 textArea.append(rsa[0].toString());
175                 try{
176                     // connect to card
177                     card = ctx.Connect(reader, PCSC.SHARE_EXCLUSIVE, PCSC.
                            PROTOCOL_T1 | PCSC.PROTOCOL_T0);
178                 }catch(Exception e){
179                     textArea.append("Card.Connect()␣failed!\n");
180                     textArea.append("Error␣class:␣" + e.getClass() + "\n");
181                     textArea.append("Error␣message:␣" + e.getMessage() + "\n");
182                     return;
183                 }
184             }
185         });
186
187         // applet connector: try to select applet with given aid
188         appletConnector.addActionListener(new ActionListener() {
189             public void actionPerformed(ActionEvent ev) {
190                 if (card == null){
```

```
191                     JOptionPane.showMessageDialog(APDUTool.this, "No active reader
                            connection!", "Error", JOptionPane.ERROR_MESSAGE);
192                     return;
193                 }
194                 // construct select command
195                 Apdu apdu = new Apdu(256);
196                 try{
197                     String s = appletField.getText();
198                     if (s.length() == 0) throw new RuntimeException("no AID given")
                            ;
199                     if ((s.length() % 2) != 0) throw new RuntimeException("odd
                            length of" + s);
200                     apdu.set(0x0, 0xA4, 0x4, 0x0);
201                     apdu.addString(s);
202                 }catch(Exception e){
203                     JOptionPane.showMessageDialog(APDUTool.this, "Invalid input: "
                             + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
204                     return;
205                 }
206                 // send the select command
207                 textArea.append("Selecting applet " + apdu + "\n");
208                 textArea.append("Sending " + apdu + "\n");
209                 try{
210                     byte[] response = card.Transmit(apdu);
211                     textArea.append("Received " + Apdu.ba2s(response, format) + "\n
                            ");
212                 }catch(PCSCException pe){
213                     textArea.append("Card.Transmit(): failed!\n");
214                     textArea.append("PCSC Error Message: " + pe.getMessage() + "\n"
                            );
215                     return;
216                 }catch(Exception e){
217                     textArea.append("Card.Transmit() failed!\n");
218                     textArea.append("Error class: " + e.getClass() + "\n");
219                     textArea.append("Error message: " + e.getMessage() + "\n");
220                     return;
221                 }
222             }
223         });
224 
225         // take edited apdu and send it to card
226         apduSender.addActionListener(new ActionListener() {
227             public void actionPerformed(ActionEvent ev) {
228                 if (card == null){
229                     JOptionPane.showMessageDialog(APDUTool.this, "No active reader
                            connection!", "Error", JOptionPane.ERROR_MESSAGE);
230                     return;
231                 }
232                 Apdu apdu = new Apdu(256);
233                 try{
234                     String s = apduField.getText();
235                     if (s.length() == 0) throw new RuntimeException("no APDU given"
                            );
236                     if ((s.length() % 2) != 0) throw new RuntimeException("odd
                            length of" + s);
237                     apdu.set(s);
238                 }catch(Exception e){
239                     JOptionPane.showMessageDialog(APDUTool.this, "Invalid input: "
                             + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
240                     return;
241                 }
242                 // send the apdu
243                 textArea.append("Sending " + apdu + "\n");
244                 try{
245                     byte[] response = card.Transmit(apdu);
246                     textArea.append("Received " + Apdu.ba2s(response, format) + "\n
```

```
                        ");
247                 }catch(PCSCException pe){
248                     textArea.append("Card.Transmit():␣" + pe.getMessage() + "\n");
249                     return;
250                 }catch(Exception e){
251                     textArea.append("Card.Transmit()␣failed!\n");
252                     textArea.append("Error␣class:␣" + e.getClass() + "\n");
253                     textArea.append("Error␣message:␣" + e.getMessage() + "\n");
254                     return;
255                 }
256             }
257         });
258     }
259 }
260
261
262 class NumberField extends JTextField{
263     public NumberField(){
264         super();
265     }
266
267     public NumberField(int cols, int maxCnt){
268         super(cols);
269         ((NumberDocument) getDocument()).maxCnt = maxCnt;
270     }
271
272     public NumberField(String text, int cols, int maxCnt){
273         super(text, cols);
274         ((NumberDocument) getDocument()).maxCnt = maxCnt;
275     }
276
277     public NumberField(int cols, int maxCnt, int radix){
278         super(cols);
279         ((NumberDocument) getDocument()).maxCnt = maxCnt;
280         ((NumberDocument) getDocument()).radix = radix;
281     }
282
283     public NumberField(String text, int cols, int maxCnt, int radix){
284         super(text, cols);
285         ((NumberDocument) getDocument()).maxCnt = maxCnt;
286         ((NumberDocument) getDocument()).radix = radix;
287         setText(text);
288     }
289
290     protected Document createDefaultModel(){
291         return new NumberDocument();
292     }
293
294     public final int getNumber(){
295         String s = super.getText().trim();
296         if (s.length() == 0)
297             return 0;
298         try{
299             return Integer.parseInt(s, ((NumberDocument) getDocument()).radix);
300         }catch(Exception e){
301             throw new RuntimeException("internal␣error");
302         }
303     }
304
305     public final String getText(){
306         return super.getText().trim();
307     }
308
309     static class NumberDocument extends PlainDocument{
310         int maxCnt;
311         int radix;
```

```
312
313         NumberDocument(){
314             this.maxCnt = -1;
315             this.radix = 10;
316         }
317
318         public void insertString(int offs, String str, AttributeSet a)  throws
                BadLocationException{
319             if (str == null){
320                 return;
321             }
322             for (int i = 0; i < str.length(); i++){
323                 if (Character.digit(str.charAt(i), radix) == -1){
324                     Toolkit.getDefaultToolkit().beep();
325                     return;
326                 }
327             }
328             if ((maxCnt != -1) && ((getLength() + str.length()) > maxCnt)){
329                  Toolkit.getDefaultToolkit().beep();
330                 return;
331             }
332             super.insertString(offs, str, a);
333         }
334     }
335 }
```